```
-- File: MessagingToolBWSDefs.mesa - Created by Loreene Terry.  Last edit:
-- Loreene D. Terry:OSBU South:Xerox 29-Jun-89 13:55:12

-- Copyright (C) 1988, 1989 by Xerox Corporation. All rights reserved.


DIRECTORY
  Context,
  StarWindowShell,
  System,
  Window,
  XFormat,
  XString;


MessagingToolBWSDefs: DEFINITIONS =
BEGIN


  -- ************************************************************
  -- *                          TYPES
  -- ************************************************************


  AddressData: TYPE = ARRAY [1..maxAddresses] OF Addresses;

  Addresses: TYPE = RECORD [name, netAddress: XString.ReaderBody];

  BeepType: TYPE = {everyMessage, once, never};

  Data: TYPE = LONG POINTER TO DataObject;
  DataObject: TYPE = RECORD [
    activateOnReceipt: BOOLEAN ← TRUE,
    addresses: ARRAY [1..maxAddresses] OF Addresses ← nullAddresses,
    beep: BeepType ← once,
    busy: BOOLEAN ← FALSE,
    fileSW: Window.Handle ← NIL,
    formSW: Window.Handle ← NIL,
    message: LONG STRING ← NIL,
    msgSW: Window.Handle ← NIL,
    nAddresses: CARDINAL ← 0,
    notInOffice: BOOLEAN ← FALSE,
    optionSheetFW: Window.Handle ← NIL,
    optionSheetOpen: BOOLEAN ← FALSE,
    out: XFormat.Handle ← NIL,
    outObject: XFormat.Object,
    receiverName: LONG STRING ← NIL,
    recordedMessage: LONG STRING ← NIL,
    shell: StarWindowShell.Handle ← [NIL],
    shouldBeep: BOOLEAN ← TRUE,
    tableWindow: Window.Handle ← NIL,
    to: LONG STRING ← NIL,
    zone: UNCOUNTED ZONE ← NIL];



  -- ************************************************************
  -- *              CONSTANTS AND DATA OBJECTS
  -- ************************************************************


  contextType: Context.Type;
  maxAddresses: CARDINAL = 250;
  nullAddresses: ARRAY [1..maxAddresses] OF Addresses =
    ALL[[XString.nullReaderBody, XString.nullReaderBody]];



  -- ************************************************************
  -- *                       PROCEDURES
  -- ************************************************************


  MakeEditCacheAddressSheet: PROCEDURE [data: Data],
```

```
    ReadAddressesFromFile: PROCEDURE [data: Data];


END.

-- Log (when, who, what) --
 4-May-88 17:45:51 - Terry - Created.
 5-May-88 17:14:19 - Terry - Data: Added more fields for the upcoming Edit Cache Addresses sheet.  Also
 sorted the fields in alphabetical order.
24-Jun-88 10:49:52 - Terry - Addresses: Changed name & netAddress' type to XS.RB.  Data: Removed
cacheFileName.  ReadAddressesFromFile: Added.
29-Jun-89 13:55:11 - LTerry - maxAddresses: Bumped up to 250 from 100.
```

```
-- File: MessagingToolBWSImpl.mesa - last edit:
-- Loreene D. Terry:OSBU South:Xerox 18-Jul-89 15:33:11
-- Breisacher          20-Jul-87 11:35:06
-- Bowers:OSBU South:Xerox 26-Jan-87 14:38:28

-- Copyright (C) 1986, 1987, 1988, 1989 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Attention USING [AddMenuItem, PostAndConfirm],
  BWSZone USING [shortLifetime],
  Event USING [AddDependency, AgentProcedure],
  Format USING [StringProc],
  FormWindow USING [AppendItem, AppendLine, BooleanChangeProc, ChoiceChangeProc, ChoiceItem,
  DoneLookingAtTextItemValue, FreeTextHintsProc, GetZone, LayoutProc, Line, LookAtTextItemValue,
  MakeBooleanItem, MakeChoiceItem, MakeItemsProc, MakeTextItem, NeededDims, NextOutOfProc,
  SetBooleanItemValue, SetChoiceItemValue, SetInputFocus, SetTextItemValue, SetVisibility,
  TextHintsProc],
  Heap USING [Create, systemZone],
  LogFile USING [Create, Close, Handle, PutXString],
  LogStringWindow USING [BackingWriter, ForceOut, MakeLogStringSW, XFormatObject],
  LogStringWindowX USING [Clear],
  MenuData USING [AddItem, CreateItem, CreateMenu, ItemHandle, MenuHandle, MenuProc],
  MessageWindow USING [Clear, PostSTRING],
  MessagingToolBWSDefs,
  MessagingToolCommon USING [AddReceiverName, Deliver, Error, FindPCAddress, FindRecipientAddressProc,
  maxMessageLength, NotifyProc, ReadFileProc, ReceiveMessages, RemoveReceiverName,
  StopReceivingMessages],
  NetworkStream USING [uniqueConnID],
  NSAddressTranslation USING [Error, StringToNetworkAddress],
  NSFile USING [nullReference],
  NSString USING [AppendToMesaString, FreeString, String],
  OptionFile USING [Error, GetBooleanValue, GetStringValue],
  Process USING [Abort, Detach, MsecToTicks, Pause],
  Runtime USING [IsBound],
  SimpleTextDisplay USING [systemFontHeight],
  StarDesktop USING [GetCurrentDesktopFile],
  StarWindowShell USING [Create, EnumeratePopupMenus, Handle, IsCloseLegalProc, MenuEnumProc,
  PoppedProc, Push, SetRegularCommands],
  StarWindowShellExtra5 USING [ManagerFromShell],
  String USING [CopyToNewString, Equivalent, Empty, FreeString, MakeString],
  Subwindower USING [MakeFormSW, MakeMessageSW],
  SubwindowManager USING [ResizeSW],
  TIP USING [UserAbort],
  UserTerminal USING [Beep],
  Window USING [Handle, Object],
  XFormat USING [Handle, Object, String],
  XString USING [ByteLength, CopyReader, CopyToNewWriterBody, Dereference, Empty, Equivalent,
  FreeReaderBytes, FromSTRING, NSStringFromReader, nullReaderBody, Reader, ReaderBody,
  ReaderFromWriter, Writer, WriterBody],
  XStringTableWindow USING [Create],
  XTime USING [Append],
  XToken USING [FreeReaderHandle, FreeTokenString, Handle, MaybeQuoted, NonWhiteSpace, Quote,
  ReaderToHandle, UnterminatedQuote];

MessagingToolBWSImpl: MONITOR
  IMPORTS
    Atom, Attention, BWSZone, Event, FormWindow, Heap, LogFile, LogStringWindow, LogStringWindowX,
    MenuData, MessageWindow, MessagingToolBWSDefs, MessagingToolCommon, NetworkStream,
    NSAddressTranslation, NSString, OptionFile, Process, Runtime, SimpleTextDisplay, StarDesktop,
    StarWindowShell, StarWindowShellExtra5, String, Subwindower, SubwindowManager, TIP, UserTerminal,
    XFormat, XString, XStringTableWindow, XTime, XToken =
BEGIN

  -- TYPEs

  FormItems: TYPE = {activateOnReceipt, beep, to, message, notInOffice, recordedMessage};


  -- Global data

  addressData: MessagingToolBWSDefs.AddressData ← MessagingToolBWSDefs.nullAddresses;
  data: MessagingToolBWSDefs.Data ← NIL;
  fileSWLines: CARDINAL = 25;
  fileSWh: INTEGER = fileSWLines * SimpleTextDisplay.systemFontHeight;
  messageHints, recordedMessageHints, toHints: XString.Reader ← NIL;

MessagingToolBWSImpl.mesa          18-Jul-89 15:33:11 PDT
```

```
    newLogSW, realLogWindow, swmgr: Window.Handle ← NIL;
    zone: UNCOUNTED ZONE ← Heap.Create[initial: 4];



    -- ************************************************************
    -- *                        PROCEDURES
    -- *                    in alphabetical order
    -- ************************************************************



    ActivateChangeProc: FormWindow.BooleanChangeProc =

    <<  FormWindow.BooleanChangeProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey,
    calledBecauseOf: FormWindow.ChangeReason, newValue: BOOLEAN];  >>

    BEGIN
      data.activateOnReceipt ← newValue;
    END;    -- of Procedure ActivateChangeProc




    AddCommandsToAuxMenu: PROC [data: MessagingToolBWSDefs.Data] =
    BEGIN
      editCacheAddresses: XString.ReaderBody ← XString.FromSTRING ["Edit Address List"L];

      MenuEnumProc: StarWindowShell.MenuEnumProc =
      BEGIN
        --  Add the "Edit Address List" command to aux menu.
        MenuData.AddItem[
          menu, MenuData.CreateItem[
          zone: data.zone, name: @editCacheAddresses,
          proc: EditCacheAddressesProc, itemData: data]];

        -- Don't enumerate any other popup menus.
        stop ← TRUE;
      END;    -- of NESTED procedure MenuEnumProc

      -- MAIN CODE for AddCommandsToAuxMenu
      -- ASSUMPTION: First Popup Menu will be Aux menu!
      StarWindowShell.EnumeratePopupMenus[data.shell, MenuEnumProc];
    END;    -- of Procedure AddCommandsToAuxMenu




    AddReceiverName: PROC =
    BEGIN
      --  If the name exists, then add it to the PC Protocol Session in order to receive msgs.
      IF data.receiverName # NIL THEN
        MessagingToolCommon.AddReceiverName[
          name: data.receiverName !
          MessagingToolCommon.Error => IF error = nameInUse THEN CONTINUE];
    END;    -- of Procedure AddReceiverName




    AttemptingLogoffEvent: Event.AgentProcedure =

    <<  Event.AgentProcedure: TYPE = PROCEDURE [event: Event.EventType, eventData: LONG POINTER, myData:
    LONG POINTER] RETURNS [remove: BOOLEAN ← FALSE, veto: BOOLEAN ← FALSE];  >>

    -- This proc is provided in case the Edit Address List psheet was open when user
    -- invokes Logoff.  Logoff closes the psheet automatically so set this internal
    -- variable for a clean logoff...

    BEGIN
      data.optionSheetOpen ← FALSE;
```

```
  END;    -- of Procedure AttemptingLogoffEvent



  BeepChangeProc: FormWindow.ChoiceChangeProc =

  <<  FormWindow.ChoiceChangeProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey,
  calledBecauseOf: FormWindow.ChangeReason, oldValue: FormWindow.ChoiceIndex, newValue:
  FormWindow.ChoiceIndex];  >>

  BEGIN
    --  If the user chose the same value, return.
    IF oldValue = newValue OR calledBecauseOf = client THEN RETURN;

    data.beep ←
      SELECT newValue FROM
        0 => everyMessage,
        1 => once,
        2 => never,
      ENDCASE => once;
  END;    -- of Procedure BeepChangeProc



  ClearLogProc: MenuData.MenuProc =

  <<  MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
  LONG UNSPECIFIED]; >>

  BEGIN
    okay: BOOLEAN ← FALSE;
    question: XString.ReaderBody ← XString.FromSTRING ["Okay to clear log? "L];

    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    --  Ask the user if okay to clear log.
    okay ← Attention.PostAndConfirm [s: @question].confirmed;
    IF NOT okay THEN RETURN;

    --  All clear, go for it!
    LogStringWindowX.Clear[data.fileSW];
  END;    -- of Procedure ClearLogProc



  CloseProc: StarWindowShell.IsCloseLegalProc =

  << StarWindowShell.IsCloseLegalProc: TYPE = PROCEDURE [sws: StarWindowShell.Handle, closeAll: BOOLEAN
  ← FALSE] RETURNS [BOOLEAN]; >>

  BEGIN
    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    --  Check to see if Edit Address List property sheet is open.
    IF data.optionSheetOpen THEN
    BEGIN
      MessageWindow.PostSTRING [data.msgSW, "Edit Address List property sheet is open.  Please close it
      and try again."L];
      RETURN[FALSE];
    END;
    RETURN[TRUE];
  END;    -- of CloseProc



  CreateMessageHints: FormWindow.TextHintsProc =
```

```
<<  FormWindow.TextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey]
RETURNS [hints: LONG DESCRIPTOR FOR ARRAY CARDINAL OF XString.ReaderBody, freeHints:
FormWindow.FreeTextHintsProc, hintAction: FormWindow.TextHintAction ← replace];  >>

--  Stole some code from Lee Breisacher's RunSomeAppls hack.

BEGIN
  StringSeq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF XString.ReaderBody];
  hintSeq: LONG POINTER TO StringSeq ← NIL;
  each, index: CARDINAL ← 0;
  maxHints: CARDINAL = 41;  -- Max for 15" screen; 40 for messages & 1 for clear.
  nullRB, tempRB: XString.ReaderBody ← XString.nullReaderBody;
  tokenHandle: XToken.Handle;

  --  Clear the Message window.
  MessageWindow.Clear [data.msgSW];

  --  Allocate the hint sequence.
  hintSeq ← BWSZone.shortLifetime.NEW[StringSeq [maxHints]];

  --  Make first hint blank so users can clear out the Message: field quickly.
  hintSeq[each] ← XString.Dereference[
      XString.CopyReader[@nullRB, BWSZone.shortLifetime]];
  each ← each + 1;

  IF messageHints # NIL THEN
  BEGIN
    --  Get token handle from reader representing message hints.
    tokenHandle ← XToken.ReaderToHandle [r: messageHints];

    --  Fill in remaining hints with messages.
  FOR index IN [2..maxHints] DO

    --  Get a reader body (tempRB) representing a message.  The message may be a sequence of
    non-white-space characters or a sequence of characters, containing white-space characters,
    enclosed in quotes.
    tempRB ← XToken.MaybeQuoted [
      h: tokenHandle,
      data: NIL,
      filter: XToken.NonWhiteSpace,
      isQuote: XToken.Quote,
      skip: whiteSpace,
      temporary: TRUE ! XToken.UnterminatedQuote =>
        BEGIN
          MessageWindow.PostSTRING [data.msgSW, "ERROR: A closing quote is missing in the
          MessageHints entry of your User Profile's [Messaging Tool] section."L];
          RESUME;
        END;];

    IF XString.Empty[@tempRB] THEN
      BEGIN
        --  We probably hit the end of file so free rb and exit the loop.
        [] ← XToken.FreeTokenString[@tempRB];
        EXIT;
      END
      ELSE
        BEGIN
          --  Copy new hint into array.
          hintSeq[each] ← XString.Dereference[
            XString.CopyReader[@tempRB, BWSZone.shortLifetime]];
          each ← each + 1;
          IF each = maxHints THEN EXIT;

          -- Free token string.
          [] ← XToken.FreeTokenString[@tempRB];
        END;

    ENDLOOP;

    --  Free token handle.
    [] ← XToken.FreeReaderHandle [h: tokenHandle];
  END;

  --  Set input focus in the Message: field.
  FormWindow.SetInputFocus [
```

```
      window: window,
      item: item];

    RETURN[
      hints: DESCRIPTOR[hintSeq, each],
      freeHints: FreeMessageHints,
      hintAction: replace];
  END;    -- of Procedure CreateMessageHints



  CreateRecordedMessageHints: FormWindow.TextHintsProc =

  << FormWindow.TextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey]
  RETURNS [hints: LONG DESCRIPTOR FOR ARRAY CARDINAL OF XString.ReaderBody, freeHints:
  FormWindow.FreeTextHintsProc, hintAction: FormWindow.TextHintAction ← replace]; >>

  --  Stole some code from Lee Breisacher's RunSomeAppls hack.

  BEGIN
    StringSeq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF XString.ReaderBody];
    hintSeq: LONG POINTER TO StringSeq ← NIL;
    each, index: CARDINAL ← 0;
    maxHints: CARDINAL = 41;  -- Max for 15" screen; 40 for recorded messages & 1 for clear.
    nullRB, tempRB: XString.ReaderBody ← XString.nullReaderBody;
    tokenHandle: XToken.Handle;

    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    --  Allocate the hint sequence.
    hintSeq ← BWSZone.shortLifetime.NEW[StringSeq [maxHints]];

    --  Make first hint blank so users can clear out the Message: field quickly.
    hintSeq[each] ← XString.Dereference[
        XString.CopyReader[@nullRB, BWSZone.shortLifetime]];
    each ← each + 1;

    IF recordedMessageHints # NIL THEN
    BEGIN
      --  Get token handle from reader representing message hints.
      tokenHandle ← XToken.ReaderToHandle [r: recordedMessageHints];

      --  Fill in remaining hints with messages.
    FOR index IN [2..maxHints] DO

      --  Get a reader body (tempRB) representing a message.  The message may be a sequence of
      non-white-space characters or a sequence of characters, containing white-space characters,
      enclosed in quotes.
      tempRB ← XToken.MaybeQuoted [
        h: tokenHandle,
        data: NIL,
        filter: XToken.NonWhiteSpace,
        isQuote: XToken.Quote,
        skip: whiteSpace,
        temporary: TRUE ! XToken.UnterminatedQuote =>
          BEGIN
            MessageWindow.PostSTRING [data.msgSW, "ERROR: A closing quote is missing in the
            RecordedMessageHints entry of your User Profile's [Messaging Tool] section."L];
            RESUME;
          END;];

      IF XString.Empty[@tempRB] THEN
        BEGIN
          --  We probably hit the end of file so free rb and exit the loop.
          [] ← XToken.FreeTokenString[@tempRB];
          EXIT;
        END
        ELSE
          BEGIN
            --  Copy new hint into array.
            hintSeq[each] ← XString.Dereference[
              XString.CopyReader[@tempRB, BWSZone.shortLifetime]];
            each ← each + 1;
```

```
            IF each = maxHints THEN EXIT;

            -- Free token string.
            [] ← XToken.FreeTokenString[@tempRB];
          END;

      ENDLOOP;

      --  Free token handle.
      [] ← XToken.FreeReaderHandle [h: tokenHandle];
    END;

    --  Set input focus in the Message: field.
    FormWindow.SetInputFocus [
      window: window,
      item: item];

    RETURN[
      hints: DESCRIPTOR[hintSeq, each],
      freeHints: FreeRecordedMessageHints,
      hintAction: replace];
  END;    -- of Procedure CreateRecordedMessageHints



  CreateToHints: FormWindow.TextHintsProc =

  <<  FormWindow.TextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey]
  RETURNS [hints: LONG DESCRIPTOR FOR ARRAY CARDINAL OF XString.ReaderBody, freeHints:
  FormWindow.FreeTextHintsProc, hintAction: FormWindow.TextHintAction ← replace];  >>

  --  Stole some code from Lee Breisacher's RunSomeAppls hack.

  BEGIN
    StringSeq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF XString.ReaderBody];
    hintSeq: LONG POINTER TO StringSeq ← NIL;
    each, index: CARDINAL ← 0;
    maxHints: CARDINAL = 41;  -- Max for 15" screen; 40 for names & 1 for clear.
    nullRB, tempRB: XString.ReaderBody ← XString.nullReaderBody;
    tokenHandle: XToken.Handle;

    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    --  Allocate the hint sequence.
    hintSeq ← BWSZone.shortLifetime.NEW[StringSeq [maxHints]];

    --  Make first hint blank so users can clear out the To: field quickly.
    hintSeq[each] ← XString.Dereference[
        XString.CopyReader[@nullRB, BWSZone.shortLifetime]];
    each ← each + 1;

    --  If user provided hints for the To's field in the User profile, use them.  Otherwise, use cached
    names.
    IF toHints # NIL THEN
    BEGIN
      --  Get token handle from reader representing to's hints.
      tokenHandle ← XToken.ReaderToHandle [r: toHints];

      --  Fill in remaining hints for the To's field.
    FOR index IN [2..maxHints] DO

      --  Get a reader body (tempRB) representing a message.  The message may be a sequence of
      non-white-space characters or a sequence of characters, containing white-space characters,
      enclosed in quotes.
      tempRB ← XToken.MaybeQuoted [
        h: tokenHandle,
        data: NIL,
        filter: XToken.NonWhiteSpace,
        isQuote: XToken.Quote,
        skip: whiteSpace,
        temporary: TRUE ! XToken.UnterminatedQuote =>
          BEGIN
            MessageWindow.PostSTRING [data.msgSW, "ERROR: A closing quote is missing in the ToHints
```

```
            entry of your User Profile's [Messaging Tool] section."L];
            RESUME;
          END;];

      IF XString.Empty[@tempRB] THEN
        BEGIN
          -- We probably hit the end of file so free rb and exit the loop.
          [] ← XToken.FreeTokenString[@tempRB];
          EXIT;
        END
        ELSE
          BEGIN
            -- Copy new hint into array.
            hintSeq[each] ← XString.Dereference[
              XString.CopyReader[@tempRB, BWSZone.shortLifetime]];
            each ← each + 1;
            IF each = maxHints THEN EXIT;

            -- Free token string.
            [] ← XToken.FreeTokenString[@tempRB];
          END;

      ENDLOOP;

      -- Free token handle.
      [] ← XToken.FreeReaderHandle [h: tokenHandle];
    END
    ELSE
      -- Fill in remaining hints with cached names.
      FOR index IN [1..data.nAddresses] DO
        hintSeq[each] ← XString.Dereference[
          XString.CopyReader[@data.addresses[index].name, BWSZone.shortLifetime]];
        each ← each + 1;
        IF each = maxHints THEN EXIT;
      ENDLOOP;

    -- Set input focus in the To: field.
    FormWindow.SetInputFocus [
      window: window,
      item: item];

    RETURN[
      hints: DESCRIPTOR[hintSeq, each],
      freeHints: FreeToHints,
      hintAction: replace];
  END;   -- of Procedure CreateToHints




  Deliver: ENTRY PROC [toString, text: LONG STRING] =
  BEGIN
    done: BOOLEAN ← FALSE;
    process: PROCESS ← NIL;

    CleanUp: PROC [] =
    BEGIN
      FreeString [s: toString, zone: zone];
      FreeString [s: text, zone: zone];
      data.busy ← FALSE;
    END;   -- of nested Procedure CleanUp

    DeliverInternal: PROC [] =
    BEGIN
      anonymous: LONG STRING = "unknown sender"L;
      MessagingToolCommon.Deliver[
        toList: toString,
        sender: IF NOT String.Empty[data.receiverName] THEN data.receiverName ELSE anonymous,
        text: text, displayProc: Display, findAddressProc: FindRecipientAddress,
        privateDLProc: ExpandPrivateDL !ABORTED => CONTINUE];
      done ← TRUE;
    END;   -- of nested Procedure DeliverInternal

    -- Main code for Deliver
    BEGIN
```

```
      ENABLE UNWIND => CleanUp[];
      process ← FORK DeliverInternal[];
      Process.Detach[process];
      UNTIL done DO
        Process.Pause[ticks: Process.MsecToTicks[msec: 100]];
        IF process # NIL AND (TIP.UserAbort[data.shell] OR TIP.UserAbort[NIL])
          THEN Process.Abort[process: process]; process ← NIL;
        ENDLOOP;
    END;
    CleanUp[];
  END;   -- of Procedure Deliver



  DeliverProc: MenuData.MenuProc =

  << MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
  LONG UNSPECIFIED]; >>

  BEGIN
    to, message: XString.ReaderBody ← XString.nullReaderBody;
    MessageWindow.Clear [data.msgSW];

    -- If messaging tool is busy, post error message in Message window.
    IF data.busy THEN
    BEGIN
      MessageWindow.PostSTRING [data.msgSW, "Messaging Tool is busy."L];
      RETURN;
    END;

    -- Look at the To: and Message: fields.
    to ← FormWindow.LookAtTextItemValue [data.formSW, FormItems.to.ORD];
    message ← FormWindow.LookAtTextItemValue [data.formSW, FormItems.message.ORD];

    -- If the To: field is empty, quit looking at both fields and return.
    -- (It's okay to have an empty Message: field because it allows the user
    -- to determine if others are "listening" without actually sending them
    -- a message.)
    IF XString.Empty [@to] THEN
    BEGIN
      FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.to.ORD];
      FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.message.ORD];
      RETURN;
    END;

    -- If message exceeds the maximum length, post error message in Message window.
    -- Otherwise, signify that Messaging Tool is now busy and fork the Deliver process.
    IF XString.ByteLength [@message] > MessagingToolCommon.maxMessageLength THEN
    BEGIN
      MessageWindow.PostSTRING[data.msgSW, "! The message is too long."L];
    END
    ELSE
      BEGIN
        data.busy ← TRUE;
        Process.Detach[process: FORK Deliver[toString: StringFromReader [@to, zone], text:
        StringFromReader [@message, zone]]];
      END;

    -- Quit looking at the To: and Message: fields.
    FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.to.ORD];
    FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.message.ORD];
  END;   -- of Procedure DeliverProc



  Display: Format.StringProc =
  BEGIN
    data.out.String[s];

    -- Have log window scroll up automatically if necessary.
    LogStringWindow.ForceOut[data.fileSW];
  END;   -- of Procedure Display
```

```
  EditCacheAddressesProc: MenuData.MenuProc =

  <<  MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
  LONG UNSPECIFIED]; >>

  BEGIN
    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    --  Make sure that Table Windows is running so that user won't crash with unbound procedure error
    while the Edit Cache Address sheet is being made.  If it's not running, output error message and
    return.
    IF ~Runtime.IsBound[LOOPHOLE[XStringTableWindow.Create]] THEN
    BEGIN
      MessageWindow.PostSTRING [data.msgSW, "Table Windows is not running!"L];
      RETURN;
    END;

    --  Make the Edit Cache Address sheet if it is not already open.
    IF NOT data.optionSheetOpen THEN MessagingToolBWSDefs.MakeEditCacheAddressSheet [data: data]
    ELSE
    BEGIN
      MessageWindow.PostSTRING [data.msgSW, "Edit Address List property sheet is already open."L];
      RETURN;    -- Don't need this but just in case more code got added after IF clause.
    END;
  END;    -- of Procedure EditCacheAddressesProc




  ExpandPrivateDL: MessagingToolCommon.ReadFileProc =

  <<  MessagingToolCommon.ReadFileProc: TYPE = PROCEDURE [fileName: LONG STRING, zone: UNCOUNTED ZONE]
  RETURNS [fileContent: LONG STRING ← NIL];  >>

  BEGIN
    <<file: MFile.Handle;
    content: MStream.Handle;
    file ← MFile.Acquire[
      name: fileName, access: readOnly, release: [] ! MFile.Error => GOTO done];
    fileContent ← String.MakeString[
      z: zone, maxlength: CARDINAL[MIN[MFile.GetLength[file: file], LAST[CARDINAL]]]];
    -- files greater than LAST[CARDINAL] bytes long will be truncated
    content ← MStream.Create[file: file, release: []];
    UNTIL MStream.EndOf[stream: content] OR fileContent.length = fileContent.maxlength DO
      String.AppendChar[s: fileContent, c: Stream.GetChar[sH: content]];
      ENDLOOP;
    Stream.Delete[sH: content];
    EXITS done => NULL;>>
  END;    -- of Procedure ExpandPrivateDL



  FindRecipientAddress: MessagingToolCommon.FindRecipientAddressProc =
  BEGIN
    index: CARDINAL;
    recipientRB: XString.ReaderBody ← XString.FromSTRING [s: recipient];

    --  Assign an unique ConnectionID to localConnID.
    localConnID ← NetworkStream.uniqueConnID;

    --  If the To: field is either empty or exceeds 86 characters, return.
    IF String.Empty[s: recipient] OR recipient.length > 86
      <<NSName.maxFullNameLength>> THEN RETURN;

    --  Search thru addressData for recipient's name & get network address.
    FOR index IN [1..data.nAddresses] DO
      --  If the current name is NIL, get out of loop.
```

```
      IF data.addresses[index].name = XString.nullReaderBody THEN EXIT
      ELSE
        -- Check to see if recipient matches the current name.
        IF XString.Equivalent[r1: @recipientRB, r2: @data.addresses[index].name] THEN
        BEGIN
          -- Found it!! Convert network address, which is currently a readerbody, into a LONG STRING.
          s: NSString.String ← XString.NSStringFromReader[
            r: @data.addresses[index].netAddress,
            z: zone];

          -- Assign network address, which is now a LONG STRING, to remote and return.
          remote ←
            NSAddressTranslation.StringToNetworkAddress[
              s: s,
              id: NIL,
              defaultDomainOrg: NIL !
              NSAddressTranslation.Error =>
              BEGIN
                MessageWindow.PostSTRING [data.msgSW, "The corresponding cached address is invalid."L];
                CONTINUE;
              END].addr;
          RETURN [remote: remote, localConnID: localConnID];
        END;
    ENDLOOP;

    -- Recipient is not in cache address file, so broadcast on LAN for recipient's name using the XC
    20 name lookup protocol.
    [remote: remote, localConnID: localConnID] ←
      MessagingToolCommon.FindPCAddress[recipient: recipient, sender: sender];
  END;   -- of Procedure FindRecipientAddress



  FreeString: PROCEDURE [s: LONG STRING, zone: UNCOUNTED ZONE] =
  BEGIN
    String.FreeString [z: zone, s: s];
  END;   -- of Procedure FreeString



<<***** [5/24/89]  The following three procedures are identical.  I wonder if I can safely combine them
into a single FreeHints.  May try this later when not making an update so soon.*****>>



  FreeMessageHints: FormWindow.FreeTextHintsProc =

  <<  FormWindow.FreeTextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey,
  hints: LONG DESCRIPTOR FOR ARRAY CARDINAL OF XString.ReaderBody];  >>

  --  Stole code from Lee Breisacher's RunSomeAppls hack.

  BEGIN
    FOR i: CARDINAL IN [0..hints.LENGTH) DO
       XString.FreeReaderBytes[@hints[i], BWSZone.shortLifetime]; ENDLOOP;
    BWSZone.shortLifetime.FREE[@(LOOPHOLE[BASE[hints], LONG POINTER])];
  END;   -- of Procedure FreeMessageHints



  FreeRecordedMessageHints: FormWindow.FreeTextHintsProc =

  <<  FormWindow.FreeTextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey,
  hints: LONG DESCRIPTOR FOR ARRAY CARDINAL OF XString.ReaderBody];  >>

  --  Stole code from Lee Breisacher's RunSomeAppls hack.

  BEGIN
    FOR i: CARDINAL IN [0..hints.LENGTH) DO
```

```
      XString.FreeReaderBytes[@hints[i], BWSZone.shortLifetime]; ENDLOOP;
    BWSZone.shortLifetime.FREE[@(LOOPHOLE[BASE[hints], LONG POINTER])];
  END;   -- of Procedure FreeRecordedMessageHints



  FreeToHints: FormWindow.FreeTextHintsProc =

  <<  FormWindow.FreeTextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey,
  hints: LONG DESCRIPTOR FOR ARRAY CARDINAL OF XString.ReaderBody];  >>

  --  Stole code from Lee Breisacher's RunSomeAppls hack.

  BEGIN
    FOR i: CARDINAL IN [0..hints.LENGTH) DO
      XString.FreeReaderBytes[@hints[i], BWSZone.shortLifetime]; ENDLOOP;
    BWSZone.shortLifetime.FREE[@(LOOPHOLE[BASE[hints], LONG POINTER])];
  END;   -- of Procedure FreeToHints



  GetUserProfileData: PROC [] =
  BEGIN
    activateEntry: XString.ReaderBody ← XString.FromSTRING ["ActivateOnReceipt"L];
    beepEntry: XString.ReaderBody ← XString.FromSTRING ["Beep"L];
    messageHintsEntry: XString.ReaderBody ← XString.FromSTRING ["MessageHints"L];
    nameEntry: XString.ReaderBody ← XString.FromSTRING ["Name"];
    notInOfficeEntry: XString.ReaderBody ← XString.FromSTRING ["NotInOffice"L];
    recordedMsgEntry: XString.ReaderBody ← XString.FromSTRING ["RecordedMessage"L];
    recordedMsgHintsEntry: XString.ReaderBody ← XString.FromSTRING ["RecordedMessageHints"L];
    section: XString.ReaderBody ← XString.FromSTRING ["Messaging Tool"L];
    toHintsEntry: XString.ReaderBody ← XString.FromSTRING ["ToHints"L];

    GetBeep: PROCEDURE [value: XString.Reader] =
    BEGIN
      s: LONG STRING ← StringFromReader [value, zone];

      data.beep ←
        SELECT TRUE FROM
          String.Equivalent[s, "EveryMessage"L] => everyMessage,
          String.Equivalent[s, "Once"L] => once,
          String.Equivalent[s, "Never"L] => never,
        ENDCASE => once;
    END;   -- of nested Procedure GetBeep

    GetMessageHints: PROCEDURE [value: XString.Reader] =
    BEGIN
      IF value # NIL THEN messageHints ← XString.CopyReader [r: value, z: zone];
    END;   -- of nested Procedure GetMessageHints

    GetName: PROCEDURE [value: XString.Reader] =
    BEGIN
      data.receiverName ← StringFromReader [value, zone];
    END;   -- of nested Procedure GetName

    GetRecordedMsg: PROCEDURE [value: XString.Reader] =
    BEGIN
      data.recordedMessage ← StringFromReader [value, zone];
    END;   -- of nested Procedure GetRecordedMsg

    GetRecordedMsgHints: PROCEDURE [value: XString.Reader] =
    BEGIN
      IF value # NIL THEN recordedMessageHints ← XString.CopyReader [r: value, z: zone];
    END;   -- of nested Procedure GetRecordedMsgHints

    GetToHints: PROCEDURE [value: XString.Reader] =
    BEGIN
      IF value # NIL THEN toHints ← XString.CopyReader [r: value, z: zone];
    END;   -- of nested Procedure GetToHints


    --  Extract the Activate On Receipt value from User Profile, if it exists.
```

```
    data.activateOnReceipt ← OptionFile.GetBooleanValue[@section, @activateEntry !
      OptionFile.Error => CONTINUE];

    -- Extract the Beep value from User Profile, if it exists.
    OptionFile.GetStringValue[@section, @beepEntry, GetBeep ! OptionFile.Error => CONTINUE];

    -- Extract the Message's hints value from User Profile, if it exists.
    OptionFile.GetStringValue[@section, @messageHintsEntry, GetMessageHints ! OptionFile.Error =>
    CONTINUE];

    -- Extract the Name value from User Profile, if it exists.
    OptionFile.GetStringValue[@section, @nameEntry, GetName ! OptionFile.Error => CONTINUE];

    -- Extract the Not In Office value from User Profile, if it exists.
    data.notInOffice ← OptionFile.GetBooleanValue[@section, @notInOfficeEntry !
      OptionFile.Error => CONTINUE];

    -- Extract the Recorded Message value from User Profile, if it exists.
    OptionFile.GetStringValue[@section, @recordedMsgEntry, GetRecordedMsg ! OptionFile.Error =>
    CONTINUE];

    -- Extract the Recorded Message's hints value from User Profile, if it exists.
    OptionFile.GetStringValue[@section, @recordedMsgHintsEntry, GetRecordedMsgHints ! OptionFile.Error
    => CONTINUE];

    -- Extract the To's hints value from User Profile, if it exists.
    OptionFile.GetStringValue[@section, @toHintsEntry, GetToHints ! OptionFile.Error => CONTINUE];

  END;   -- of Procedure GetUserProfileData



  Init: PROCEDURE =
  BEGIN
    clearLog: XString.ReaderBody ← XString.FromSTRING ["Clear Log"L];
    deliver: XString.ReaderBody ← XString.FromSTRING ["Deliver"L];
    makeLog: XString.ReaderBody ← XString.FromSTRING ["Make Log"L];
    msgSWLines: CARDINAL = 2;
    msgSWh: INTEGER = msgSWLines * SimpleTextDisplay.systemFontHeight;
    toolName: XString.ReaderBody ← XString.FromSTRING["Messaging Tool"L];
    windowHeaderCommands: ARRAY [0..3) OF MenuData.ItemHandle;
    windowHeaderMenu: MenuData.MenuHandle;

    -- Add "Messaging Tool" to the Attention menu.
    Attention.AddMenuItem [
      MenuData.CreateItem[zone: NIL, name: @toolName, proc: ShowShellMenuProc]];

    -- Allocate data.
    data ← zone.NEW[MessagingToolBWSDefs.DataObject];
    data.zone ← zone;

    -- Create the shell for Messaging Tool and use subwindows.
    data.shell ← StarWindowShell.Create [
      name: @toolName,
      considerShowingCoverSheet: FALSE,
      zone: zone,
      isCloseLegalProc: CloseProc];
    swmgr ← StarWindowShellExtra5.ManagerFromShell [data.shell];

    -- Make window header commands.
    windowHeaderCommands ←
      [MenuData.CreateItem[
        zone: zone,
        name: @deliver,
        proc: DeliverProc,
        itemData: data],
       MenuData.CreateItem[
        zone: zone,
        name: @makeLog,
        proc: MakeLogProc,
        itemData: data],
       MenuData.CreateItem[
        zone: zone,
        name: @clearLog,
```

```
        proc: ClearLogProc,
        itemData: data]];

    -- Create menu for window header commands.
    windowHeaderMenu ← MenuData.CreateMenu[
      zone: zone,
      title: NIL,
      array: DESCRIPTOR[windowHeaderCommands]];

    -- Set the tool's window header commands.
    StarWindowShell.SetRegularCommands[sws: data.shell, commands: windowHeaderMenu];

    -- Add additional commands to the tool's aux menu.
    AddCommandsToAuxMenu[data];

    -- Make the message subwindow and fill in data's msgSW.
    data.msgSW ← Subwindower.MakeMessageSW [swmanager: swmgr, lines: msgSWLines, zone: zone,
    horizScrollbar: FALSE, size: msgSWh];

    -- Make the form subwindow, fill in data's formSW, and resize fw's height to be larger.
    data.formSW ← Subwindower.MakeFormSW [swmanager: swmgr, makeItemsProc: MakeFormSWItems, layoutProc:
    LayoutFormSW, zone: zone, horizScrollbar: FALSE];
    [] ← SubwindowManager.ResizeSW [data.formSW, FormWindow.NeededDims[data.formSW].h + 100];

    -- Make log subwindow and fill in some of data's other fields.
    data.fileSW ← LogStringWindow.MakeLogStringSW [swmanager: swmgr, size: fileSWh, zone: zone];
    data.outObject ← LogStringWindow.XFormatObject [data.fileSW];
    data.out ← @data.outObject;

    -- Register LogonEvent/LogoffEvent/AttemptingLogoffEvent procs that are to be called when
    Logon/Logoff occurs.
    [] ← Event.AddDependency [agent: LogonEvent, myData: NIL, event: Atom.MakeAtom["Logon"L]];
    [] ← Event.AddDependency [agent: LogoffEvent, myData: NIL, event: Atom.MakeAtom["Logoff"L]];
    [] ← Event.AddDependency [agent: AttemptingLogoffEvent, myData: NIL, event:
    Atom.MakeAtom["AttemptingLogoff"L]];

    -- If user already has access to a desktop, call LogonEvent.
    IF StarDesktop.GetCurrentDesktopFile[] # NSFile.nullReference THEN [] ← LogonEvent[Atom.null, NIL,
    NIL];

    <<IF user is logged on, then ShowShell[]>>
  END;   -- of Procedure Init



  LayoutFormSW: FormWindow.LayoutProc =
  BEGIN
    margin: CARDINAL = 10;
    line: FormWindow.Line;

    -- Line 1
    line ← FormWindow.AppendLine [window, margin];
    FormWindow.AppendItem [window, FormItems.activateOnReceipt.ORD, line, margin];
    FormWindow.AppendItem [window, FormItems.beep.ORD, line, margin];
    FormWindow.AppendItem [window, FormItems.notInOffice.ORD, line, margin];

    -- Line 2
    line ← FormWindow.AppendLine [window, margin];
    FormWindow.AppendItem [window, FormItems.to.ORD, line, margin];

    -- Line 3
    line ← FormWindow.AppendLine [window, margin];
    FormWindow.AppendItem [window, FormItems.message.ORD, line, margin];

    -- Line 4
    line ← FormWindow.AppendLine [window, margin];
    FormWindow.AppendItem [window, FormItems.recordedMessage.ORD, line, margin];
  END;   -- of Procedure LayoutFormSW



  LogoffEvent: Event.AgentProcedure =
```

```
<<  Event.AgentProcedure: TYPE = PROCEDURE [event: Event.EventType, eventData: LONG POINTER, myData:
LONG POINTER] RETURNS [remove: BOOLEAN ← FALSE, veto: BOOLEAN ← FALSE];  >>

BEGIN
  MessagingToolCommon.StopReceivingMessages[];
  RemoveReceiverName[];
END;   -- of Procedure LogoffEvent



LogonEvent: Event.AgentProcedure =

<<  Event.AgentProcedure: TYPE = PROCEDURE [event: Event.EventType, eventData: LONG POINTER, myData:
LONG POINTER] RETURNS [remove: BOOLEAN ← FALSE, veto: BOOLEAN ← FALSE];  >>

BEGIN
  --  Reset messageHints and toHints to NIL before reading data from User Profile.
  messageHints ← NIL;
  toHints ← NIL;

  --  Read Messaging Tool's entries from User Profile.
  GetUserProfileData[];

  --  Set Activate On Receipt.
  FormWindow.SetBooleanItemValue[
    window: data.formSW,
    item: FormItems.activateOnReceipt.ORD,
    newValue: data.activateOnReceipt,
    repaint: FALSE];

  --  Set Beep.
  FormWindow.SetChoiceItemValue[
    window: data.formSW,
    item: FormItems.beep.ORD,
    newValue: data.beep.ORD,
    repaint: FALSE];

  --  Set Not In Office & repaint form window.
  FormWindow.SetBooleanItemValue[
    window: data.formSW,
    item: FormItems.notInOffice.ORD,
    newValue: data.notInOffice,
    repaint: TRUE];

  --  Read cache addresses from a file, if it exists.
  MessagingToolBWSDefs.ReadAddressesFromFile [data: data];

  --  Begin listening for user's new messages.
  MessagingToolCommon.ReceiveMessages[
    displayProc: Display, notifyProc: NotifyNewMessage];

  --  Add user's name to the PC Protocol Session.
  AddReceiverName[];
END;   -- of Procedure LogonEvent



MakeFormSWItems: FormWindow.MakeItemsProc =
BEGIN
  fwz: UNCOUNTED ZONE = FormWindow.GetZone[window];
  rb: XString.ReaderBody;

  ReaderFromString: PROCEDURE [s: LONG STRING] RETURNS [r: XString.Reader] = INLINE
  BEGIN
    rb ← XString.FromSTRING [s]; RETURN [@rb];
  END;   -- of nested Procedure ReaderFromString

  --  Make a boolean item for Activate On Receipt.
  FormWindow.MakeBooleanItem [
    window: window,
    myKey: FormItems.activateOnReceipt.ORD,
```

```
      changeProc: ActivateChangeProc,
      label: [string[ReaderFromString ["Activate On Receipt"L]↑]],
      initBoolean: data.activateOnReceipt];

    -- Make a choice item for Beep.
    BEGIN
      choice0: XString.ReaderBody ← XString.FromSTRING ["every message"L];
      choice1: XString.ReaderBody ← XString.FromSTRING ["once"L];
      choice2: XString.ReaderBody ← XString.FromSTRING ["never"L];
      choices: ARRAY [0..3) OF FormWindow.ChoiceItem ← [
        [string[0, choice0]],
        [string[1, choice1]],
        [string[2, choice2]]];
      tag: XString.ReaderBody ← XString.FromSTRING ["Beep"L];

      FormWindow.MakeChoiceItem [
        window: window,
        myKey: FormItems.beep.ORD,
        tag: @tag,
        values: DESCRIPTOR[choices],
        changeProc: BeepChangeProc,
        initChoice: data.beep.ORD,
        fullyDisplayed: FALSE]
    END;

    -- Make a boolean item for Not In Office.
    FormWindow.MakeBooleanItem [
      window: window,
      myKey: FormItems.notInOffice.ORD,
      changeProc: NotInOfficeChangeProc,
      label: [string[ReaderFromString ["Not In Office"L]↑]],
      initBoolean: data.notInOffice];

    -- Make a text item for To: field.
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.to.ORD,
      tag: ReaderFromString ["To:"L],
      width: 400,
      hintsProc: CreateToHints,
      nextOutOfProc: NextOutOfToProc,
      visibility: IF data.notInOffice THEN invisible ELSE visible];

    -- Make a text item for Message: field.
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.message.ORD,
      tag: ReaderFromString ["Message:"L],
      width: 400,
      hintsProc: CreateMessageHints,
      nextOutOfProc: NextOutOfMessageProc,
      visibility: IF data.notInOffice THEN invisible ELSE visible];

    -- Make a text item for Recorded Message: field.
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.recordedMessage.ORD,
      tag: ReaderFromString ["Recorded Message:"L],
      width: 400,
      hintsProc: CreateRecordedMessageHints,
      visibility: IF data.notInOffice THEN visible ELSE invisible];

  END;   -- of Procedure MakeFormSWItems



  MakeLogProc: MenuData.MenuProc =

  << MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
  LONG UNSPECIFIED]; >>

  BEGIN
    handle: LogFile.Handle;
    name: XString.ReaderBody ← XString.FromSTRING ["MessagingTool Log of "L];
```

```
    w: XString.Writer ← LogStringWindow.BackingWriter[data.fileSW];
    r: XString.Reader ← XString.ReaderFromWriter[w];
    wbName: XString.WriterBody;

    -- Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    -- Allocate document's name from a zone.
    wbName ← XString.CopyToNewWriterBody[r: @name, z: data.zone, extra: 50];

    -- Append the current time to the prefix.
    XTime.Append[@wbName];

    -- Get a readerbody which represents the document's complete name.
    name ← XString.ReaderFromWriter[@wbName]↑;

    -- Use Deb Lewis' hack for a painless way of creating a simple text document log.
    handle ← LogFile.Create[name: @name];
    LogFile.PutXString[handle: handle, s: r];
    LogFile.Close[handle: handle];
  END;   -- of Procedure MakeLogProc



NextOutOfMessageProc: FormWindow.NextOutOfProc =

<<  FormWindow.NextOutOfProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey]; >>

--  Just about all the code were copied from DeliverProc.  However, the only difference is that the
Message: field is cleared out at the end if the message was not too long.

BEGIN
  to, message, nullRB: XString.ReaderBody ← XString.nullReaderBody;
  messageTooLong: BOOLEAN ← FALSE;

  --  Clear the Message window.
  MessageWindow.Clear [data.msgSW];

  --  If messaging tool is busy, post error message in Message window.
  IF data.busy THEN
  BEGIN
    MessageWindow.PostSTRING [data.msgSW, "Messaging Tool is busy."L];
    RETURN;
  END;

  --  Look at the To: and Message: fields.
  to ← FormWindow.LookAtTextItemValue [data.formSW, FormItems.to.ORD];
  message ← FormWindow.LookAtTextItemValue [data.formSW, FormItems.message.ORD];

  --  If the To: field is empty, quit looking at both fields and return.
  -- (It's okay to have an empty Message: field because it allows the user
  -- to determine if others are "listening" without actually sending them
  -- a message.)
  IF XString.Empty [@to] THEN
  BEGIN
    FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.to.ORD];
    FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.message.ORD];
    RETURN;
  END;

  --  If message exceeds the maximum length, post error message in Message window.
  -- Otherwise, signify that Messaging Tool is now busy and fork the Deliver process.
  IF XString.ByteLength [@message] > MessagingToolCommon.maxMessageLength THEN
  BEGIN
    MessageWindow.PostSTRING[data.msgSW, "! The message is too long."L];
    messageTooLong ← TRUE;
  END
  ELSE
    BEGIN
      data.busy ← TRUE;
      Process.Detach[process: FORK Deliver[toString: StringFromReader [@to, zone], text:
      StringFromReader [@message, zone]]];
    END;
```

```
    --  Quit looking at the To: and Message: fields.
    FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.to.ORD];
    FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.message.ORD];

    --  Clear out the Message: field if the message wasn't too long.
    -- (At this point, there's no easy way to tell whether the delivery failed.)
    IF ~messageTooLong THEN
       FormWindow.SetTextItemValue [
         window: data.formSW,
         item: FormItems.message.ORD,
         newValue: @nullRB,
         repaint: TRUE];
  END;    -- of Procedure NextOutOfMessageProc



  NextOutOfToProc: FormWindow.NextOutOfProc =

  <<  FormWindow.NextOutOfProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey]; >>

  BEGIN
    nullRB: XString.ReaderBody ← XString.nullReaderBody;

    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    --  Clear out the Message: field.
    FormWindow.SetTextItemValue [
      window: data.formSW,
      item: FormItems.message.ORD,
      newValue: @nullRB,
      repaint: TRUE];

    --  Set the input focus in the Message: field.
    FormWindow.SetInputFocus [
      window: data.formSW,
      item: FormItems.message.ORD];
  END;    -- of Procedure NextOutOfToProc



  NopPopProc: StarWindowShell.PoppedProc =
  BEGIN
    data.shouldBeep ← TRUE;
  END;    -- of Procedure NopPopProc



  NotifyNewMessage: MessagingToolCommon.NotifyProc =

  <<  MessagingToolCommon.NotifyProc: TYPE = PROCEDURE RETURNS [sendResponse: BOOLEAN ← FALSE,
  responseText: LONG STRING ← NIL];  >>

  BEGIN
    beepDuration: CARDINAL = 200;  -- milliseconds
    beepFrequency: CARDINAL = 1500;  -- Hertz

    --  If Not In Office is on, then prepare the response text to send back to original sender.
    IF data.notInOffice THEN
    BEGIN
      recordedMessage: XString.ReaderBody ← XString.nullReaderBody;
      s: LONG STRING;
      sendResponse ← TRUE;
      recordedMessage ← FormWindow.LookAtTextItemValue [data.formSW, FormItems.recordedMessage.ORD];
      s ← StringFromReader [@recordedMessage, zone];
      -- First use text in the Recorded Message field if it's not empty;
      -- otherwise use the default recorded message from User Profile.  [If user
      -- didn't specify a default recording, responseText remains NIL and the
      -- internal MessagingToolCommonImpl will say "I'm not in my office.".]
      IF ~XString.Empty [@recordedMessage] THEN responseText ← String.CopyToNewString [s: s, z:
```

```
      Heap.systemZone]
      ELSE IF data.recordedMessage # NIL THEN responseText ← String.CopyToNewString [s:
      data.recordedMessage, z: Heap.systemZone];
      FreeString [s, zone];
      FormWindow.DoneLookingAtTextItemValue [data.formSW, FormItems.recordedMessage.ORD];
    END;

    --  If Activate On Receipt is on, "activate" the Messaging Tool shell.
    IF data.activateOnReceipt THEN ShowShell[];

    --  If Beep is set to every message, beep.
    IF data.beep = everyMessage THEN UserTerminal.Beep[frequency: beepFrequency, duration:
    beepDuration];

    --  If this is the first message since the shell has been closed, check to see if Beep is set to
    once.  If so, beep.
    IF data.shouldBeep THEN
      BEGIN
      IF data.beep = once THEN UserTerminal.Beep[frequency: beepFrequency, duration: beepDuration];
      data.shouldBeep ← FALSE;
      END;
  END;    -- of Procedure NotifyNewMessage



  NotInOfficeChangeProc: FormWindow.BooleanChangeProc =

  <<  FormWindow.BooleanChangeProc: TYPE = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey,
  calledBecauseOf: FormWindow.ChangeReason, newValue: BOOLEAN];  >>

  BEGIN
    data.notInOffice ← newValue;

    -- If Not In Office is on, make To & Message invisible and Recorded Message visible.
    -- Otherwise, make Recorded Message invisible and To & Message visible.

    FormWindow.SetVisibility[
      window: data.formSW,
      item: FormItems.to.ORD,
      visibility: IF newValue THEN invisible ELSE visible,
      repaint: FALSE];

    FormWindow.SetVisibility[
      window: data.formSW,
      item: FormItems.message.ORD,
      visibility: IF newValue THEN invisible ELSE visible,
      repaint: FALSE];

    FormWindow.SetVisibility[
      window: data.formSW,
      item: FormItems.recordedMessage.ORD,
      visibility: IF newValue THEN visible ELSE invisible,
      repaint: calledBecauseOf = user];
  END;    -- of Procedure NotInOfficeChangeProc



  RemoveReceiverName: ENTRY PROC =
  BEGIN
    IF data.receiverName # NIL THEN
    BEGIN
      MessagingToolCommon.RemoveReceiverName[name: data.receiverName];
      FreeString[s: data.receiverName, zone: zone];
    END;
  END;    -- of Procedure RemoveReceiverName



  ShowShell: PROCEDURE =
  BEGIN
```

```
    StarWindowShell.Push [newShell: data.shell, poppedProc: NopPopProc];
  END;    -- of Procedure ShowShell



  ShowShellMenuProc: MenuData.MenuProc =
  BEGIN
    ShowShell[];
  END;    -- of Procedure ShowShellMenuProc



  StringFromReader: PROCEDURE [r: XString.Reader, zone: UNCOUNTED ZONE] RETURNS [s: LONG STRING] =

  --  This procedure first converts the XString.Reader into a NSString.String, makes a new LONG STRING,
  appends the NSString.String's contents onto the LONG STRING, and frees the NSString.String.

  BEGIN
    ns: NSString.String ← XString.NSStringFromReader [r: r, z: zone];
    s ← String.MakeString [z: zone, maxlength: ns.maxlength];
    NSString.AppendToMesaString [to: s, from: ns];
    NSString.FreeString [z: zone, s: ns];
  END;    -- of Procedure StringFromReader



  -- Mainline code

  Init[];

END.


-- Log (when, who, what) --
26-Jan-87 10:59:08 - Bowers - added beep and activateOnReceipt
 6-Mar-87 - LFB - Adapted for BWS. Receive-only for now.
22-May-87 - LFB - Add crude send capability.
26-Feb-88 16:20:47 - Terry - Fully upgraded it to have the same functionality as the latest XDE
version.
25-Apr-88 15:35:38 - Terry - maxAddresses: Changed limit from 40 to 100.
 4-May-88 17:48:45 - Terry - Moved most types to MessagingToolBWSDefs.
24-Jun-88 14:52:35 - Terry - EditCacheAddressesProc: Add Runtime.IsBound check for Table Windows.  Now
reading in cache addresses from a file, if it exists, in the EditCacheAddressPSheet module.
 1-Jul-88 15:08:13 - Terry - Provide CreateToHints for the To field.  FindRecipientAddress: Catch
 NSAddressTranslation.Error if corresponding cached address is invalid.
 5-Jul-88 12:34:12 - Terry - NextOutOfMessageProc: Added to allow user to NEXT out of Message field to
 deliver a msg.  StringFromReader & FreeString: Removed amazing piece of kludge and used standard
 interfaces to convert XS.R into a LS.
 6-Jul-88 17:24:17 - Terry - Allows user to specify hints for To & Message fields.  If user does not
 specify hints for the To field, use cached names.
 7-Jul-88 11:03:32 - Terry - CreateMessageHints & CreateToHints: Set input focus in text item.
 NextOutOfToProc: Added to clear Message field when user NEXTs out of To field.
11-Aug-88 15:22:37 - Terry - Create*Hints: EXIT if each = maxHints.  FindRecipientAddress: Use
XString.Equivalent instead of Equal so To: field is now case insensitive.
25-Aug-88 16:39:23 - Terry - NextOutOfMessageProc: Don't clear out message if it was too long.
10-Apr-89 19:34:25 - LTerry - Init: Changed editCacheAddresses to be "Edit Address List".
17-May-89 17:27:52 - LTerry - LogonEvent: Appended "InSystemFolder" to
MessagingToolBWSDefs.ReadAddressesFromFile call.
18-May-89 18:25:29 - LTerry - LogonEvent: Undid 5/17 action.
19-May-89 17:46:55 - LTerry - Added new window header commands: Clear Log and Make Log.  Also added
Recorded Message text item.
22-May-89  9:44:59 - LTerry - LayoutFormSW: Laid out Recorded Message item.
23-May-89 15:10:39 - LTerry - MakeLogProc: Threw in few lines of code to create simple text document
log during idle time (my SUN was rebuilding s/w).
24-May-89 19:25:41 - LTerry - Allowed Recorded Message hints and read from UserProfile.
EditCacheAddressesProc: Don't make psheet if already open.  Create*Hints: Began dummy-proofing
UserProfile entries by informing users of missing closing quote instead of crashing.  CloseProc: Added
and checked if Edit Address List psheet is open when user closes tool.  Post all informative messages
in the message sw always instead of to Attn window sometimes.
25-May-89 12:56:05 - LTerry - NotInOfficeChangeProc & MakeFormSWItems: Made To & Message's visibility
dependent on Not In Office's value.  AddCommandsToAuxMenu: Added & moved some wh commands to aux menu.
```

31-May-89 20:05:42 - LTerry - NotifyNewMessage: If Recorded Message field is empty, use the default recording from the UserProfile if the user provided it. ClearLogProc: Removed comments and started working on the implementation again.
 8-Jun-89 12:42:27 - LTerry - ClearLogProc: Added Window.InsertIntoTree call.
21-Jun-89 18:59:52 - LTerry - ClearLogProc: Undid 6/8 work.
23-Jun-89 19:37:10 - LTerry - ClearLogProc: Continued struggling. Init: Pass considerShowingCoverSheet = FALSE in SWS.Create call.
27-Jun-89 16:55:05 - LTerry - ClearLogProc: Made newLogSW global to see if that would do the trick.
28-Jun-89 17:56:55 - LTerry - ClearLogProc: Finally implemented Clear in LogStringImpl and called it per JPhillips' suggestion.
16-Jul-89 16:27:32 - LTerry - AttemptingLogoffEvent: Set data.optionSheetOpen to FALSE so user can log off cleanly. Init: Registered AttemptingLogoffEvent and put Make Log back in window header. AddCommandsToAuxMenu: Removed Make Log.
18-Jul-89 15:32:52 - LTerry - Init: Fixed VPPM typo on "attemptingLogoff" for Atom.MakeAtom call.

```
-- File: EditCacheAddressPSheet.mesa - Created by Loreene Terry.  Last edit:
-- Loreene D. Terry:OSBU South:Xerox 16-Jul-89 16:24:18

-- Copyright (C) 1988, 1989 by Xerox Corporation. All rights reserved.


-- ***************************************************************************
-- *
-- *     EditCacheAddressPSheet exports the following procedures:
-- *       o  MakeEditCacheAddressSheet
-- *       o  ReadAddressesFromFile
-- *
-- ***************************************************************************


DIRECTORY
  Attention USING [PostAndConfirm],
  BodyWindowParent USING [GetInteriorDims, ParentFromBody],
  BWSFileTypes USING [systemFileCatalog],
  Catalog USING [Open],
  FormWindow USING [AppendItem, AppendLine, LayoutProc, Line, MakeItemsProc, MakeWindowItem,
  SetTabStops, TabStops],
  FormWindowExtra USING [SetWindowItemSizeExtra],
  LogFile USING [Close, Create, Handle, PutXChar, PutXString],
  MenuData USING [AddItem, CreateItem, MenuHandle, MenuProc],
  MessageWindow USING [Clear, PostSTRING],
  MessagingToolBWSDefs USING [Data, maxAddresses],
  NSAssignedTypes USING [tText],
  NSFile USING [Attribute, ChangeAttributes, Close, Create, Delete, Error, Filter, Find, Handle, Move,
  nullHandle, OpenByReference, Reference, Scope, String, Type],
  NSFileStream USING [Create, Handle, SetLength],
  NSString USING [String, StringFromMesaString],
  PropertySheet USING [Create, GetFormWindows, MenuItemProc],
  Selection USING [Convert, Clear, Value],
  StarWindowShell USING [EnumeratePopupMenus, Handle, MenuEnumProc, ShellFromChild],
  Stream USING [Delete, GetPosition, Handle],
  TableWindow USING [Cell, CellBox, GetMinDims, nullCell, NumberofRowsAndColumns, RedisplayArea,
  SetColumnWidth],
  Window USING [Box, Dims, GetBox, Handle, Object, Place, SlideAndSize, ValidateTree],
  XChar USING [Character, Make],
  XCharSet0 USING [Codes0],
  XCharSets USING [Sets],
  XFormat USING [Char, CR, Handle, Object, ReaderBody, StreamObject],
  XString USING [CopyToNewReaderBody, CopyToNewWriterBody, Empty, FreeReaderBytes, FromSTRING,
  nullReaderBody, Reader, ReaderBody, ReaderFromWriter, WriterBody],
  XStringTableWindow USING [AppendRows, CanDeleteProc, CanSelectProc, Cell, CellContentProc,
  CellNotifyProc, Create, DeleteColumns, DeleteRows, DestroyProc, EnumerateCells, HasAnyBeenChanged,
  MinDimsChangeProc, NextOutOfProc, Options, SetCell, SetCellSelection, SetInputFocus],
  XStringTableWindowExtra1 USING [AssignFALSEToAnyChanged],
  XTime USING [Append],
  XToken USING [Filtered, FreeStreamHandle, FreeTokenString, Handle, Line, NonWhiteSpace,
  StreamToHandle];


EditCacheAddressPSheet: PROGRAM
  IMPORTS Attention, BodyWindowParent, Catalog, FormWindow, FormWindowExtra, LogFile, MenuData,
  MessageWindow, NSFile, NSFileStream, NSString, PropertySheet, Selection, StarWindowShell, Stream,
  TableWindow, Window, XChar, XFormat, XString, XStringTableWindow, XStringTableWindowExtra1, XTime,
  XToken
  EXPORTS MessagingToolBWSDefs =

  BEGIN

  -- TYPEs

  Items: TYPE = {tableWindow};



  -- ***********************************************************
  -- *             EDIT CACHE ADDRESS PUBLIC PROCEDURES
  -- *                   in alphabetical order
  -- ***********************************************************
```

```
MakeEditCacheAddressSheet: PUBLIC PROCEDURE [data: MessagingToolBWSDefs.Data] =
BEGIN
  placeToDisplay: Window.Place ← [670, 450];
  pSheet: StarWindowShell.Handle;
  shellDims: Window.Dims ← [460, 400];
  title: XString.ReaderBody ← XString.FromSTRING ["Edit Address List"L];

  -- Create the Edit Cache Address property sheet.
  pSheet ← PropertySheet.Create[
    formWindowItems: MakeItemsProc,
    menuItemProc: MenuItemProc,
    menuItems: [
      done: TRUE, apply: TRUE, cancel: TRUE,
      defaults: FALSE, start: FALSE, reset: FALSE],
    size: shellDims,
    title: @title,
    placeToDisplay: placeToDisplay,
    formWindowItemsLayout: LayoutProc,
    windowAttachedTo: data.shell,
    display: TRUE,
    clientData: data];

  -- Add commands to option sheet's aux menu.
  AddCommandsToAuxMenu[shell: pSheet, data: data];

  -- Set boolean to signify that the option sheet is currently open.
  data.optionSheetOpen ← TRUE;
END;  -- of Procedure MakeEditCacheAddressSheet



ReadAddressesFromFile: PUBLIC PROCEDURE [data: MessagingToolBWSDefs.Data] =

-- This procedure reads in cache addresses, which consists of a network address and a name, from a
file.

BEGIN
  cacheHandle, systemFolder, tempHandle: NSFile.Handle ← NSFile.nullHandle;
  cacheName: NSString.String ← NSString.StringFromMesaString["MessagingToolAddresses.cache"L];
  cacheFilterList: ARRAY [0..1) OF NSFile.Filter ← [[equal[[name[cacheName]]]]];
  cacheFilter: NSFile.Filter ← [and[DESCRIPTOR[cacheFilterList]]];
  cacheScope: NSFile.Scope ← [filter: cacheFilter];
  index: CARDINAL;
  nameAttr: ARRAY [0..1) OF NSFile.Attribute ← [[name[cacheName]]];
  streamHandle: NSFileStream.Handle ← NSFileStream.Handle[NIL];
  tempName: NSString.String ← NSString.StringFromMesaString["TempMessagingToolAddresses.cache"L];
  tempFilterList: ARRAY [0..1) OF NSFile.Filter ← [[equal[[name[tempName]]]]];
  tempFilter: NSFile.Filter ← [and[DESCRIPTOR[tempFilterList]]];
  tempAddressRB, tempNameRB: XString.ReaderBody;
  tempScope: NSFile.Scope ← [filter: tempFilter];
  tokenHandle: XToken.Handle;

  BEGIN
    ENABLE
      UNWIND =>
        BEGIN
          IF streamHandle # NSFileStream.Handle[NIL] THEN Stream.Delete[sH: streamHandle];
          IF cacheHandle # NSFile.nullHandle THEN NSFile.Close[file: cacheHandle];
          IF tempHandle # NSFile.nullHandle THEN NSFile.Close[file: tempHandle];
          IF systemFolder # NSFile.nullHandle THEN NSFile.Close[file: systemFolder];
        END;

    -- Open up System Folder.
    systemFolder ← Catalog.Open[BWSFileTypes.systemFileCatalog];

    -- Find & get a handle to temporary file if it exists.
    tempHandle ← NSFile.Find[directory: systemFolder, scope: tempScope ! NSFile.Error => CONTINUE; ];

    -- Find & get a handle to Messaging Tool Addresses cache file.  If it couldn't be found then set
    some minimal values, close the system folder, and get outta here!
    cacheHandle ← NSFile.Find[directory: systemFolder, scope: cacheScope ! NSFile.Error =>
      IF tempHandle = NSFile.nullHandle THEN
      BEGIN
```

```
        nullRB: XString.ReaderBody ← XString.nullReaderBody;
        data.addresses[1] ← [nullRB, nullRB];
        data.nAddresses ← 1;
        IF systemFolder # NSFile.nullHandle THEN NSFile.Close[file: systemFolder];
        GOTO ImGettingOutOfHere;
      END
      ELSE CONTINUE;];

    --  If a temporary file has been found then the workstation has just been rebooted after a crash,
    so clean up.
    IF tempHandle # NSFile.nullHandle THEN
    BEGIN
      --  Delete the old file if it exists.
      IF cacheHandle # NSFile.nullHandle THEN NSFile.Delete[file: cacheHandle];

      --  Rename temp file.
      NSFile.ChangeAttributes[file: tempHandle, attributes: DESCRIPTOR[nameAttr]];

      --  Make assignment to cacheHandle so later code can use it.
      cacheHandle ← tempHandle;
      tempHandle ← NSFile.nullHandle;
    END;

    --  Get stream handle to cache file.
    streamHandle ← NSFileStream.Create[file: cacheHandle, closeOnDelete: FALSE];

    --  Get token handle from the stream.
    tokenHandle ← XToken.StreamToHandle[streamHandle];

    --  Read cache file and fill in the address data.
    FOR index IN [1..MessagingToolBWSDefs.maxAddresses] DO

      --  Get a reader body (rb) representing a network address.
      tempAddressRB ← XToken.Filtered [
        h: tokenHandle,
        data: NIL,
        filter: XToken.NonWhiteSpace,
        skip: none,
        temporary: TRUE];

      --  Get a reader body (rb) representing a name.
      tempNameRB ← XToken.Filtered [
        h: tokenHandle,
        data: NIL,
        filter: XToken.Line,
        skip: none,
        temporary: TRUE];

      --  If both the name and address were blank, we probably hit the end
      --  of the file so free readerbodies and exit the loop.
      IF XString.Empty[@tempAddressRB] AND XString.Empty[@tempNameRB] THEN
      BEGIN
        [] ← XToken.FreeTokenString[@tempAddressRB];
        [] ← XToken.FreeTokenString[@tempNameRB];
        IF data.nAddresses = 0 THEN data.nAddresses ← 1;  -- check for empty file!
        EXIT;
      END;

      --  Copy the network address and store it into addressData's array.
      data.addresses[index].netAddress ← XString.CopyToNewReaderBody [
        r: @tempAddressRB,
        z: data.zone];

      --  Copy the name and store it into addressData's array.
      data.addresses[index].name ← XString.CopyToNewReaderBody [
        r: @tempNameRB,
        z: data.zone];

      --  Update data's nAddresses.
      data.nAddresses ← index;

      --  Free token strings.
      [] ← XToken.FreeTokenString[@tempAddressRB];
      [] ← XToken.FreeTokenString[@tempNameRB];

    ENDLOOP;
```

```
      -- Free token handle.
      [] ← XToken.FreeStreamHandle[tokenHandle];

      -- Delete stream handle.
      Stream.Delete[sH: streamHandle];

      -- Close cache file and System folder.
      NSFile.Close[cacheHandle];
      NSFile.Close[systemFolder];

    END;  -- enable

    EXITS ImGettingOutOfHere => NULL;

  END;  -- of Procedure ReadAddressesFromFile




  -- ************************************************************
  -- *                     PRIVATE PROCEDURES
  -- *                   in alphabetical order
  -- ************************************************************



  AddAddressesProc: MenuData.MenuProc =

<<  MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
LONG UNSPECIFIED]; >>

  BEGIN
    cacheHandle: NSFile.Handle ← NSFile.nullHandle;
    data: MessagingToolBWSDefs.Data ← itemData;
    okay: BOOLEAN ← FALSE;
    question: XString.ReaderBody ← XString.FromSTRING ["Okay to add to existing address list?"L];
    ref: LONG POINTER TO NSFile.Reference;
    value: Selection.Value;

    -- Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    -- Get the selection's file type.
    value ← Selection.Convert[target: fileType];

    -- Check the file type to see if the selection is a simple text document.
    IF (value.value = NIL) OR (value.value↑ # NSAssignedTypes.tText) THEN
    BEGIN
      MessageWindow.PostSTRING [data.msgSW, "A simple text document must be selected before invoking
      Add Addresses.  Try again."L];
      RETURN;
    END;

    -- If we get here, we know the selection is a simple text document.  Convert the selection again to
    a file target.  value.value will contain a long pointer to NSFile.Reference for file.
    value ← Selection.Convert[target: file];
    ref ← value.value;

    -- Get handle to file.
    cacheHandle ← NSFile.OpenByReference[reference: ref↑];

    -- Ask the user to confirm adding addresses.
    okay ← Attention.PostAndConfirm[s: @question].confirmed;

    --If user okayed it, actually read addresses from simple text document.
     IF okay THEN ReadAddressesFromSimpleTextDocAndUpdateTable[
        data: data,
        cacheHandle: cacheHandle,
        replace: FALSE];

    -- Close cache file.
    NSFile.Close[cacheHandle];
```

```
  END;   -- of Procedure AddAddressesProc



  AddCommandsToAuxMenu: PROC [shell: StarWindowShell.Handle, data: MessagingToolBWSDefs.Data] =
  BEGIN
    addAddresses: XString.ReaderBody ← XString.FromSTRING ["Add Addresses"L];
    makeAddresses: XString.ReaderBody ← XString.FromSTRING ["Make Addresses"L];
    replaceAddresses: XString.ReaderBody ← XString.FromSTRING ["Replace Addresses"L];

    MenuEnumProc: StarWindowShell.MenuEnumProc =
    BEGIN
     --  Add the "Add Addresses" command to aux menu.
      MenuData.AddItem[
        menu, MenuData.CreateItem[
        zone: data.zone, name: @addAddresses, proc: AddAddressesProc,
        itemData: data]];

      --  Add the "Make Addresses" command to aux menu.
      MenuData.AddItem[
        menu, MenuData.CreateItem[
        zone: data.zone, name: @makeAddresses, proc: MakeAddressListProc,
        itemData: data]];

      --  Add the "Replace Addresses" command to aux menu.
      MenuData.AddItem[
        menu, MenuData.CreateItem[
        zone: data.zone, name: @replaceAddresses, proc: ReplaceAddressesProc,
        itemData: data]];

      -- Don't enumerate any other popup menus.
      stop ← TRUE;
    END;   -- of NESTED procedure MenuEnumProc

    -- MAIN CODE for AddCommandsToAuxMenu
    -- ASSUMPTION: First Popup Menu will be Aux menu!
    StarWindowShell.EnumeratePopupMenus[shell, MenuEnumProc];
  END;   -- of Procedure AddCommandsToAuxMenu



<<  Keep around if window header command(s) are added in future...
  AddWindowHeaderCommands: PROCEDURE [shell: StarWindowShell.Handle, data: MessagingToolBWSDefs.Data] =
  BEGIN
    loadAddresses: XString.ReaderBody ← XString.FromSTRING ["Load Addresses"L];
    loadItem: MenuData.ItemHandle;
    menuHandle: MenuData.MenuHandle;

    --  Get a handle to the regular property sheet commands.
    menuHandle ← StarWindowShell.GetRegularCommands[shell];

    --  Create Load Addresses command.
    loadItem ← MenuData.CreateItem [zone: data.zone, name: @loadAddresses, proc: LoadAddressListProc,
    itemData: data];

    --  Here is some hokey code to avoid multiple repaints in the addition of menu items. First reset
    the swapItemProc to be the simple one that will NOT cause a repaint when items get added.
    --oldSwapItemProc ← MenuData.SetSwapItemProc [menu: menuHandle, new:
    MenuData.UnpostedSwapItemProc];

    --  Add the Load Addresses command to window header.
    --MenuData.AddItem [menu: menuHandle, new: loadItem];

    --  Restore the original swapItemProc so a repaint will occur when the final item gets added.
    --[] ← MenuData.SetSwapItemProc [menu: menuHandle, new: oldSwapItemProc];

    --  Add final command, which in this case is the Load Addresses, to window header.
    MenuData.AddItem [menu: menuHandle, new: loadItem];

  END;  --  of Procedure AddWindowHeaderCommands
>>
```

```
Apply: PROCEDURE [data: MessagingToolBWSDefs.Data] RETURNS [ok: BOOLEAN] =
BEGIN
  nAddresses: CARDINAL;
  zone: UNCOUNTED ZONE ← data.zone;

  -- Clear the Message window.
  MessageWindow.Clear [data.msgSW];

  -- Check to see whether the table has been edited.
  IF ~XStringTableWindow.HasAnyBeenChanged[data.tableWindow] THEN RETURN [ok: TRUE];

  -- Verify that rows in the BWS table didn't exceed the maximum limit.  If it did, post an error
  message that specifies the current limit so users won't be left in the dark.
  nAddresses ← TableWindow.NumberofRowsAndColumns [window: data.tableWindow].rows - 1;
  IF nAddresses > MessagingToolBWSDefs.maxAddresses THEN
  BEGIN
    MessageWindow.PostSTRING [data.msgSW, "Maximum limit of 250 addresses has been exceeded."L];
    RETURN[ok: FALSE];
  END;

  -- Update addresses in the data's record.
  data.nAddresses ← nAddresses;
  UpdateAddressesFromTable [window: data.tableWindow, data: data];

  -- Save the cache addresses by writing them to a file.
  WriteAddressesToFile [data: data];

  -- All done and everything is cool.
  RETURN [ok: TRUE];
END;  -- of Procedure Apply



Init: PROCEDURE = {};



LayoutProc: FormWindow.LayoutProc =

<<  FormWindow.LayoutProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];  >>
BEGIN
  line: FormWindow.Line;
  tabStopInterval: CARDINAL = 50;
  -- set the tabs for FormWindow
  tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];
  FormWindow.SetTabStops[window: window, tabStops: tabChoice];
  -- Line 1
  line ← FormWindow.AppendLine[window: window, spaceAboveLine: 20];
  FormWindow.AppendItem[
    window: window, item: Items.tableWindow.ORD, line: line,
    tabStop: 17 / tabStopInterval, preMargin: 17 MOD tabStopInterval];
END;  -- of Procedure LayoutProc



MakeAddressListProc: MenuData.MenuProc =

<<  MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
LONG UNSPECIFIED]; >>

BEGIN
  chCR: XChar.Character ← XChar.Make [set: XCharSets.Sets.latin.ORD, code:
  XCharSet0.Codes0.newLine.ORD];
  chSP: XChar.Character ← XChar.Make [set: XCharSets.Sets.latin.ORD, code:
  XCharSet0.Codes0.space.ORD];
  data: MessagingToolBWSDefs.Data ← itemData;
```

```
    handle: LogFile.Handle;
    name: XString.ReaderBody ← XString.FromSTRING ["MessagingTool Addresses of "L];
    savedTableCellName: XString.ReaderBody;
    wbName: XString.WriterBody;

    WriteNameOrAddressProc: PROC [window: Window.Handle, cell: TableWindow.Cell, cellContent:
    XString.Reader] RETURNS [stop: BOOLEAN ← FALSE] =
    BEGIN
      -- Check to see if this cell is from row 0 which are the headers.  If so, skip.
      IF cell.row = 0 THEN RETURN;

      -- If this is the name column, hang on to the name.  Otherwise, output a line to the file
      containing the address, space, name, and a carriage return; and free up the bytes used for name.
      IF cell.column = 0 THEN savedTableCellName ← XString.CopyToNewReaderBody[r: cellContent, z:
      data.zone]
      ELSE
      BEGIN
        LogFile.PutXString[handle: handle, s: cellContent];
        LogFile.PutXChar[handle: handle, c: chSP, n:1];
        LogFile.PutXString[handle: handle, s: @savedTableCellName];
        LogFile.PutXChar[handle: handle, c: chCR, n:1];
        XString.FreeReaderBytes [r: @savedTableCellName, z: data.zone];
      END;
    END;  -- of nested Procedure WriteNameOrAddressProc


    -- MAIN CODE for MakeAddressListProc

    --  Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    -- Allocate document's name from a zone.
    wbName ← XString.CopyToNewWriterBody[r: @name, z: data.zone, extra: 50];

    --  Append the current time to the prefix.
    XTime.Append[@wbName];

    --  Get a readerbody which represents the document's complete name.
    name ← XString.ReaderFromWriter[@wbName]↑;

    -- Use Deb Lewis' hack for a painless way of creating a simple text document.
    handle ← LogFile.Create[name: @name];
    XStringTableWindow.EnumerateCells[
      window: data.tableWindow,
      callBack: WriteNameOrAddressProc];
    LogFile.Close[handle: handle];
  END;  --  of Procedure MakeAddressListProc




MakeItemsProc: FormWindow.MakeItemsProc =

<<  FormWindow.MakeItemsProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG POINTER]; >>

 BEGIN
    data: MessagingToolBWSDefs.Data ← clientData;
    size: Window.Dims ← [100, 100];   -- size here is arbitrary!
    zone: UNCOUNTED ZONE ← data.zone;

    --  Save the form window for later reference.
    data.optionSheetFW ← window;

    --  Make a window item.
    data.tableWindow ← FormWindow.MakeWindowItem [
      window: window,
      myKey: Items.tableWindow.ORD,
      boxed: FALSE,
      visibility: visible,
      size: size];

    --  Turn window into a BWS table and set the values.
    RestoreAddressTable [window: data.tableWindow, data: data];
  END;  --  of Procedure MakeItemsProc
```

```
  MenuItemProc: PropertySheet.MenuItemProc =

  << PropertySheet.MenuItemProc: TYPE = PROCEDURE [shell: StarWindowShell.Handle, formWindow:
  Window.Handle, menuItem: PropertySheet.MenuItemType, clientData: LONG POINTER] RETURNS [ok:
  BOOLEAN]; >>

  BEGIN
    data: MessagingToolBWSDefs.Data = clientData;

    -- Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    SELECT menuItem FROM
      done =>
        BEGIN
          ok ← Apply[data];
          IF ok THEN
          BEGIN
            data.optionSheetOpen ← FALSE;
          END; -- if ok
        END; -- done
      cancel =>
        BEGIN
          ok ← TRUE;
          data.optionSheetOpen ← FALSE;
        END; -- cancel
      apply =>
        BEGIN
          ok ← Apply[data];
          ok ← FALSE; -- this keeps the window from closing after a successful apply.
        END; -- apply
      ENDCASE;

    RETURN[ok];
  END; -- of Procedure MenuItemProc



  ReplaceAddressesProc: MenuData.MenuProc =

 << MenuData.MenuProc: TYPE = PROCEDURE [window: Window.Handle, menu: MenuData.MenuHandle, itemData:
 LONG UNSPECIFIED]; >>

  BEGIN
    cacheHandle: NSFile.Handle ← NSFile.nullHandle;
    data: MessagingToolBWSDefs.Data ← itemData;
    okay: BOOLEAN ← FALSE;
    question: XString.ReaderBody ← XString.FromSTRING ["Okay to replace existing address list?"L];
    ref: LONG POINTER TO NSFile.Reference;
    value: Selection.Value;

    -- Clear the Message window.
    MessageWindow.Clear [data.msgSW];

    -- Get the selection's file type.
    value ← Selection.Convert[target: fileType];

    -- Check the file type to see if the selection is a simple text document.
    IF (value.value = NIL) OR (value.value↑ # NSAssignedTypes.tText) THEN
    BEGIN
      MessageWindow.PostSTRING [data.msgSW, "A simple text document must be selected before invoking
      Replace Addresses. Try again."L];
      RETURN;
    END;

    -- If we get here, we know the selection is a simple text document. Convert the selection again to
    a file target. value.value will contain a long pointer to NSFile.Reference for file.
    value ← Selection.Convert[target: file];
    ref ← value.value;
```

```
    -- Get handle to file.
    cacheHandle ← NSFile.OpenByReference[reference: ref↑];

    -- Ask the user to confirm adding addresses.
    okay ← Attention.PostAndConfirm[s: @question].confirmed;

    --If user okayed it, actually read addresses from simple text document.
     IF okay THEN ReadAddressesFromSimpleTextDocAndUpdateTable[
        data: data,
        cacheHandle: cacheHandle,
        replace: TRUE];

    -- Close cache file.
    NSFile.Close[cacheHandle];
  END;   -- of Procedure ReplaceAddressesProc



  RestoreAddressTable: PROCEDURE [window: Window.Handle, data: MessagingToolBWSDefs.Data] =

  --  This procedure turns the window into a BWS table and uses the
  -- data, which includes information about the addresses,
  -- to fill the table in.

  BEGIN
    address: XString.ReaderBody ← XString.FromSTRING ["Network Address"L];
    addressColWidth: CARDINAL ← 200;
    index: CARDINAL;
    nameColWidth: CARDINAL ← 125;
    minDims: Window.Dims;
    name: XString.ReaderBody ← XString.FromSTRING ["Name"L];
    nbrColumns: CARDINAL ← 2;
    --  Set the options before creating the BWS table to avoid problems
    -- with copying/moving rows (If setChangedFlagsOnInsertDelete is
    -- not set to TRUE then when a row is copied or moved, elements
    -- in the BWS table becomes blank).
    options: XStringTableWindow.Options ← [
      nextKeyDirection: row,
      nextKeyAtEndAddsARowOrColumn: TRUE,
      userCanAddRows: TRUE,
      userCanAddColumns: FALSE,
      userCanAdjustColumns: FALSE,
      userCanSelectRow: TRUE,
      userCanSelectColumn: TRUE,
      setChangedFlagsOnInsertDelete: TRUE];

    --  Create BWS table with structure information.
    XStringTableWindow.Create[
      window: window,
      columns: nbrColumns,
      rows: data.nAddresses + 1,
      cellContent: CellContentProc,
      columnWidths: varying,
      options: options,
      cellNotify: CellNotifyProc,
      canDeleteProc: CanDeleteProc,
      canSelectProc: CanSelectProc,
      destroyProc: DestroyProc,
      nextOutOfProc: NextOutOfProc,
      minDimsChangeProc: MinDimsChangeProc,
      clientData: data];

    --  Set the column widths.
    TableWindow.SetColumnWidth[window: window, column: 0, width: nameColWidth];
    TableWindow.SetColumnWidth[window: window, column: 1, width: addressColWidth];

    --  Get the minimum dimensions needed.  Take the maximum of the tableWindow's height or 450, so the
    psheet won't be too short in height.
    minDims ← TableWindow.GetMinDims[window];
    minDims.h ← MAX[minDims.h, 425];

    --  Set initial BWS table window size.
    FormWindowExtra.SetWindowItemSizeExtra[
      window: data.optionSheetFW,
      windowItemKey: Items.tableWindow.ORD,
```

```
      newSize: minDims,
      repaint: FALSE];

    --  Set the cells in row 0 to contain the column headers.
    XStringTableWindow.SetCell [
      window: window,
      cell: [0, 0],
      cellContent: @name,
      repaint: FALSE];
    XStringTableWindow.SetCell [
      window: window,
      cell: [0, 1],
      cellContent: @address,
      repaint: FALSE];

    --  Fill in the table with address list data.
    FOR index IN [1..data.nAddresses] DO
      XStringTableWindow.SetCell [
        window: window,
        cell: [index, 0],
        cellContent: @data.addresses[index].name,
        repaint: FALSE];
      XStringTableWindow.SetCell [
        window: window,
        cell: [index, 1],
        cellContent: @data.addresses[index].netAddress,
        repaint: FALSE];
    ENDLOOP;

    -- Now that we're done changing the table, reset an internal TW boolean variable so we can find out
    later whether the user edited the BWS table.
    XStringTableWindowExtra1.AssignFALSEToAnyChanged[window];

    -- Set selection and input focus in the first name in first row.
    XStringTableWindow.SetCellSelection [window: window, cell: [1, 0]];
    XStringTableWindow.SetInputFocus [window: window, cell: [1, 0]];

  END;  -- of Procedure RestoreAddressTable



  UpdateAddressesFromTable: PROCEDURE [window: Window.Handle, data: MessagingToolBWSDefs.Data] =

  --  This procedure enumerately stores the address info from table into
  -- the data's addresses.

  BEGIN

    StoreNameOrAddressProc: PROC [window: Window.Handle, cell: TableWindow.Cell, cellContent:
    XString.Reader] RETURNS [stop: BOOLEAN ← FALSE] =

    BEGIN
      --  Check to see if this cell is from row 0 which are the headers.  If so, skip.
      IF cell.row = 0 THEN RETURN;

      --  Update a particular cell in AddressData.
      IF cell.column = 0 THEN
        data.addresses[cell.row].name ←
          XString.CopyToNewReaderBody [r: cellContent, z: data.zone]
      ELSE
        data.addresses[cell.row].netAddress ←
          XString.CopyToNewReaderBody [r: cellContent, z: data.zone];
    END;  -- of nested Procedure StoreNameOrAddressProc


    --  MAIN CODE for UpdateAddressesFromTable

    --  Free all the reader bodies used by addresses
    FOR index: CARDINAL IN [1..data.nAddresses] DO
      XString.FreeReaderBytes [r: @data.addresses[index].name, z: data.zone];
      XString.FreeReaderBytes [r: @data.addresses[index].netAddress, z: data.zone];
    ENDLOOP;

    XStringTableWindow.EnumerateCells[
```

```
        window: window,
        callBack: StoreNameOrAddressProc];

    END;  -- of Procedure UpdateCacheAddressesFromTable



    WriteAddressesToFile: PROC [data: MessagingToolBWSDefs.Data] =

      --  This procedure saves cache addresses by writing them to a file.

    BEGIN
      fileName: NSString.String ← NSString.StringFromMesaString["MessagingToolAddresses.cache"L];
      fileHandle, nullDirectoryHandle, oldFileHandle: NSFile.Handle ← NSFile.nullHandle;
      filterList: ARRAY [0..1) OF NSFile.Filter ← [[equal[[name[fileName]]]]];
      filter: NSFile.Filter ← [and[DESCRIPTOR[filterList]]];
      fileSize: LONG CARDINAL ← 512;  -- 1 page
      fileType: NSFile.Type ← NSAssignedTypes.tText;
      index: CARDINAL;
      lengthInBytes: LONG CARDINAL;
      nameAttr: ARRAY [0..1) OF NSFile.Attribute ← [[name[fileName]]];
      oldFileExist: BOOLEAN ← TRUE;
      scope: NSFile.Scope ← [filter: filter];
      streamHandle: NSFileStream.Handle ← NSFileStream.Handle[NIL];
      systemFolder: NSFile.Handle ← NSFile.nullHandle;
      tempFileAttr: ARRAY [0..3) OF NSFile.Attribute;
      tempFileName: NSString.String ← NSString.StringFromMesaString["TempMessagingToolAddresses.cache"L];
      xfo: XFormat.Object;

      --  Set up attributes for a file which will soon be created.
      tempFileAttr ← [
        [name[tempFileName]],
        [type[fileType]],
        [sizeInBytes[fileSize]]];

      --  Create a temporary file and store the attributes.
      fileHandle ← NSFile.Create[directory: nullDirectoryHandle, attributes: DESCRIPTOR[tempFileAttr]];

      --  Get stream handle to temp file.
      streamHandle ← NSFileStream.Create[file: fileHandle, closeOnDelete: FALSE];

      --  Set up XFormat using a StreamObject.
      xfo ← XFormat.StreamObject[sH: streamHandle];

      --  Actually write the cache addresses out to temp file.
      FOR index IN [1..data.nAddresses] DO
        XFormat.ReaderBody[h: @xfo, rb: data.addresses[index].netAddress];
        XFormat.Char[h: @xfo, char: ' .ORD];
        XFormat.ReaderBody[h: @xfo, rb: data.addresses[index].name];
        XFormat.CR[h: @xfo, n: 1];
      ENDLOOP;

      --  Get & set length of the stream.
      lengthInBytes ← Stream.GetPosition[streamHandle];
      NSFileStream.SetLength[streamHandle, lengthInBytes];

      --  Delete stream handle.
      Stream.Delete[sH: streamHandle];

      --  Open up System Folder.
      systemFolder ← Catalog.Open[BWSFileTypes.systemFileCatalog];

      --  Make temp file permanent by moving it into directory.  (This file
      --  will now be referred as new file.)
      NSFile.Move[file: fileHandle, destination: systemFolder];

      --  Find old MessagingToolAddresses.cache file.
      oldFileHandle ← NSFile.Find[directory: systemFolder, scope: scope !
        NSFile.Error => {oldFileExist ← FALSE; CONTINUE; }];

      --  Delete the old file if it exists.
      IF oldFileExist THEN NSFile.Delete[file: oldFileHandle];

      --  Rename new file.
```

```
    NSFile.ChangeAttributes[file: fileHandle, attributes: DESCRIPTOR[nameAttr]];

    -- Close new file and system folder.
    NSFile.Close[file: fileHandle];
    NSFile.Close[file: systemFolder];

  END;  -- of Procedure WriteAddressesToFile




-- ************************************************************
-- *                    BWS TABLE PROCEDURES
-- *                   in alphabetical order
-- ************************************************************



  CanDeleteProc: PUBLIC XStringTableWindow.CanDeleteProc =

    <<  XStringTableWindow.CanDeleteProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
    POINTER, first: CARDINAL, n: CARDINAL, rowOrColumn: XStringTableWindow.RowOrColumn] RETURNS
    [okToDelete: BOOLEAN];  >>

    --  This procedure is provided in case the user tries to do an abnormal deletion.  If all the rows
    or columns were selected, then rows 2 ... n will be deleted and row 1 (the row beneath the headers)
    will be cleared.  If the headers were part of a selection, then they will not be deleted but the
    rest of the selection will be deleted.  Otherwise, the whole selection will be deleted.

    --  Implementation note:  When a deletion occurs, this procedure does not do a repaint because it
    is assumed that the client had provided a MinDimsChangeProc.  When the table size changes due to
    deletion/insertion, the TableWindow lower-level procedures will call the client's
    MinDimsChangeProc which should repaint the table.

  BEGIN
    allRowsAreSelected, allColsAreSelected: BOOLEAN;
    index, nCols, nRows: CARDINAL;
    nullRB: XString.ReaderBody ← XString.nullReaderBody;

    --  Find out how many rows and columns there are in the table.
    [nRows, nCols] ← TableWindow.NumberofRowsAndColumns [window: window];

    --  Find out whether all the rows or columns were selected.
    allRowsAreSelected ← ((rowOrColumn = row) AND (((first = 0) AND (n = nRows)) OR ((first = 1) AND (n
    = nRows - 1))));
    allColsAreSelected ← ((rowOrColumn = column) AND (first = 0) AND (n = nCols));

    --  Are all rows or columns selected?
    IF allRowsAreSelected OR allColsAreSelected THEN
    BEGIN
      --  Clear row 1.
      FOR index IN [0..nCols) DO
        XStringTableWindow.SetCell[window: window, cell: [1, index], cellContent: @nullRB, repaint:
        FALSE];
      ENDLOOP;

      --  Delete all rows except headers and row 1.
      IF nRows > 2 THEN XStringTableWindow.DeleteRows[window: window, firstRow: 2, nRows: nRows - 2,
      repaint: FALSE];

      --  1st row was cleared so repaint the small table because table size didn't change.
      TableWindow.RedisplayArea [window: window, area: [[0, 0], [1, nCols - 1]]];
    END
    ELSE
      --  Are row(s) selected?
      IF rowOrColumn = row THEN
      BEGIN
        --  Were headers are part of selection?  If so, don't delete them but do delete rest of
        selection.
        IF first = 0 THEN
        BEGIN
```

```
          IF n > 1 THEN XStringTableWindow.DeleteRows[window: window, firstRow: 1, nRows: n - 1,
          repaint: FALSE];
        END
        ELSE
          -- Headers were not selected.
          XStringTableWindow.DeleteRows[window: window, firstRow: first, nRows: n, repaint: FALSE];
      END
      ELSE
        -- Column(s) were selected.
        XStringTableWindow.DeleteColumns[window: window, firstColumn: first, nColumns: n, repaint:
        FALSE];

    -- Need to clear the selection since TableWindows won't clear the selection if it receives FALSE
    from this procedure. This will workaround a bug if user hits DELETE immediately after deleting
    some rows without making a new selection..
    Selection.Clear[];

    -- Set okToDelete to FALSE because deletion was already taken care of, and return.
    RETURN[okToDelete: FALSE];

  END; -- of Procedure CanDeleteProc



  CanSelectProc: PUBLIC XStringTableWindow.CanSelectProc =

    <<  XStringTableWindow.CanSelectProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
    POINTER, first: CARDINAL, n: CARDINAL, rowOrColumn: XStringTableWindow.RowOrColumn] RETURNS
    [okToSelect: BOOLEAN]; >>

    -- This procedure does not allow the user to select a column so
    -- columns can't be added to or deleted from the table.

  BEGIN
    IF rowOrColumn = column THEN RETURN[okToSelect: FALSE]
    ELSE RETURN[okToSelect: TRUE];
  END; -- of Procedure CanSelectProc



  CellContentProc: PUBLIC XStringTableWindow.CellContentProc =

    <<  XStringTableWindow.CellContentProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
    POINTER, cell: XStringTableWindow.Cell, callBack: PROCEDURE [XString.Reader]];  >>

    -- This is called by XStringTableWindow at various times.
    -- Maintenance programmers: Unfortunately, this CellContentProc must
    -- be provided. It doesn't do anything because we don't want to provide a
    -- backing file for the Cache Address table. I tried
    -- assigning NIL for CellContentProc in XStringTableWindow.Create inside
    -- Procedure RestoreBWSTable, and crashed with ControlFault in Traps.

    -- 30-May-86. Added callBack inside the procedure as a workaround to a new
    -- bug in BWS 4.2a TableWindows application. What happens is that if the
    -- callBack is not provided in the CellContentProc then the rowHeight will
    -- be defaulted to 0, which results in a table that is extremely short in
    -- height.

  BEGIN
    nullRB: XString.ReaderBody ← XString.nullReaderBody;
    callBack[@nullRB];
  END; -- of Procedure CellContentProc



  CellNotifyProc: PUBLIC XStringTableWindow.CellNotifyProc =

    <<  XStringTableWindow.CellNotifyProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
    POINTER, cell: XStringTableWindow.Cell, results: TIP.Results] RETURNS [processedResults: BOOLEAN ←
    FALSE];  >>
```

```
  --  This procedure is called whenever a notification arrives for a cell.
  -- We're providing this procedure because we want to dump all TIP actions
  -- if the cell is from row 0 of the BWS table which are the headers, so
  -- the user will be unable to select, edit, delete, copy, or move that
  -- particular row.

BEGIN
  --  Is cell from row 0?  If so, dump TIP actions.
  IF cell.row = 0 THEN processedResults ← TRUE;
END;  -- of Procedure CellNotifyProc


DestroyProc: PUBLIC XStringTableWindow.DestroyProc =

  <<   XStringTableWindow.DestroyProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
  POINTER];   >>

  --   This procedure is called when XStringTableWindow.Destroy is called
  -- or when the window is destroyed.  Allows the client to destroy his
  -- clientData, which in our case is none.  Destroys the data associated
  -- with the BWS table but does not destroy the window itself.

BEGIN
END;  --  of Procedure DestroyProc



MinDimsChangeProc: XStringTableWindow.MinDimsChangeProc =

<< XStringTableWindow.MinDimsChangeProc = PROCEDURE [window: Window.Handle, clientData: LONG POINTER,
old: Window.Dims, new: Window.Dims]; >>

--  This client-provided procedure is passed in the XStringTableWindow.Create call in
RestoreAddressTable.  This proc will be called when the size of the Command List Table changes so it
can set the window item size to always make the whole table visible.

BEGIN
  data: MessagingToolBWSDefs.Data = clientData;

  --  Set the BWS table's size and repaint.
  FormWindowExtra.SetWindowItemSizeExtra[
    window: data.optionSheetFW,
    windowItemKey: Items.tableWindow.ORD,
    newSize: new,
    repaint: TRUE];
END;  -- of Procedure MinDimsChangeProc



NextOutOfProc: PUBLIC XStringTableWindow.NextOutOfProc =

  <<   XStringTableWindow.NextOutOfProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
  POINTER, cell: XStringTableWindow.Cell] RETURNS [goToNextCell: BOOLEAN];   >>

  --   This procedure is provided so when the user hits the NEXT key in
  -- the BWS table, the form window will automatically scroll if necessary.

BEGIN
  bottomOfCell, leftOfCell, rightOfCell, topOfCell: INTEGER;
  cellBox, fwBox, twBox: Window.Box;
  columns, rows: CARDINAL;
  nextCell: XStringTableWindow.Cell;
  newXPlaceForFW, newYPlaceForFW: INTEGER;
  sws: StarWindowShell.Handle ← StarWindowShell.ShellFromChild [window];
  fw: Window.Handle ← PropertySheet.GetFormWindows [[sws]].form;
  swsInteriorDims: Window.Dims ← BodyWindowParent.GetInteriorDims[
    BodyWindowParent.ParentFromBody[fw]];

  leftOfSWS, topOfSWS: INTEGER ← 0;  -- Assume sws' place is defaulted to [0,0].
  rightOfSWS: INTEGER ← swsInteriorDims.w;
```

```
    --  This procedure is called whenever a notification arrives for a cell.
    -- We're providing this procedure because we want to dump all TIP actions
    -- if the cell is from row 0 of the BWS table which are the headers, so
    -- the user will be unable to select, edit, delete, copy, or move that
    -- particular row.

  BEGIN
    --  Is cell from row 0?  If so, dump TIP actions.
    IF cell.row = 0 THEN processedResults ← TRUE;
  END;  -- of Procedure CellNotifyProc



  DestroyProc: PUBLIC XStringTableWindow.DestroyProc =

    <<   XStringTableWindow.DestroyProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
    POINTER];   >>

    --   This procedure is called when XStringTableWindow.Destroy is called
    -- or when the window is destroyed.  Allows the client to destroy his
    -- clientData, which in our case is none.  Destroys the data associated
    -- with the BWS table but does not destroy the window itself.

  BEGIN
  END;  --  of Procedure DestroyProc




  MinDimsChangeProc: XStringTableWindow.MinDimsChangeProc =

  << XStringTableWindow.MinDimsChangeProc = PROCEDURE [window: Window.Handle, clientData: LONG POINTER,
  old: Window.Dims, new: Window.Dims]; >>

  --  This client-provided procedure is passed in the XStringTableWindow.Create call in
  RestoreAddressTable.  This proc will be called when the size of the Command List Table changes so it
  can set the window item size to always make the whole table visible.

  BEGIN
    data: MessagingToolBWSDefs.Data = clientData;

    --  Set the BWS table's size and repaint.
    FormWindowExtra.SetWindowItemSizeExtra[
      window: data.optionSheetFW,
      windowItemKey: Items.tableWindow.ORD,
      newSize: new,
      repaint: TRUE];
  END;  -- of Procedure MinDimsChangeProc




  NextOutOfProc: PUBLIC XStringTableWindow.NextOutOfProc =

    <<   XStringTableWindow.NextOutOfProc: TYPE = PROCEDURE [window: Window.Handle, clientData: LONG
    POINTER, cell: XStringTableWindow.Cell] RETURNS [goToNextCell: BOOLEAN];   >>

    --   This procedure is provided so when the user hits the NEXT key in
    -- the BWS table, the form window will automatically scroll if necessary.

  BEGIN
    bottomOfCell, leftOfCell, rightOfCell, topOfCell: INTEGER;
    cellBox, fwBox, twBox: Window.Box;
    columns, rows: CARDINAL;
    nextCell: XStringTableWindow.Cell;
    newXPlaceForFW, newYPlaceForFW: INTEGER;
    sws: StarWindowShell.Handle ← StarWindowShell.ShellFromChild [window];
    fw: Window.Handle ← PropertySheet.GetFormWindows [[sws]].form;
    swsInteriorDims: Window.Dims ← BodyWindowParent.GetInteriorDims[
      BodyWindowParent.ParentFromBody[fw]];

    leftOfSWS, topOfSWS: INTEGER ← 0;  -- Assume sws' place is defaulted to [0,0].
    rightOfSWS: INTEGER ← swsInteriorDims.w;
```

```
    bottomOfSWS: INTEGER ← swsInteriorDims.h;

    << The following offsets refer to imaginary lines drawn in the psheet where
       the next table cell might line up against if it's out of viewing range.
       Currently the left & right offsets imagine that there is a vertical line
       in the middle of the psheet.  The top & bottom offsets imagine that there
       is a horizontal line in the middle of the psheet.  This is so that no
       matter where the invisible cell is (left, right, top, or bottom of the
       psheet), it will try to always line up in the same place when scrolled.
       When scrolling horizontally, it will line up with the vertical line.  When
       scrolling vertically, it will line up with the horizontal line.  It may not
       always line up in those places because this procedure imposes a restriction
       that form window's x & y places can't exceed 0.  These offsets can be
       changed such as the left offset can be 1/3 of the psheet while the right
       offset can be 2/3 of the psheet.  >>
    leftOffset, rightOffset: INTEGER ← INTEGER [swsInteriorDims.w / 2];
    topOffset, bottomOffset: INTEGER ← INTEGER [swsInteriorDims.h / 2];


    -- If no cell has the input focus, return.
    IF cell = TableWindow.nullCell THEN RETURN [goToNextCell: FALSE];

    -- Find out how many rows & columns there are in the table.
    [rows, columns] ← TableWindow.NumberofRowsAndColumns [window: window];

    -- Are we in the last cell?  If so, append a new row.
    IF (cell.column = columns - 1) AND (cell.row = rows - 1) THEN XStringTableWindow.AppendRows
    [window: window, nRows: 1, repaint: TRUE];

    -- Determine which cell is the "next" cell.
    IF (cell.column = columns - 1) THEN nextCell ← [column: 0, row: cell.row + 1]
    ELSE  nextCell ← [column: cell.column + 1, row: cell.row];

    -- Set the selection & input focus in the "next" cell.
    XStringTableWindow.SetCellSelection [window: window, cell: nextCell];
    XStringTableWindow.SetInputFocus [window: window, cell: nextCell];

    -- Get box information for "next" cell, table window, and form window.
    cellBox ← TableWindow.CellBox [window: window, cell: nextCell];
    twBox ← Window.GetBox [window];
    fwBox ← Window.GetBox [fw];

    -- Assign values to related Cell variables.
    leftOfCell ← fwBox.place.x + twBox.place.x + cellBox.place.x;
    topOfCell ← fwBox.place.y + twBox.place.y + cellBox.place.y;
    rightOfCell ← leftOfCell + cellBox.dims.w;
    bottomOfCell ← topOfCell + cellBox.dims.h;

    -- Do a quick check to see if "next" cell is completely visible?  If so, return.
    IF leftOfCell >= leftOfSWS THEN
      IF rightOfCell <= rightOfSWS THEN
        IF topOfCell >= topOfSWS THEN
          IF bottomOfCell <= bottomOfSWS THEN
            RETURN [goToNextCell: FALSE];


    << The "next" cell was not completely visible.  The remaining lines of code will
       automatically scroll the form window to make it become completely visible or
       align the left or top side if it's too large to fit completely within the psheet.  >>

    -- Initialize the new place variables for fw.
    newXPlaceForFW ← fwBox.place.x;
    newYPlaceForFW ← fwBox.place.y;

    IF leftOfCell < leftOfSWS THEN
      << See if we can align the form window's left side with the psheet without having
         the next table cell's left side go past the right offset.  If we can't, then
         move the form window to the right such that the next table cell's left side
         aligns with the left offset.  >>
      IF (twBox.place.x + cellBox.place.x) < rightOffset THEN newXPlaceForFW ← 0
      ELSE  newXPlaceForFW ← fwBox.place.x + ABS[leftOfCell] + leftOffset;

    IF rightOfCell > rightOfSWS THEN
      << See if we can align the form window's left side with the psheet.  >>
      IF (fwBox.place.x + rightOffset - leftOfCell) > leftOfSWS THEN newXPlaceForFW ← 0
      ELSE  newXPlaceForFW ← fwBox.place.x + rightOffset - leftOfCell;
```

```
    IF topOfCell < topOfSWS THEN
      IF (twBox.place.y + cellBox.place.y) < bottomOffset THEN newYPlaceForFW ← 0
      ELSE  newYPlaceForFW ← fwBox.place.y + ABS[topOfCell] + topOffset;

    IF bottomOfCell > bottomOfSWS THEN
    BEGIN
      IF (fwBox.place.y + bottomOffset - topOfCell) > topOfSWS THEN newYPlaceForFW ← 0
      ELSE  newYPlaceForFW ← fwBox.place.y + bottomOffset - topOfCell;
    END;

    -- Change the fwBox's place.
    fwBox.place.x ← newXPlaceForFW;
    fwBox.place.y ← newYPlaceForFW;

    -- Slide form window.
    Window.SlideAndSize [window: fw, newBox: fwBox];

    -- Repaint.
    Window.ValidateTree;

    RETURN [goToNextCell: FALSE];
  END;  -- of Procedure NextOutOfProc



  ReadAddressesFromSimpleTextDocAndUpdateTable: PROCEDURE [data: MessagingToolBWSDefs.Data,
  cacheHandle: NSFile.Handle, replace: BOOLEAN ← FALSE] =

  -- This procedure reads in cache addresses, which consists of a network address and a name, from a
  file.

  BEGIN
    emptyFile: BOOLEAN ← TRUE;
    index, startRow: CARDINAL;
    nbrOfBWSRows: CARDINAL;
    streamHandle: NSFileStream.Handle ← NSFileStream.Handle[NIL];
    tempAddressRB, tempNameRB: XString.ReaderBody;
    tokenHandle: XToken.Handle;

    BEGIN
      ENABLE
        UNWIND =>
          BEGIN
            IF streamHandle # NSFileStream.Handle[NIL] THEN Stream.Delete[sH: streamHandle];
          END;

      -- Get stream handle to cache file.
      streamHandle ← NSFileStream.Create[file: cacheHandle, closeOnDelete: FALSE !
        NSFile.Error =>
        BEGIN
          WITH myError: error SELECT FROM
            access =>
              SELECT myError.problem FROM
                fileNotLocal => MessageWindow.PostSTRING[
                  data.msgSW,
                  "Copy the remote simple text document onto your desktop, select it and try again."L];
              ENDCASE => MessageWindow.PostSTRING[data.msgSW, "ERROR: Cannot access file."L];
          ENDCASE => MessageWindow.PostSTRING[data.msgSW, "ERROR: Undefined file problem."L];

          GOTO ImGettingOutOfHere;
        END;
        ];

      -- Get token handle from the stream.
      tokenHandle ← XToken.StreamToHandle[streamHandle];

      -- Find out how many rows there are in the table.
      [nbrOfBWSRows, ] ← TableWindow.NumberofRowsAndColumns [window: data.tableWindow];

      -- If user wants to replace addresses, free all reader bodies.
      IF replace THEN
      BEGIN
        -- Delete all the rows except headers.
        XStringTableWindow.DeleteRows[
```

```
            window: data.tableWindow,
            firstRow: 1,
            nRows: nbrOfBWSRows - 1,
            repaint: FALSE];
          startRow ← 1;
        END
        ELSE startRow ← nbrOfBWSRows;

        -- Read cache file and fill in the address data.
        FOR index IN [startRow..MessagingToolBWSDefs.maxAddresses] DO

          -- Get a reader body (rb) representing a network address.
          tempAddressRB ← XToken.Filtered [
            h: tokenHandle,
            data: NIL,
            filter: XToken.NonWhiteSpace,
            skip: none,
            temporary: TRUE];

          -- Get a reader body (rb) representing a name.
          tempNameRB ← XToken.Filtered [
            h: tokenHandle,
            data: NIL,
            filter: XToken.Line,
            skip: none,
            temporary: TRUE];

          -- If both the name and address were blank, we probably hit the end
          -- of the file so free readerbodies and exit the loop.
          IF XString.Empty[@tempAddressRB] AND XString.Empty[@tempNameRB] THEN
          BEGIN
            [] ← XToken.FreeTokenString[@tempAddressRB];
            [] ← XToken.FreeTokenString[@tempNameRB];
            EXIT;
          END;

          -- If we ever get here, file is NOT empty!
          emptyFile ← FALSE;

          -- Append a row to table.
          XStringTableWindow.AppendRows[
            window: data.tableWindow,
            nRows: 1,
            repaint: FALSE];

          -- Set the appropriate table's address cell.
          XStringTableWindow.SetCell[
            window: data.tableWindow,
            cell: [index, 1],
            cellContent: @tempAddressRB,
            repaint: FALSE];

          -- Set the appropriate table's name cell.
          XStringTableWindow.SetCell[
            window: data.tableWindow,
            cell: [index, 0],
            cellContent: @tempNameRB,
            repaint: FALSE];

          -- Keep track of # of rows that the table has.
          nbrOfBWSRows ← index;

          -- Free token strings.
          [] ← XToken.FreeTokenString[@tempAddressRB];
          [] ← XToken.FreeTokenString[@tempNameRB];

        ENDLOOP;

        -- The selected file turned out to be empty!  If this was a
        -- replace operation, we'd better append an empty row.
        -- Otherwise, subtract 1 from nbrOfBWSRows.
        IF emptyFile THEN
        BEGIN
          IF replace THEN
          BEGIN
            XStringTableWindow.AppendRows[
```

```
            window: data.tableWindow,
            nRows: 1,
            repaint: FALSE];
          nbrOfBWSRows ← 1;
        END
        ELSE nbrOfBWSRows ← nbrOfBWSRows - 1;
      END;

      --  Repaint the ENTIRE table.
      TableWindow.RedisplayArea[
        window: data.tableWindow,
        area: [[0, 0], [nbrOfBWSRows, 1]]];

      --  Free token handle.
      [] ← XToken.FreeStreamHandle[tokenHandle];

      --  Delete stream handle.
      Stream.Delete[sH: streamHandle];
    END;  -- enable

    EXITS ImGettingOutOfHere => NULL;

  END;  -- of Procedure ReadAddressesFromSimpleTextDocAndUpdateTable




  -- MAINLINE CODE

  Init[];

END.


-- Log (when, who, what) --
24-Jun-88 15:20:43 - Terry - Created by stealing some of Delphi's code & modifying it for this tool's
needs.
 1-Jul-88 15:29:48 - Terry - ReadAddressesFromFile: Use XToken.Line to get ALL chars after address and
 before CR.
 6-Jul-88 18:17:24 - Terry - Apply: If the maximum limit for cache addresses has been exceeded, post an
 error message which specifies the current max limit.
10-Apr-89 20:33:10 - LTerry - MakeEditCacheAddressSheet: Changed title, shellDims's width to 460 (so
new wh cmd shows) and placeToDisplay's x to 670.  AddWindowHeaderCommands, LoadAddresses: Added.
13-Apr-89 20:48:58 - LTerry - AddWindowHeaderCommands: Added "Make Addresses" command.
MakeAddressListProc: Added.  LoadAddresses: Renamed to LoadAddressListProc.
15-May-89 21:01:48 - LTerry - LoadAddressListProc: Finally got it to only allow a simple text document
to be selected.
18-May-89 20:04:19 - LTerry - Hacked in different areas to update BWS table from selected s.t. doc.
This way, the user can decide whether to accept the updated addresses or not.  (If not, they invoke
"Cancel" and the data is unaffected with old values.)
23-May-89 16:21:29 - LTerry - MakeAddressListProc: Implemented.
24-May-89 20:02:47 - LTerry - LoadAddressListProc: Added NSFile.Close call.  Post info messages in
message sw always instead of sometimes to Attn window.  Temporarily hardcoded limit in an error msg
when max limit of addresses is exceeded since I'll be removing this limit in the future.  Clear
message sw when user invokes a command.
25-May-89 18:13:49 - LTerry - AddCommandsToAuxMenu: Added.
29-Jun-89 16:22:04 - LTerry - ReadAddressesFromSimpleTextDocAndUpdateTable: Catch NSFile error to avoid
crash when st doc is remote.  Apply: Changed text in error message when max limit of 250 (not 100)
addresses has been exceeded.
30-Jun-89 19:44:41 - LTerry - ReadAddressesFromFile: Assign 1 to data.nAddresses if couldn't
successfully read the first entry in cache file.  This will avoid a crash later on down the road.
Rewrote a little so that it doesn't kick out immediately when a blank address is found -- it will now
try to get a name also.  ReadAddressesFromSimpleTextDocAndUpdateTable: Changed also to be consistent
with not kicking out early with blank address.
 5-Jul-89 18:25:16 - LTerry - AddAddressesProc & ReplaceAddressesProc: Added.  LoadAddressesProc:
 Removed.  MakeEditCacheAddressSheet: No longer calls AddWindowHeaderCommands.
 AddWindowHeaderCommands: Commented out.  AddCommandsToAuxMenu: Added "Add Addresses" and "Replace
 Addresses" commands.
 6-Jul-89 14:53:29 - LTerry - ReadAddressesFromSimpleTextDocAndUpdateTable: Handle case of empty cache
 file whether adding or replacing.
```

```
-- File: LogStringWindowX.mesa - Created by Loreene Terry.  Last edit:
-- Loreene D. Terry:OSBU South:Xerox 28-Jun-89 17:46:14

-- Copyright (C) 1989 by Xerox Corporation. All rights reserved.


DIRECTORY
  Window;

LogStringWindowX: DEFINITIONS =
BEGIN

  Clear: PROCEDURE [Window.Handle];

END.

-- Log (when, who, what) --
28-Jun-89 17:09:28 - LTerry - Created.
```

```
-- File: LogStringWindowImpl.mesa - last edit:
-- Loreene D. Terry:OSBU South:Xerox 16-Jul-89 16:50:01
-- Breisacher          24-Dec-86 12:08:32

-- Copyright (C) 1985, 1986, 1989 by Xerox Corporation. All rights reserved.

DIRECTORY
  AdjustableWindow,
  BodyWindowParent,
  Context,
  Display,
  Heap,
  Inline,
  LogStringWindow,
  LogStringWindowX,
  SimpleTextDisplay,
  <<SimpleTextEdit,>>
  SimpleTextFont,
  SpecialSimpleText,
  SubwindowManager,
  TIP,
  Window,
  XFormat,
  XString;

LogStringWindowImpl: MONITOR
  IMPORTS BodyWindowParent, Context, Display, Heap, Inline, SimpleTextDisplay, <<SimpleTextEdit,
  >>SimpleTextFont, SpecialSimpleText, SubwindowManager, TIP, Window, XFormat, XString
  EXPORTS LogStringWindow, LogStringWindowX SHARES XString = BEGIN
  OPEN LogStringWindow;

  -- TYPEs

  LogData: TYPE = LONG POINTER TO LogDataObject;

  LogDataObject: TYPE = RECORD [
    zone: UNCOUNTED ZONE ← NIL,
    formatObject: XFormat.Object,
    formatH: XFormat.Handle ← NIL,
    <<fieldContext: SimpleTextEdit.FieldContext ← NIL,
    field: SimpleTextEdit.Field ← NIL,>>
    fieldOK: BOOLEAN ← FALSE,
    wb: XString.WriterBody ← XString.nullWriterBody,
    lineTable: LineTable ← NIL,
    visible: Window.Box ← Window.nullBox];

  LineTable: TYPE = LONG POINTER TO LineTableObject;
  LineTableObject: TYPE = RECORD [
    length: CARDINAL,
    lines: SEQUENCE maxLength: CARDINAL OF Line];
  Line: TYPE = RECORD [
    offset: CARDINAL,
    bitWidth: CARDINAL];

  -- Data

  logContext: Context.Type ← Context.UniqueType[];

  lineToLine: INTEGER ← SimpleTextDisplay.systemFontHeight<< + lineLeading>>;
  leftMargin: INTEGER = 0;
  topMargin: INTEGER = 0;

  -- Procedures

  Adjust: PUBLIC AdjustableWindow.AdjustProc = {
    logW: Window.Handle = BodyWindowParent.GetBody [window];
    logData: LogData = GetLogContext [logW];
    newBox: Window.Box ← Window.GetBox [logW];
    BodyWindowParent.Adjust [window, box, when];
    IF when = before THEN RETURN;

    newBox.dims.w ← BodyWindowParent.GetInteriorDims [window].w;
    Window.SlideAndSize [logW, newBox];
    logData.lineTable.length ← 1; -- causes entire line table to be rebuilt
    AppendToLineTable [logData, newBox.dims.w];
    SetCaretPlace [CalculateCaretPlace [logW]];
```

```
    Window.InvalidateBox [logW, Window.EntireBox [logW]];
    };

  Append: PUBLIC PROCEDURE [window: Window.Handle, r: XString.Reader, repaint: BOOLEAN] = {
    logW: Window.Handle = BodyWindowParent.GetBody [window];
    logData: LogData = GetLogContext [logW];
    logData.formatH.Reader [r];
    logData.fieldOK ← FALSE;
    IF repaint OR logData.wb.bytes [logData.wb.limit-1] = 15B THEN
      ForceOutInternal [logW, logData];
    };

  AppendToLineTable: PROCEDURE [logData: LogData, width: CARDINAL] = {
    rest: XString.ReaderBody;
    result: SimpleTextDisplay.Result ← stop;
    lines: CARDINAL ← logData.lineTable.length-1;
    rest ← XString.ReaderFromWriter [@logData.wb]↑;
    rest.offset ← logData.lineTable[lines].offset;
    UNTIL result = normal DO
      IF lines = logData.lineTable.maxLength THEN
        logData.lineTable ← NewLineTable [
          old: logData.lineTable, size: lines + 100, copy: TRUE,
          zone: logData.zone];
      logData.lineTable.length ← lines + 1;
      logData.lineTable[lines] ← [offset: rest.offset, bitWidth: TRASH];
      -- We're using SuperTextBlt because we want the TAB feature.
      [logData.lineTable[lines].bitWidth, result, rest] ← MyMeasureString [
        string: @rest, lineWidth: width];
      lines ← lines + 1;
      ENDLOOP;
    };

  Back: PUBLIC PROCEDURE [window: Window.Handle, amount: LogStringWindow.BackAmount] = {
    << No BackWord quite yet! >>
    logW: Window.Handle = BodyWindowParent.GetBody [window];
    logData: LogData = GetLogContext [logW];
    c: XString.Character = XString.ReverseLop [XString.ReaderFromWriter [@logData.wb],
      @logData.wb.context];
    cbox: Window.Box = BackUpLineTable [logData, c];
    Display.White [logW, cbox];
    };

  BackingWriter: PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [XString.Writer] = {
    logW: Window.Handle = BodyWindowParent.GetBody [window];
    logData: LogData = GetLogContext [logW];
    RETURN [@logData.wb];
    };

  BackUpLineTable: PROCEDURE [logData: LogData, c: XString.Character]
    RETURNS [characterBox: Window.Box] = {
    charWidth: CARDINAL ← SimpleTextDisplay.GetCharWidth [c];
    lastLine: CARDINAL ← logData.lineTable.length - 1;
    lastLineWidth: CARDINAL ← logData.lineTable [lastLine].bitWidth;
    IF c = '\n.ORD THEN charWidth ← SimpleTextDisplay.GetCharWidth [' .ORD]; -- wierd -
    SimpleTextDisplay treats a CR as same width as space (?)
    IF lastLineWidth < charWidth THEN {-- last char on line, subtract line
      logData.lineTable.length ← MAX [1, logData.lineTable.length-1];
      lastLine ← logData.lineTable.length - 1;
      lastLineWidth ← logData.lineTable [lastLine].bitWidth;
      };
    logData.lineTable [lastLine].bitWidth ← lastLineWidth - charWidth;
    characterBox ← [place: [
        x: leftMargin + lastLineWidth - charWidth,
        y: YFromLine [lastLine]],
      dims: [w: charWidth, h: lineToLine]];
    SetCaretPlace [characterBox.place];
    };

  Clear: PUBLIC PROCEDURE [window: Window.Handle] = {
    logW: Window.Handle = BodyWindowParent.GetBody [window];
    logData: LogData = GetLogContext [logW];
    -- Free writer.
    XString.FreeWriterBytes[w: @logData.wb];
    -- Free line table and create new one.
    logData.zone.FREE[@logData.lineTable];
```

```
    logData.lineTable ← NewLineTable [NIL, 100, FALSE, logData.zone];
    -- Call ForceOut so scrolling is set at beginning of log for new msgs.
    -- ForceOut[window: window];
    -- Invalidate & validate the log window to make change visually effective.
    Window.InvalidateBox [logW, Window.EntireBox [logW]];
    Window.Validate [logW];
    };

  Create: PUBLIC PROCEDURE [window: Window.Handle,
    zone: UNCOUNTED ZONE] = {
    BodyWindowParent.Create [parent: window, zone: zone];
    CreateInternal[window, zone];
    };

  CreateInternal: PROCEDURE [parent: Window.Handle,
    zone: UNCOUNTED ZONE] = {
    logData: LogData;
    logW: Window.Handle ← NIL;
    logW ← BodyWindowParent.CreateBody [parent: parent,
      box: [[0,0], [0,30000]],
      displayProc: LogDisplay,
      notifyProc: NIL<<LogNotify>>];
    Context.Create [type: logContext,
      data: logData ← zone.NEW [LogDataObject ← [
        zone: zone, wb: XString.NewWriterBody [2*LargeNodeThresh[zone], zone],
        lineTable: NewLineTable [NIL, 100, FALSE, zone]]],
      proc: DestroyLogContext, window: logW];
    logData.formatObject ← WriterObject [@logData.wb];
    logData.formatH ← @logData.formatObject;
    };

  Destroy: PUBLIC PROCEDURE [window: Window.Handle] = {
    [] ← Window.SetDisplayProc [BodyWindowParent.GetBody [window], NIL];
    <<[] ← TIP.SetNotifyProc [BodyWindowParent.GetBody [window], NIL];>>
    BodyWindowParent.Destroy [window];
    Context.Destroy [logContext, window];
    };

  DestroyLogContext: PROCEDURE [logData: LogData,
    window: Window.Handle] = {
    z: UNCOUNTED ZONE = logData.zone;
    <<IF logData.field # NIL THEN {
      SimpleTextEdit.DestroyField [logData.field];
      SimpleTextEdit.DestroyFieldContext [logData.fieldContext];
      };>>
    DoCaret [window, stop];
    z.FREE [@logData.lineTable];
    z.FREE [@logData];
    };

  ForceOut: PUBLIC PROCEDURE [window: Window.Handle] = {
    logW: Window.Handle = BodyWindowParent.GetBody [window];
    logData: LogData = GetLogContext [logW];
    ForceOutInternal [logW, logData];
    };

  ForceOutInternal: PROCEDURE [logW: Window.Handle, logData: LogData] = {
    lastLineBefore: CARDINAL = logData.lineTable.length - 1;
    lastLineBeforeWidth: CARDINAL = logData.lineTable [lastLineBefore].bitWidth;
    lastLineNow: CARDINAL;
    AppendToLineTable [logData, Window.GetBox[logW].dims.w];
    lastLineNow ← logData.lineTable.length - 1;
    MaybeScroll [logData: logData, logW: logW,
      lastLineBefore: lastLineBefore,
      lastLineNow: lastLineNow,
      lastLineBeforeWidth: lastLineBeforeWidth,
      lastLineBeforeWidthNow: logData.lineTable [lastLineBefore].bitWidth,
      lastLineNowWidth: logData.lineTable [lastLineNow].bitWidth];
    SetCaretPlace [[
      x: logData.lineTable [lastLineNow].bitWidth,
      y: YFromLine [lastLineNow]]];
    Window.Validate [logW];
    };

  GetLogContext: PROCEDURE  [window: Window.Handle]
    RETURNS [logData: LogData] = {
```

```
    logData ← Context.Find [logContext, window];
    IF logData = NIL THEN ERROR; -- not a LogWindow
    };

  IsIt: PUBLIC PROCEDURE [w: Window.Handle] RETURNS [yes: BOOLEAN] = {
    RETURN [Context.Find [logContext, w] # NIL]};

  LineFromY: PROCEDURE [y: INTEGER] RETURNS [CARDINAL] = INLINE
    {RETURN [ MAX [0, (y<<-topMargin>>) / lineToLine] ]};

  LogDisplay: Window.DisplayProc = {
    logData: LogData = GetLogContext [window];
    firstLine, lastLine: CARDINAL ← 0;
    minY: INTEGER ← INTEGER.LAST;
    maxY: INTEGER ← 0;
    linerb: XString.ReaderBody ← XString.ReaderFromWriter [@logData.wb]↑;
    width: CARDINAL = Window.GetBox[window].dims.w;
    line: CARDINAL ← 0;
    y: INTEGER ← 0;

    EachBox: PROCEDURE [w: Window.Handle, box: Window.Box] = {
      minY ← MIN [minY, box.place.y];
      maxY ← MAX [maxY, box.place.y+box.dims.h];
      };

    PaintLine: SimpleTextDisplay.BufferProc = {
      <<[result: TextBlt.Result, string: XString.Reader,
      address: Environment.BitAddress, dims: Window.Dims,
      bitsPerLine: CARDINAL] RETURNS [continue: BOOLEAN];>>
      textBox: Window.Box ← [dims: dims, place: [x: leftMargin, y: y]];
      whiteBox: Window.Box ← [
        dims: [w: width - dims.w, h: dims.h],
        place: [x: textBox.place.x + textBox.dims.w, y: textBox.place.y]];
      Display.Bitmap[
        window: window, box: textBox, address: address,
        bitmapBitWidth: bitsPerLine, flags: Display.replaceFlags];
      Display.White[window, whiteBox];
      y ← y + dims.h;
      RETURN [continue: FALSE];
      };

    Window.EnumerateInvalidBoxes [window, EachBox];
    IF minY > maxY THEN RETURN; -- nothing to paint
    firstLine ← LineFromY [minY];
    lastLine ← MIN [LineFromY [maxY], logData.lineTable.length-1];
    y ← YFromLine [firstLine];
    FOR line IN [firstLine..lastLine] DO
      linerb.offset ← logData.lineTable[line].offset;
      linerb.limit ← IF line = lastLine THEN logData.wb.limit ELSE logData.lineTable[line+1].offset;
      [] ← MyStringIntoBuffer [
        string: @linerb, lineWidth: width, bufferProc: PaintLine ];
      ENDLOOP;
    };

<<  LogNotify: TIP.NotifyProc = {
    logData: LogData = GetLogContext [window];
    IF logData.field = NIL THEN {
      logData.fieldContext ← SimpleTextEdit.CreateFieldContext [z: logData.zone,
          window: window, changeSizeProc: NullChangeSizeProc];
      logData.field ← SimpleTextEdit.CreateField [clientData: logData, context: logData.fieldContext,
      dims: Window.GetBox[window].dims, backingWriter: @logData.wb, readOnly: TRUE];
      SimpleTextEdit.SetPlace [logData.field, [0,0]];
      };
    << KLUDGE to get SimpleTextEdit to recompute its line table >>
    IF NOT logData.fieldOK THEN {
      SimpleTextEdit.SetFont [logData.field,
        SimpleTextEdit.GetFont [logData.field]];
      logData.fieldOK ← TRUE};
    [] ← SimpleTextEdit.TIPResults[logData.field, results];
    };
>>
  LogWindowFromRealLogWindow: PUBLIC PROCEDURE [w: Window.Handle]
    RETURNS [Window.Handle] = {
    RETURN [BodyWindowParent.ParentFromBody [w]];
    };
```

```
MakeLogStringSW: PUBLIC PROCEDURE [
  swmanager: Window.Handle,
  size: INTEGER ← SubwindowManager.minSize,
  vertScrollbar, horizScrollbar, adjustable: BOOLEAN ← TRUE,
  zone: UNCOUNTED ZONE]
  RETURNS [logString: Window.Handle] = {
  -- Should be in Subwindower.mesa.
  logString ← SubwindowManager.MakeSW[swmanager, vanilla, size,
    vertScrollbar, horizScrollbar, adjustable, zone];
  CreateInternal[logString, zone];
  };

MaybeScroll: PROCEDURE [logData: LogData, logW: Window.Handle,
  lastLineBefore, lastLineNow,
  lastLineBeforeWidth, lastLineBeforeWidthNow,
  lastLineNowWidth: CARDINAL] = {
  visible: Window.Box = IF logData.visible = Window.nullBox THEN Window.TrimBoxStickouts [logW,
  Window.EntireBox [logW]] ELSE logData.visible;
  oldPlace: Window.Place = Window.GetBox [logW].place;
  fudge: INTEGER = 0;
  lastLineBeforeY: INTEGER = YFromLine [lastLineBefore];
  lastLineNowY: INTEGER = YFromLine [lastLineNow];
  bottomLineBox: Window.Box ← visible;
  bottomLineBox.place.y ← visible.place.y + visible.dims.h - lineToLine - fudge;
  bottomLineBox.dims.h ← lineToLine + fudge;
  Window.InvalidateBox [logW, [
    [x: visible.place.x + lastLineBeforeWidth, y: lastLineBeforeY],
    [w: lastLineBeforeWidthNow - lastLineBeforeWidth, h: lineToLine]]];
  IF lastLineNowY = lastLineBeforeY THEN RETURN;
  IF Window.IsPlaceInBox [
    place: [x: visible.place.x, y: lastLineNowY],
    box: bottomLineBox] THEN
      Window.Slide [logW, [x: oldPlace.x, y: oldPlace.y - (lastLineNowY - lastLineBeforeY)]];
  Window.InvalidateBox [logW, [
    [x: visible.place.x, y: lastLineBeforeY + lineToLine],
    [w: visible.dims.w, h: lastLineNowY - lastLineBeforeY + lineToLine]]];
  };

systemFont: SimpleTextFont.MappedFontHandle = SimpleTextFont.MappedFont[];
bpl: CARDINAL = 1184;
  -- width of 19" (maximum) display; must be 0 MOD 16
  -- copied from STDImpl.
MyMeasureString: PROC [
  string: XString.Reader, lineWidth: CARDINAL ← CARDINAL.LAST]
  RETURNS [
    --width:-- CARDINAL, --result:-- SimpleTextDisplay.Result, --rest:--
    XString.ReaderBody] = {
  lineWidth ← MIN[lineWidth, bpl - 2];
  IF string.Empty THEN RETURN[0, normal, XString.nullReaderBody];
  RETURN SpecialSimpleText.SuperTextBlt [
    string, lineWidth, TRUE, fromFirstChar, NIL, NIL, LOOPHOLE [systemFont], format, tabStop]};

MyStringIntoBuffer: PROC [
  string: XString.Reader, bufferProc: SimpleTextDisplay.BufferProc,
  lineWidth: CARDINAL ← CARDINAL.LAST]
  RETURNS [
    --lastLineWidth:-- CARDINAL, --result:-- SimpleTextDisplay.Result, --rest:--
    XString.ReaderBody] = {
  lineWidth ← MIN[lineWidth, bpl - 2];
  IF string.Empty THEN RETURN[0, normal, XString.nullReaderBody];
  RETURN SpecialSimpleText.SuperTextBlt[
    string, lineWidth, TRUE, fromFirstChar, bufferProc, NIL, LOOPHOLE [systemFont],
    display, tabStop]};

NewLineTable: PROCEDURE [old: LineTable, size: CARDINAL, copy: BOOLEAN,
  zone: UNCOUNTED ZONE] RETURNS [new: LineTable] = {
  new ← zone.NEW[LineTableObject[size] ← [
    length: 1, lines: TRASH]];
  new[0] ← [offset: 0, bitWidth: 0];
  Inline.LongCOPY[from: @new[0], to: @new[1],
    nwords: (new.maxLength-1)*SIZE[Line]];
  IF old # NIL AND copy THEN
    Inline.LongCOPY[from: @old[0], to: @new[0],
      nwords: (old.maxLength)*SIZE[Line]];
  IF copy AND (old # NIL) THEN zone.FREE[@old];
  };
```

```
<<NullChangeSizeProc: SimpleTextEdit.ChangeSizeProc = {};>>

RealLogWindow: PUBLIC PROCEDURE [window: Window.Handle]
  RETURNS [Window.Handle] = {
  RETURN [BodyWindowParent.GetBody [window]];
  };

XFormatObject: PUBLIC PROCEDURE [window: Window.Handle]
  RETURNS [o: XFormat.Object] = {
  RETURN [[proc: XFormatProc, data: window]]};

XFormatProc: XFormat.FormatProc = {
<<[r: XString.Reader, h: XFormat.Handle];>>
  window: Window.Handle ← h.data;
  Append [window, r, FALSE];
  };

YFromLine: PROCEDURE [l: INTEGER] RETURNS [CARDINAL] = INLINE
  {RETURN [ l*lineToLine + topMargin ]};

-- mostly stolen from XFormat, but with the Extra stuff added

WriterObject: PROC [w: XString.Writer] RETURNS [XFormat.Object] = {
  RETURN[[WriterProc, XString.vanillaContext, w]]};

WriterProc: XFormat.FormatProc = {
  XString.AppendReader[to: h.data, from: r, extra: Extra [h.data, r]];
  h.context ← XString.WriterInfo[h.data].endContext};

Extra: PROCEDURE [w: XString.Writer, r: XString.Reader]
  RETURNS [bytes: CARDINAL] = {
  RETURN [2*LargeNodeThresh[w.zone] - XString.ByteLength [r] - 3];
  };

LargeNodeThresh: PROCEDURE [zone: UNCOUNTED ZONE]
  RETURNS [lnt: CARDINAL] = {
  attrs: Heap.Attributes ← Heap.GetAttributes [zone].attributes;
  WITH a: attrs SELECT FROM
    normal => RETURN [a.largeNodeThreshold];
    uniform => RETURN [0]; --??--
    ENDCASE;
  };

<< Caret stuff >>

CaretData: TYPE = RECORD [
  blinking, waitOne, painted: BOOLEAN ← FALSE,
  w: Window.Handle ← NIL, place: Window.Place ← [0,0]];

caretData: CaretData ← [];

Blink: PUBLIC ENTRY TIP.NotifyProc = {
  IF caretData.blinking THEN {
    IF ~caretData.waitOne THEN InvertCaret[];
    caretData.waitOne ← FALSE}};

DoCaret: PUBLIC ENTRY PROCEDURE [w: Window.Handle,
  action: CaretAction] = {
  SELECT action FROM
    start => IF ~caretData.blinking THEN {
      caretData.place ← CalculateCaretPlace [w];
      caretData.blinking ← TRUE};
    stop => IF caretData.blinking THEN {
      IF caretData.painted THEN InvertCaret[];
      caretData.blinking ← FALSE};
    paint => IF ~caretData.painted THEN InvertCaret[];
    erase => IF caretData.painted THEN InvertCaret[];
    ENDCASE;
  caretData.w ← w;
  };

InitCaret: PROCEDURE = {
  [] ← TIP.CreatePeriodicNotify [results: NIL, milliSeconds: 500, notifyProc: Blink];
  };
```

```
    InvertCaret: INTERNAL PROCEDURE = {
      Display.Invert [caretData.w, [caretData.place, [w:4, h: lineToLine]]];
      caretData.painted ← ~caretData.painted;
      };

    SetCaretPlace: ENTRY PROCEDURE [place: Window.Place] = {
      IF caretData.painted THEN InvertCaret[];
      caretData.place ← place;
      };

    CalculateCaretPlace: PROCEDURE [w: Window.Handle]
      RETURNS [place: Window.Place] = {
      logData: LogData = GetLogContext [w];
      lastLine: CARDINAL = logData.lineTable.length - 1;
      RETURN [[x: logData.lineTable [lastLine].bitWidth, y: YFromLine [lastLine]]];
      };

    -- Main line

    InitCaret[];

    END.

-- Log (when, who, what) --
28-Jun-89 17:29:31 - LTerry - Clear: Implemented.
29-Jun-89 10:03:44 - LTerry - Clear: Invalidate & validate log window.
16-Jul-89 16:34:08 - LTerry - Clear: Call ForceOut.  Commented it out since it didn't do what I wanted
and I was afraid of side effects.
```

```
-- File: MessagingToolBWS.config - last edit:
-- Loreene D. Terry:OSBU South:Xerox 28-Jun-89 18:09:47
-- Breisacher          20-Jul-87 11:00:06
-- Bowers:OSBU South:Xerox 17-Jan-87 19:21:49

-- Copyright (C) 1987, 1988, 1989 by Xerox Corporation. All rights reserved.


MessagingToolBWS: CONFIGURATION LINKS: CODE
  IMPORTS Atom, Attention, AuthSpecial, BodyWindowParent, Buffer, BWSZone, ByteBlt, Catalog, CH,
  CHCommonLookups, CommHeap, Context, Courier, Display, ExtendedString, Event, Format, FormWindow,
  FormWindowExtra, Heap, Inline, LogStringWindow, MenuData, MessageWindow, MessagingToolCommon,
  NetworkStream, NSAddressTranslation, NSFile, NSFileStream, NSName, NSString, OptionFile,
  PacketExchange, PacketExchangeInternal, PacketStream, Process, PropertySheet, RouterInternal,
  Runtime, Selection, SimpleTextDisplay, SimpleTextFont, Socket, SocketInternal, SpecialSimpleText,
  StarDesktop, StarWindowShell, StarWindowShellExtra5, Stream, String, Subwindower, SubwindowManager,
  System, TableWindow, Time, TIP, UserTerminal, Window, XFormat, XString, XStringTableWindow,
  XStringTableWindowExtra1, XTime, XToken
  CONTROL MessagingToolBWSImpl =
  BEGIN

  MessagingToolCommonConfig: CONFIGURATION LINKS: CODE
    IMPORTS Buffer, ByteBlt, CommHeap, Format, Heap, NetworkStream, PacketExchange,
    PacketExchangeInternal, PacketStream, Process, RouterInternal, Socket, SocketInternal, Stream,
    String, System, Time
    EXPORTS MessagingToolCommon =
    BEGIN
    MessagingToolCommonImpl;

    HandleManagerImpl;
    OneWordHandleManagerImpl;
    PCProtocolDataObjectImpl;
    PCProtocolSessionServerImpl;
    OurNetworkStreamImpl;
    PCProtocolSessionClientImpl;
    PCProtocolMessageServerAImpl;
    PCProtocolMessageServerBImpl;
    PCProtocolMessageClientImpl;
    END;  -- of MessagingToolCommonConfig

  MessagingToolCommonConfig;
  MessagingToolBWSImpl;

  EditCacheAddressPSheet;

  LogStringWindowImpl;
  LogFileImpl;   -- Deb Lewis' hack for simple text doc manipulation

  NSAddressTranslationImpl;
  ExtendedStringImpl;

  END.  -- of MessagingTool

LOG (date - person - action)
17-Jan-87 - Bowers - Creation
 6-Mar-87 - LFB - Adapted for BWS.
24-Feb-88 - Terry - Upgraded to have same functionality as XDE version.
24-Jun-88 - Terry - Added EditCacheAddressPSheet.
23-May-89 15:18:49 - LTerry - Added LogFileImpl.
```

-- File: MessagingToolBWS.doc - last edit:
-- Loreene D. Terry:OSBU South:Xerox 19-Jul-89 14:15:31
-- Bowers:OSBU South:Xerox 18-Jul-87 14:22:30

Please see "Installing and Using the Messaging Tool" or print "InstallingAndUsingTheMessagingTool.ip" from [Butler:OSBU South:Xerox]<MessagingTool>Doc>. This is a ViewPoint document with a Xerox Look that explains how to set MessagingToolBWS up and use it.

```
-- File: VPMaintain.config - last edit:
-- ANg:OSBU North:Xerox 21-Apr-88 17:16:04
-- JGS                31-Oct-85 15:48:03

-- Copyright (C) 1985, 1988 by Xerox Corporation. All rights reserved.

VPMaintain: CONFIGURATION
  IMPORTS
     Atom, Attention, Auth, BodyWindow, Catalog, CH, CHEntries, Containee,
    Context, Cursor, Courier,Directory, Display, Divider, FormWindow, FormWindowMessageParse, Heap,
    Inline, LogWindow,
    MCHNameExtras, MCHStringExtras, MenuData, MessageWindow, MoreCH, NSFile ,NSFileStream, NSName,
    NSString,
    Process, Selection, ServicesErrorMessage, SimpleTextDisplay, SimpleTextEdit, SimpleTextEditExtra,
    SimpleTextEditExtra5, SimpleTextFont, Space, StarWindowShell,
    StarWindowShellExtra2, Stream, String, TIP, TIPStar, Volume, Window, XCharSet0, XFormat,
    XMessage, XString

  CONTROL VPMShellImpl =
  BEGIN
    VPMShellImpl;
    VPMCommandsImpl;
    -- VPMMessageFileImpl;
    VPMMessagesImpl;
    LogSubwindow;
    NewMessageWindowImpl;
    MCHNameExtrasImpl;
  END...

-- LOG [Time - Person - Action]
--  2-Mar-88 16:15:17 - Curbow - Prettied up.
```

```
-- File: LogSubwindow.config - last edit:
-- ANg:OSBU North:Xerox 21-Apr-88 17:15:07
-- guzik.ES            16-Jun-87 15:41:45

-- Copyright (C) 1985, 1986, 1987, 1988 by Xerox Corporation. All rights reserved.

LogSubwindow: CONFIGURATION LINKS: FRAME
  IMPORTS Atom, BodyWindow, Context, Cursor, Display, Heap, Inline, NSFileStream, Process,
  SimpleTextDisplay, SimpleTextEdit, SimpleTextEditExtra,SimpleTextEditExtra5, SimpleTextFont, Stream,
  TIP, TIPStar, Window, XString
  EXPORTS BodyWindow, LogWindow, Scrollbar
  CONTROL BodyWindowImpl, ScrollbarImpl, LogWindowImpl  = BEGIN
    BodyWindowImpl;
    ScrollbarImpl;
    LogWindowImpl;
  END.
```

```
-- File: BodyWindow.mesa - last edit:
-- guzik.ES             2-Jan-86 17:50:17
-- Breisacher.ES        9-Sep-85  9:04:57

-- Copyright (C) 1985, 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  TIP USING [NotifyProc],
  Window USING [Box, DisplayProc, Handle, Place];

BodyWindow: DEFINITIONS = BEGIN

  Create: PROCEDURE [window: Window.Handle,
    verticalScrollbar: BOOLEAN ← TRUE,
    horizontalScrollbar: BOOLEAN ← TRUE,
    zone: UNCOUNTED ZONE,
    scrollLimitProc: ScrollLimitProc ← NIL<<, -- cannot be NIL
    scrollbarFeedbackProc: ScrollbarFeedbackProc ← NIL,
    moreScrollProc: MoreScrollProc ← NIL,
    garbageCollectBodiesProc: GarbageCollectBodiesProc ← NIL>>];

  CreateBody: PROCEDURE [window: Window.Handle, box: Window.Box ← [[0,0], [0,INTEGER.LAST]],
    displayProc: Window.DisplayProc, notify: TIP.NotifyProc]
    RETURNS [body: Window.Handle];
    -- If box.dims.w = 0 THEN box.dims.w ← size of parent.
    -- If box.dims.h = 0 THEN box.dims.h ← size of parent.

  Destroy: PROCEDURE [window: Window.Handle];

  GetBody: PROCEDURE [window: Window.Handle]
    RETURNS [body: Window.Handle];

  ScrollLimitProc: TYPE = PROCEDURE [window: Window.Handle]
    RETURNS [limit: Window.Place];

  <<ScrollbarFeedbackProc: TYPE = PROCEDURE [window: Window.Handle]
    RETURNS [offset, portion: Percent];
  Percent: TYPE = [0..100];
    -- To display where the user is positioned (offset) and how much is
    --  currently visible (portion).

  MoreScrollProc: TYPE = PROCEDURE [window: Window.Handle,
    vertical: BOOLEAN, flavor: MoreFlavor, amount: CARDINAL];
  MoreFlavor: TYPE = {before, after};
    -- called when we run out of body windows during scrolling

  GarbageCollectBodiesProc: TYPE = PROCEDURE [window: Window.Handle,
    body: Window.Handle];
    -- called when body is no longer visible>>

  END...
```

```
-- File: BodyWindowImpl.mesa - last edit:
-- guzik.ES             2-Jan-86 18:06:34
-- Breisacher.ES        9-Sep-85  9:07:06

-- Copyright (C) 1985, 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  BodyWindow,
  Context,
  Display,
  Scrollbar,
  SimpleTextDisplay,
  TIP,
  TIPStar,
  Window;

BodyWindowImpl: PROGRAM
  IMPORTS Context, Display, Scrollbar, SimpleTextDisplay, TIP, TIPStar, Window
  EXPORTS BodyWindow = BEGIN OPEN BodyWindow;

  << Window tree:

        client window
          interior
            client body windows
          vertical scrollbar
          horizontal scrollbar

     The client body windows are slid around inside the interior.

  >>

  -- TYPEs

  BodyData: TYPE = LONG POINTER TO BodyDataObject;

  BodyDataObject: TYPE = RECORD [
    numberOfBodies: CARDINAL ← 0,
    vertical: BOOLEAN ← FALSE,
    horizontal: BOOLEAN ← FALSE,
    zone: UNCOUNTED ZONE ← NIL,
    scrollLimitProc: ScrollLimitProc ← NIL];

  -- Data

  bodyContext: Context.Type = Context.UniqueType[];

  scrollAmount: INTEGER = SimpleTextDisplay.systemFontHeight;

  -- Procedures

  Create: PUBLIC PROCEDURE [window: Window.Handle,
    verticalScrollbar: BOOLEAN ← TRUE,
    horizontalScrollbar: BOOLEAN ← TRUE,
    zone: UNCOUNTED ZONE,
    scrollLimitProc: ScrollLimitProc ← NIL<<,
    scrollbarFeedbackProc: ScrollbarFeedbackProc ← NIL,
    moreScrollProc: MoreScrollProc ← NIL,
    garbageCollectBodiesProc: GarbageCollectBodiesProc ← NIL>>] = {
    interior: Window.Handle ← Window.Create [display: InteriorDisplay,
      box: [[0,0], Window.GetBox[window].dims], parent: window];
    [] ← Window.SetChild [window: window, newChild: interior];
    IF verticalScrollbar THEN
      Scrollbar.Create [windowToBeScrolled: interior, type: vertical,
        single: VerticalSingle, thumb: Thumb, zone: zone];
    IF horizontalScrollbar THEN
      Scrollbar.Create [windowToBeScrolled: interior, type: horizontal,
        single: HorizontalSingle, zone: zone];
    IF verticalScrollbar OR horizontalScrollbar THEN
      Scrollbar.Adjust [interior];
    Context.Create [type: bodyContext, data: zone.NEW [BodyDataObject ← [
      numberOfBodies: 0, vertical: verticalScrollbar,
      horizontal: horizontalScrollbar, zone: zone, scrollLimitProc: scrollLimitProc]],
      proc: DestroyBodyContext, window: window];
    };
```

```
CreateBody: PUBLIC PROCEDURE [window: Window.Handle, box: Window.Box,
  displayProc: Window.DisplayProc, notify: TIP.NotifyProc]
  RETURNS [body: Window.Handle] = {
  interior: Window.Handle = Window.GetChild[window];
  interiorDims: Window.Dims = Window.GetBox[window].dims;
  bodyData: BodyData = GetBodyContext [window];

  << *** TEMPORARY *** >>
  IF bodyData.numberOfBodies >= 1 THEN ERROR;
  << only one body window for now >>

  bodyData.numberOfBodies ← bodyData.numberOfBodies + 1;
  IF box.dims.w = 0 THEN box.dims.w ← interiorDims.w;
  IF box.dims.h = 0 THEN box.dims.h ← interiorDims.h;
  body ← Window.Create [display: displayProc, box: box,
    parent: interior];
  TIP.SetTableAndNotifyProc [body, TIPStar.NormalTable[], notify];
  IF Window.IsDescendantOfRoot [interior]
    THEN Window.InsertIntoTree [body]
    ELSE {
      [] ← Window.SetSibling [body, interior.GetChild[]];
      [] ← Window.SetChild [interior, body]};
  Window.ValidateTree [body];
  };

Destroy: PUBLIC PROCEDURE [window: Window.Handle] = {
  Context.Destroy [bodyContext, window];
  };

DestroyBodyContext: PROCEDURE [bodyData: BodyData, window: Window.Handle] = {
  z: UNCOUNTED ZONE = bodyData.zone;
  interior: Window.Handle = Window.GetChild[window];
  IF bodyData.vertical THEN Scrollbar.Destroy [interior, vertical];
  IF bodyData.horizontal THEN Scrollbar.Destroy [interior, horizontal];
  z.FREE [@bodyData];
  };

GetBody: PUBLIC PROCEDURE [window: Window.Handle]
  RETURNS [body: Window.Handle] = {
  RETURN [Window.GetChild [Window.GetChild [window]]];
  };

GetBodyContext: PROCEDURE [window: Window.Handle]
  RETURNS [bodyData: BodyData] = {
  bodyData ← Context.Find [bodyContext, window];
  IF bodyData = NIL THEN ERROR; -- not a BodyWindow
  };

HorizontalSingle: Scrollbar.SingleScrollProc = {
  body: Window.Handle = Window.GetChild [windowToBeScrolled];
  bodyDims: Window.Dims = Window.GetBox [body].dims;
  interiorDims: Window.Dims = Window.GetBox [Window.GetParent [body]].dims;
  bodyPlace: Window.Place = Window.GetBox [body].place;
  newPlace: Window.Place ← [y: bodyPlace.y, x: bodyPlace.x + (SELECT flavor FROM
      pageFwd => -interiorDims.w,
      pageBwd => interiorDims.w,
      forward => -scrollAmount,
      backward => scrollAmount,
      ENDCASE => 0)];
  IF arrowScrollAction = stop THEN RETURN;
  IF newPlace.x < interiorDims.w - bodyDims.w THEN newPlace.x ← interiorDims.w - bodyDims.w;
  IF newPlace.x > 0 THEN newPlace.x ← 0;
  Window.Slide [window: body, newPlace: newPlace];
  Window.ValidateTree [Window.GetParent [body]];
  };

InteriorDisplay: Window.DisplayProc = {
  Display.Gray [window, Window.EntireBox [window]]};

Thumb: Scrollbar.ThumbScrollProc = {
  };

VerticalSingle: Scrollbar.SingleScrollProc = {
  bodyData: BodyData = GetBodyContext [Window.GetParent [windowToBeScrolled]];
  body: Window.Handle = Window.GetChild [windowToBeScrolled];
  bodyDims: Window.Dims = Window.GetBox [body].dims;
```

```
    interiorDims: Window.Dims = Window.GetBox [Window.GetParent [body]].dims;
    bodyPlace: Window.Place = Window.GetBox [body].place;
    scrollLimit: Window.Place = bodyData.scrollLimitProc [Window.GetParent [windowToBeScrolled]];
    newPlace: Window.Place ← [x: bodyPlace.x, y: bodyPlace.y + (SELECT flavor FROM
        pageFwd => -interiorDims.h,
        pageBwd => interiorDims.h,
        forward => -scrollAmount,
        backward => scrollAmount,
        ENDCASE => 0)];
    IF arrowScrollAction = stop THEN RETURN;
    IF newPlace.y < -scrollLimit.y + INTEGER[SimpleTextDisplay.systemFontHeight] THEN newPlace.y ←
    -scrollLimit.y + INTEGER[SimpleTextDisplay.systemFontHeight];
    IF newPlace.y < interiorDims.h - bodyDims.h THEN newPlace.y ← interiorDims.h - bodyDims.h;
    IF newPlace.y > 0 THEN newPlace.y ← 0;
    Window.Slide [window: body, newPlace: newPlace];
    Window.ValidateTree [Window.GetParent [body]];
    };

END...
```

```
-- File: LogWindow.mesa - last edit:
-- ANg:OSBU North:Xerox 27-May-88 15:07:17
-- guzik.ES            27-Feb-87 14:55:18
-- Agbulos.ES          28-Apr-86 13:10:10

-- Copyright (C) 1985, 1986, 1987, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  NSFileStream USING [Handle],
  SimpleTextFont USING [MappedFontHandle],
  Window USING [Dims, Handle],
  XFormat USING [Object],
  XString USING [Reader];

LogWindow: DEFINITIONS = BEGIN


  FontInfo: TYPE = LONG POINTER TO FontInfoObject;
  FontInfoObject: TYPE = RECORD [
    fontHeight: CARDINAL ← 0,
    fontWidth: CARDINAL ← 0,
    fieldWidth: CARDINAL ← 0 ];



  LogWindowFullProc: TYPE = PROCEDURE [window: Window.Handle];


  ClearLog: PROCEDURE [window: Window.Handle];

  Create: PUBLIC PROCEDURE [
    window: Window.Handle,
    zone: UNCOUNTED ZONE,
    bodyWindowDims: Window.Dims ← [0, 0],
    horizontalScrollbars: BOOLEAN ← FALSE,
    verticalScrollbars: BOOLEAN ← TRUE,
    font: SimpleTextFont.MappedFontHandle ← NIL,
    lwFull: LogWindowFullProc];

  Destroy: PROCEDURE [Window.Handle];


  Append: PROCEDURE [window: Window.Handle, r: XString.Reader];

  SaveLog: PROCEDURE [sh: NSFileStream.Handle, window: Window.Handle];

  BuildLogWindowFromFile: PROCEDURE [sh: NSFileStream.Handle, window: Window.Handle, zone: UNCOUNTED
  ZONE]
    RETURNS [ok: BOOLEAN];


  XFormatObject: PROCEDURE [window: Window.Handle]
    RETURNS [o: XFormat.Object];



  END.
```

```
-- File: LogWindowImpl.mesa - last edit:
-- ANg:OSBU North:Xerox 15-Jun-88 16:31:57
-- guzik.ES            21-Dec-87 17:04:22
-- Agbulos.ES          28-Apr-86 13:10:16

-- Copyright (C) 1985, 1986, 1987, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, MakeAtom],
  BWSFontFileFormat USING [MappedFontHandle],
  BodyWindow USING [Create, CreateBody, Destroy, GetBody, ScrollLimitProc],
  Context USING [Create, Destroy, Find, Type, UniqueType],
  Display USING [Handle, White],
  Environment,
  Heap USING [Attributes, Create, GetAttributes, Type],
  LogWindow USING [LogWindowFullProc],
  NSFileStream,
  SimpleTextDisplay USING [systemFontHeight],
  SimpleTextEdit,
  SimpleTextEditExtra5 USING [AppendReader],
  SimpleTextFont USING [MappedDefaultFont, MappedFontHandle],
  Stream,
  System USING [Pulses],
  TIP USING [ATOM, NotifyProc, ResultObject, Results, SetNotifyProc],
  TIPStar USING [SetMode],
  Window USING [Box, Dims, DisplayProc, EntireBox, EnumerateInvalidBoxes, GetBox, Handle,
  IntersectBoxes, IsPlaceInBox, nullBox, Object, Place, SetDisplayProc, Slide, TrimBoxStickouts,
  Validate],
  XChar,
  XFormat USING [FormatProc, Handle, Object],
  XString USING [AppendReader, Block, ByteLength, Context, Empty, FreeReaderBytes, FromBlock,
  FromSTRING, InsufficientRoom, NewWriterBody, nullWriterBody, Reader, ReaderBody, ReaderFromWriter,
  ReaderInfo, vanillaContext, Writer, WriterBody, WriterInfo];

LogWindowImpl: PROGRAM
  IMPORTS Atom, BodyWindow, Context, Display, Heap, NSFileStream, SimpleTextDisplay, SimpleTextEdit,
  SimpleTextEditExtra5, SimpleTextFont, Stream, TIP, TIPStar, Window, XChar, XString
  EXPORTS LogWindow SHARES XString = BEGIN
  OPEN LogWindow;

  -- TYPEs

  LogData: TYPE = LONG POINTER TO LogDataObject;

  LogDataObject: TYPE = RECORD [
    zone: UNCOUNTED ZONE ← NIL,
    formatObject: XFormat.Object,
    formatH: XFormat.Handle ← NIL,
    lwFullProc: LogWindowFullProc ← NIL,
    fieldContext: SimpleTextEdit.FieldContext ← NIL, -- field context for both currentLine and field.
    STEfield: SimpleTextEdit.Field ← NIL,
    font: SimpleTextFont.MappedFontHandle ← NIL,
    wb: XString.WriterBody ← XString.nullWriterBody,
    windowFull: BOOLEAN ← FALSE,
    visible: Window.Box ← Window.nullBox];

  GlobalData: TYPE = LONG POINTER TO GlobalDataRecord;
  GlobalDataRecord: TYPE = MACHINE DEPENDENT RECORD [
    newLine: Atom.ATOM,
    newParagraph: Atom.ATOM,
    pointDown: Atom.ATOM,
    pointMotion: Atom.ATOM,
    pointUp: Atom.ATOM,
    adjustDown: Atom.ATOM,
    adjustMotion: Atom.ATOM,
    adjustUp: Atom.ATOM,
    copyModeUp: Atom.ATOM,
    moveModeUp: Atom.ATOM,
    nextDown: Atom.ATOM,
    leftArrowDown: Atom.ATOM,
    clearDown: Atom.ATOM,
    copyDown: Atom.ATOM,
    moveDown: Atom.ATOM
    ];

  -- Data
```

```
    logContext: Context.Type ← Context.UniqueType[]; --$$ identifies the context data that hung off the
    window.


    fileStamp: CARDINAL = 50586; --$$ don't know? From what I can tell it identifies the data stream.

    --$$ Log window Statistics.
    logWindowHeight: INTEGER = 32000;
    lineToLine: CARDINAL ← SimpleTextDisplay.systemFontHeight;
      -- ↑ this is reset in Create if a client font is supplied.
    leftMargin: INTEGER = 10;
    topMargin: INTEGER = 0;

    endOfLine: XChar.Character = XChar.Make [0, 1]; --$$ make null character;

    -- Temp Globals

    logWindowZone: UNCOUNTED ZONE ← Heap.Create [initial: 1, increment: 1];
    globals: GlobalData ← logWindowZone.NEW [GlobalDataRecord ← [
      newLine: Atom.MakeAtom ["NewLine"L], --$$ Don't think we care for this.
      newParagraph: Atom.MakeAtom ["NewParagraph"L], --$$  Don't think we care for this.
      pointDown: Atom.MakeAtom ["PointDown"L],
      pointMotion: Atom.MakeAtom ["PointMotion"L],
      pointUp: Atom.MakeAtom ["PointUp"L],
      adjustDown: Atom.MakeAtom ["AdjustDown"L],
      adjustMotion: Atom.MakeAtom ["AdjustMotion"L],
      adjustUp: Atom.MakeAtom ["AdjustUp"L],
      copyModeUp: Atom.MakeAtom ["CopyModeUp"L],
      moveModeUp: Atom.MakeAtom ["MoveModeUp"L],
      nextDown: Atom.MakeAtom ["NextDown"L],
      leftArrowDown: Atom.MakeAtom ["LeftArrowDown"L],
      clearDown: Atom.MakeAtom ["ClearDown"L],
      copyDown: Atom.MakeAtom ["CopyDown"L],
      moveDown: Atom.MakeAtom ["MoveDown"L]]];

    -- Errors

    LogWindowFull: PUBLIC ERROR = CODE;

    -- Procedures

  << Append takes a string and append it to currentline field and field. >>

    Append: PUBLIC  PROCEDURE [window: Window.Handle, r: XString.Reader] = {
      logW: Window.Handle = BodyWindow.GetBody [window];
      logData: LogData = GetLogContext [logW];  --$$ get backing store behind logwindow.
      cr: XString.ReaderBody ← XString.FromSTRING ["\n"L];
      IF XString.Empty [r] THEN RETURN;
      IF LONG [logData.wb.limit] + LONG [r.limit] >= LAST[CARDINAL]-5 THEN ERROR LogWindowFull;

     SimpleTextEditExtra5.AppendReader [f:logData.STEfield, string: r, repaint: TRUE];

       };

     ChangeSizeProc:  SimpleTextEdit.ChangeSizeProc = {
      fc: SimpleTextEdit.FieldContext ← SimpleTextEdit.GetFieldContext [f];
      logW: Window.Handle ← SimpleTextEdit.GetWindow [fc];
      logData: LogData ← GetLogContext [logW];
      STEfieldBox, whiteBox: Window.Box;

      -- if either of our simple text fields has grown beyond the
      -- limits of our log window then raise an error.
      box: Window.Box ← SimpleTextEdit.GetBox [f];
      IF LONG [box.place.y] + LONG [newHeight] > LONG [Window.GetBox [logW].dims.h] THEN
       ERROR LogWindowFull;

      IF newHeight < oldHeight THEN {
          -- newHeight > oldHeight means we normally appended text
          -- to the body field.
          -- In this case, the normal display mechanisms will repaint
          -- the window. If the newHeight < oldHeight, then we must have
          -- cleared out the body field (ie: we cleared the entire log window)
          -- so we have to repaint the left over text.
          STEfieldBox ← SimpleTextEdit.GetBox [logData.STEfield];
          whiteBox ← [[0, STEfieldBox.place.y + newHeight], [STEfieldBox.dims.w + leftMargin,
```

```
        oldHeight-newHeight]];
        Display.White [logW, whiteBox]};
    MaybeScroll [logData, logW, oldHeight];
    RETURN    };


  ClearLog: PUBLIC  PROCEDURE [window: Window.Handle] = {
    logW: Window.Handle = BodyWindow.GetBody [window];
    logData: LogData ← GetLogContext [logW];
    SimpleTextEdit.SetValue [logData.STEfield, NIL];
    SimpleTextEdit.SetPlace [logData.STEfield, [leftMargin,0]];
    Window.Slide [logW, [0, 0]];
    Window.Validate [logW];
    logData.windowFull ← FALSE};

  Create: PUBLIC  PROCEDURE [
      window: Window.Handle,
      zone: UNCOUNTED ZONE,
      bodyWindowDims: Window.Dims ← [0, 0],
      horizontalScrollbars: BOOLEAN ← FALSE,
      verticalScrollbars: BOOLEAN ← TRUE,
      font: SimpleTextFont.MappedFontHandle ← NIL,
      lwFull: LogWindowFullProc] = {

    logData: LogData;
    logW: Window.Handle ← NIL;
    scrollBarWidth: CARDINAL = 11;
    width: CARDINAL ← Window.GetBox[window].dims.w;
    IF font # NIL THEN lineToLine ← (LOOPHOLE [font,
    BWSFontFileFormat.MappedFontHandle]).maximumHeight;
    BodyWindow.Create [
      window: window,
      horizontalScrollbar: horizontalScrollbars,
      verticalScrollbar: verticalScrollbars,
      zone: zone,
      scrollLimitProc: ScrollLimit];
    logW ← BodyWindow.CreateBody [
      window: window,
      box: [[0,0], IF bodyWindowDims = [0, 0] THEN [width, logWindowHeight] ELSE bodyWindowDims],
      displayProc: LogDisplay,
      notify: LogNotify];
    Context.Create [type: logContext,
      data: logData ← zone.NEW [LogDataObject ← [
        zone: zone, wb: XString.NewWriterBody [2*LargeNodeThresh[zone], zone],
        font: font]],
      proc: DestroyLogContext, window: logW];

    IF bodyWindowDims # [0, 0] THEN width ← bodyWindowDims.w;
    logData.fieldContext ← SimpleTextEdit.CreateFieldContext [
        z: logData.zone,
        window: logW,
        changeSizeProc: ChangeSizeProc];
    logData.STEfield ← SimpleTextEdit.CreateField [
        clientData: logData,
        context: logData.fieldContext,
        dims: [width - leftMargin - scrollBarWidth, lineToLine],
        font: font,
        backingWriter: @logData.wb,
        readOnly: TRUE];
    SimpleTextEdit.SetPlace [logData.STEfield, [leftMargin,0]];
    logData.formatObject ← WriterObject [@logData.wb];
    logData.formatH ← @logData.formatObject;
    logData.lwFullProc ← lwFull;
    };

  Destroy: PUBLIC  PROCEDURE [window: Window.Handle] = {
    [] ← Window.SetDisplayProc [BodyWindow.GetBody [window], NIL];
    [] ← TIP.SetNotifyProc [BodyWindow.GetBody [window], NIL];
    BodyWindow.Destroy [window];
    Context.Destroy [logContext, window];
    };

  DestroyLogContext: PROCEDURE [logData: LogData,
    window: Window.Handle] = {
    z: UNCOUNTED ZONE = logData.zone;
```

```
    IF logData.STEfield # NIL THEN {
      SimpleTextEdit.DestroyField [logData.STEfield];
      SimpleTextEdit.DestroyFieldContext [logData.fieldContext];
      };
    z.FREE [@logData];
    };


  GetLogContext: PROCEDURE  [window: Window.Handle]
    RETURNS [logData: LogData] = {
    logData ← Context.Find [logContext, window];
    IF logData = NIL THEN ERROR; -- not a LogWindow
    };


  << LogDisplay gets call everytime part or whole of the log window need repainted. One instance this
  proc gets call is when SimpleTextEdit.TIPResults writes to the backing store and to the log window.
  The new text region is invalidated and Window interface calls this routine to validate it.
  >>
  LogDisplay: Window.DisplayProc = {
    logData: LogData = GetLogContext [window];
    minY: INTEGER ← INTEGER.LAST;
    maxY: INTEGER ← 0;
    linerb: XString.ReaderBody ← XString.ReaderFromWriter [@logData.wb]↑;
    width: CARDINAL = Window.GetBox[window].dims.w - leftMargin;
    visible: Window.Box = Window.TrimBoxStickouts [window, Window.EntireBox [window]];
    line: CARDINAL ← 0;
    y: INTEGER ← 0;

    -- $$$ minY will be set to the smallest(and most upper) y value of all the invalid boxes.
    -- $$$ maxY will be set to the largest(and most lower) y value of all the invalid boxes.
      EachBox: PROCEDURE [w: Window.Handle, box: Window.Box] = {
      minY ← MIN [minY, box.place.y];
      maxY ← MAX [maxY, box.place.y+box.dims.h];
      };

    << find the list of invalid regions of the window. Each box describes th region that is invalid.>>
    Window.EnumerateInvalidBoxes [window, EachBox];

    -- $$if minY=maxY then the window has just been open.

    -- $$$ if this case is true then we grow beyond the window.
    IF minY > maxY THEN RETURN; -- nothing to paint

    -- $$$ If logData.STEfield = NIL then repaint whatever is visible.
    IF logData.STEfield # NIL  THEN
      SimpleTextEdit.RepaintField [logData.STEfield];
    };



LogNotify: TIP.NotifyProc ={
    OPEN globals; --so we don't have to type globals everytime we reference and item from it.
      place: Window.Place; -- I think SimpleTextEdit needs to know where we set the mouse
      time: System.Pulses; -- I think SimpleTextEdit needs to know if select word, or line...
      logData: LogData = GetLogContext [window];
      msg: XString.ReaderBody ← XString.FromSTRING ["Log window full."L];
      BEGIN
      ENABLE XString.InsufficientRoom, LogWindowFull => {
      logData.lwFullProc [window];
      [] ← TIPStar.SetMode [normal];
      logData.windowFull ← TRUE;
    -- due to a bug in WindowOps, selecting a simple text field below the bottom limit of the window
    containing it will crash with an address fault. To protect against this, since copying into a
    simple text field (ie: the current line field) can grow the field below the bottom, we have to
    lock out all actions on the current line until the user calls ClearLogWindow or ClearCurrentLine.
      CONTINUE};
    FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
      WITH z: input SELECT FROM
        coords => place ← z.place;
        time => time ← z.time;
        atom => SELECT z.a FROM
          nextDown,
          leftArrowDown =>{RETURN};
          pointDown, pointMotion, pointUp,
          adjustDown, adjustMotion, adjustUp,
```

```
        copyDown, moveDown,copyModeUp, moveModeUp =>
           { PassNotification[place, logData, results];
             RETURN
           };

         ENDCASE;
        ENDCASE;
    ENDLOOP;
  IF logData.windowFull THEN RETURN;
  END -- enable block
  };

PassNotification:  PROCEDURE [place: Window.Place, logData: LogData, results: TIP.Results] =
  BEGIN
     [] ← SimpleTextEdit.TIPResults[logData.STEfield, results];
  END;


MaybeScroll: PROCEDURE [logData: LogData, logW: Window.Handle, oldBoxHeight: CARDINAL] = {
  PrevLineVisible, CurrLineVisible: BOOLEAN ← FALSE;
  visibleBox: Window.Box ← Window.TrimBoxStickouts [logW, Window.EntireBox [logW]];
  STEfieldBox: Window.Box ← SimpleTextEdit.GetBox [logData.STEfield];
  logWBox: Window.Box ← Window.GetBox [logW];
  oldPlace: Window.Place ← logWBox.place;
  FieldVisible: Window.Box ← Window.IntersectBoxes [STEfieldBox, visibleBox];
  IF FieldVisible # Window.nullBox THEN {

  PrevLineVisible ← Window.IsPlaceInBox [[STEfieldBox.place.x, STEfieldBox.place.y  + oldBoxHeight -
  (lineToLine/2)], visibleBox];

 CurrLineVisible ← Window.IsPlaceInBox [[STEfieldBox.place.x, STEfieldBox.place.y +
 STEfieldBox.dims.h], visibleBox];


   IF PrevLineVisible AND NOT CurrLineVisible THEN {
    << If the the old bottom of the current line box was visible and the new bottom is not, then the
    current line has grown beyond the bottom of the log window and needs to be scrolled. If the old
    bottom wasn't visible, then we leave it alone.>>

      currentLineVisibleBottomY: CARDINAL ← FieldVisible.place.y + FieldVisible.dims.h;
      currentLineBoxBottomY: CARDINAL ← STEfieldBox.place.y + STEfieldBox.dims.h;
      amountNeededToSlide: CARDINAL ← currentLineBoxBottomY - currentLineVisibleBottomY;
      newPlace: Window.Place ← [
        x: oldPlace.x,
        y: oldPlace.y - (
          IF (amountNeededToSlide MOD lineToLine) # 0 THEN ((amountNeededToSlide / lineToLine)+1) *
          lineToLine
          ELSE amountNeededToSlide)];

      IF LONG[STEfieldBox.place.y] + LONG[STEfieldBox.dims.h] > LONG[logWBox.dims.h] THEN
        ERROR LogWindowFull;
      Window.Slide [
        window: logW,
        newPlace: newPlace];
      Window.Validate [logW]}}
  };


 SaveLog: PUBLIC  PROCEDURE [sh: NSFileStream.Handle, window: Window.Handle] = {
 oldPosition: Stream.Position ← Stream.GetPosition [sh];
 rb: XString.ReaderBody;
 logW: Window.Handle = BodyWindow.GetBody [window];
 logData: LogData = GetLogContext [logW];
   font: SimpleTextFont.MappedFontHandle;
 BEGIN ENABLE UNWIND => {
     NSFileStream.SetLength [sh, oldPosition];
     CONTINUE};

   -- save the stamp
   Stream.PutWord [sh, fileStamp];

   -- save the log window body
   rb ← SimpleTextEdit.GetValue [logData.STEfield];
   SaveReader [sh, @rb];

    -- save the font info
```

```
    font ← IF logData.font # NIL THEN logData.font ELSE SimpleTextFont.MappedDefaultFont [];
    Stream.PutWord [sh, (LOOPHOLE [font, BWSFontFileFormat.MappedFontHandle]).maximumHeight];
    Stream.PutWord [sh, (LOOPHOLE [font, BWSFontFileFormat.MappedFontHandle]).maximumWidth];
    Stream.PutWord [sh, SimpleTextEdit.GetBox [logData.STEfield].dims.w];
    END -- enable block
    };


  BuildLogWindowFromFile: PUBLIC PROCEDURE [sh: NSFileStream.Handle, window: Window.Handle, zone:
  UNCOUNTED ZONE] RETURNS [ok: BOOLEAN ← FALSE] = {
  oldPosition: Stream.Position ← Stream.GetPosition [sh];
  rb: XString.ReaderBody;
  logW: Window.Handle = BodyWindow.GetBody [window];
  logData: LogData = GetLogContext [logW];
  length, maxLength: CARDINAL ← 0;
  visibleWindowBox: Window.Box ← Window.EntireBox [window];
  bodyWindowBox: Window.Box ← Window.EntireBox [logW];
  visibleBox: Window.Box ← Window.IntersectBoxes [Window.EntireBox [logW], Window.EntireBox [window]];
  STEfieldBox: Window.Box;
  halfVisibleWindowHeight: CARDINAL ← ((visibleBox.dims.h/lineToLine)/2)*lineToLine;
  stamp: CARDINAL ← 0;
  BEGIN ENABLE
  BEGIN
    Stream.EndOfStream => {
      NSFileStream.SetLength [sh, oldPosition];
      ok ← FALSE;
      CONTINUE};
    UNWIND => NSFileStream.SetLength [sh, oldPosition];
   END;

    -- get the stamp
    stamp ← Stream.GetWord [sh];
    IF stamp # fileStamp THEN {
      NSFileStream.SetLength [sh, oldPosition];
      RETURN};


    -- get the log window body
    rb ← GetReader [sh, zone];
    SimpleTextEdit.SetValue [logData.STEfield, @rb];
    XString.FreeReaderBytes [@rb, zone];

    -- get past the font info
    THROUGH [0..3) DO
      [] ← Stream.GetWord [sh]
      ENDLOOP;

    -- set the current line to the middle of the window.
    STEfieldBox ← SimpleTextEdit.GetBox [logData.STEfield];
    rb ← SimpleTextEdit.GetValue [logData.STEfield];
    IF NOT XString.Empty [@rb] AND rb.bytes [rb.limit-1] = 15B
    OR XString.Empty [@rb] THEN
      STEfieldBox.dims.h ← STEfieldBox.dims.h - lineToLine;
      -- ↑ taken from UpdateCurrentLinePosition (it's a kludge, but it works.)
    IF STEfieldBox.place.y+STEfieldBox.dims.h >= visibleBox.place.y+visibleBox.dims.h THEN {
      IF STEfieldBox.place.y+STEfieldBox.dims.h > bodyWindowBox.dims.h - visibleWindowBox.dims.h THEN
        -- if we're near the bottom of the body window, we want to
        -- set the current line to the bottom of the visible
        -- window.
        Window.Slide [window: logW, newPlace: [x: 0, y: -(bodyWindowBox.dims.h -
        visibleWindowBox.dims.h)]]
      ELSE
        -- we're nowhere near the bottom of the body window
        -- so we set the current line to the middle of the
        -- visible window.
        Window.Slide [window: logW, newPlace: [x: 0, y: -(STEfieldBox.dims.h -
        halfVisibleWindowHeight)]]};
    Window.Validate [logW];
    ok ← TRUE
    END -- enable block
    };

  GetReader: PROCEDURE [stream: NSFileStream.Handle, zone: UNCOUNTED ZONE] RETURNS [name:
  XString.ReaderBody] = {
    pad: CARDINAL = 3;
    length: CARDINAL ← LOOPHOLE [Stream.GetWord [stream]];
```

```
    wb: XString.WriterBody ← XString.NewWriterBody [length+pad, zone];
    block: Environment.Block ← XString.Block [XString.ReaderFromWriter [@wb]].block;
    bytes: CARDINAL;

    block.stopIndexPlusOne ← length;
    block.startIndex ← 0;
    [bytes,,] ← Stream.GetBlock [stream, block];
    name ← XString.FromBlock [block]
    };

  SaveReader: PROCEDURE [fileStream: NSFileStream.Handle, name: XString.Reader] = {
    extra: CARDINAL;
    c: XString.Context;
    startsWith377B: BOOLEAN;
    block: Environment.Block;
    bytes: CARDINAL;

    [context: c, startsWith377B: startsWith377B] ← XString.ReaderInfo[name];
    bytes ← XString.ByteLength [name];
    extra ← SELECT TRUE FROM
      startsWith377B,
      (c.prefix = 0) => 0,
      ENDCASE => c.suffixSize + 1;
    bytes ← bytes + extra;
    block ← XString.Block [name].block;

    Stream.PutWord [fileStream, bytes];  -- puts the length

    SELECT extra FROM -- this allows us to throw away the context
      0 => Stream.PutBlock [fileStream, block];
      1 => ERROR; -- should never happen
      2 => {
        Stream.PutByte [fileStream, 377B];
        Stream.PutByte [fileStream, c.prefix];
        Stream.PutBlock [fileStream, block] };
      3 => {
        Stream.PutByte [fileStream, 377B];
        Stream.PutByte [fileStream, 377B];
        Stream.PutByte [fileStream, c.prefix];
        Stream.PutBlock [fileStream, block] };
      ENDCASE => ERROR;
    };


  ScrollLimit: BodyWindow.ScrollLimitProc = {
    logW: Window.Handle ← BodyWindow.GetBody [window];
    logData: LogData ← GetLogContext [logW];
    STEfieldBox: Window.Box ← SimpleTextEdit.GetBox [logData.STEfield];
    limit ← [
      x: STEfieldBox.place.x + STEfieldBox.dims.w,
      y: STEfieldBox.place.y + STEfieldBox.dims.h]};


XFormatObject:PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [o:
XFormat.Object] = {
    RETURN [[proc: XFormatProc, data: window]]};

  XFormatProc: XFormat.FormatProc = {
  <<[r: XString.Reader, h: XFormat.Handle];>>
    window: Window.Handle ← h.data;
    Append [window, r];
    };

  -- mostly stolen from XFormat, but with the Extra stuff added

  WriterObject: PROC [w: XString.Writer] RETURNS [XFormat.Object] = {
    RETURN[[WriterProc, XString.vanillaContext, w]]};

  WriterProc: XFormat.FormatProc = {
    XString.AppendReader[to: h.data, from: r, extra: Extra [h.data, r]];
    h.context ← XString.WriterInfo[h.data].endContext};

  Extra: PROCEDURE [w: XString.Writer, r: XString.Reader]
    RETURNS [bytes: CARDINAL] = {
```

```
    RETURN [2*LargeNodeThresh[w.zone] - XString.ByteLength [r] - 3];
    };

  LargeNodeThresh: PROCEDURE [zone: UNCOUNTED ZONE]
    RETURNS [CARDINAL] = {
    attrs: Heap.Attributes ← Heap.GetAttributes [zone].attributes;
    WITH a: attrs SELECT FROM
      normal => RETURN [a.largeNodeThreshold];
      uniform => RETURN [0]; --??--
      ENDCASE => RETURN[0];
    };

  -- Main line

  END.


  LOG
  Guzik - 25-Oct-85  9:39:55 - provided for a newline callback proc and modified to allow for text
  insertion.
  12-Mar-87 17:23:21 - guzik - Catch LeftArrowDown along with NextDown to account for the mode being
  switched to cursor keys on an 8010.
4-Nov-87 16:20:50 - guzik - Call SetInputFocus and SimpleTextEdit.SetSelection instead of
SimpleTextEdit.SetInputFocus in Append, ClearLog, ClearCurrentLine to ensure that the
TakeInputFocusProc gets called.
21-Dec-87 16:19:07 - guzik - AR 16840 - In pointDown arm of LogNotify - Make sure that if the input
focus is taken by the currentLine field (either thru TIPResults or STE.SetInputFocus), we will call
the takeInputFocusProc.
```

```
-- File: NewMessageWindowImpl.mesa - last edit:
-- JGS                 18-Oct-85 15:14:55
-- SAJohnson.es        21-Jun-85 15:32:57
-- Breisacher.ES       19-Jun-85 10:02:31
-- Sandman.pa          16-Oct-84 14:30:19

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Attention USING [Clear, Post],
  Context USING [Create, Data, Destroy, Find, Type, UniqueType],
  Display USING [Bitmap, paintFlags, Shift, White],
  Heap USING [Create],
  MessageWindow USING [],
  SimpleTextDisplay,
  Window USING [
    Box, Create, Dims, EntireBox, EnumerateInvalidBoxes, GetBox, Handle,
    InsertIntoTree, InvalidateBox, Object, Place, SetDisplayProc, Validate],
  XCharSet0 USING [Codes0],
  XFormat USING [FormatProc, Object],
  XString USING [
    AppendReader, Character, ClearWriter, Context, CopyToNewReaderBody, Empty,
    FreeReaderBytes, FreeWriterBytes, MapCharProc, NewWriterBody, nullReaderBody,
    Reader, ReaderBody, ReaderFromWriter, ReverseMap, WriterBody];

NewMessageWindowImpl: PROGRAM
  IMPORTS
    Attention, Context, Display, Heap, SimpleTextDisplay, Window, XString
  EXPORTS MessageWindow
  SHARES XString = BEGIN OPEN XS: XString;

  -- TYPEs

  xMargin: CARDINAL = 2;
  yMargin: CARDINAL = 0;

  Data: TYPE = LONG POINTER TO DataObject;

  DataObject: TYPE = RECORD [
    zone: UNCOUNTED ZONE ← NIL,
    lines: LONG POINTER TO LineSeq ← NIL,
    invalid: LONG POINTER TO InvalidSeq ← NIL,
    place: Window.Place ← [x: xMargin, y: yMargin],
    postLine: CARDINAL ← 0,
    firstTime: BOOLEAN ← TRUE,
    wb: XS.WriterBody];

  LineIndex: TYPE = NATURAL[0..512);

  LineSeq: TYPE = RECORD [SEQUENCE length: LineIndex OF XS.ReaderBody];

  InvalidSeq: TYPE = RECORD [PACKED SEQUENCE length: LineIndex OF BOOLEAN];

  -- Data

  context: Context.Type ← Context.UniqueType[];

  z: UNCOUNTED ZONE ← Heap.Create[initial: 4];

  -- Procedures

 AllocateAndInsert: PUBLIC PROCEDURE [
   parent: Window.Handle,
   place: Window.Place ← [0, 0],
   dims: Window.Dims ← [9999, 0],
   zone: UNCOUNTED ZONE ← NIL,
   lines: CARDINAL ← 30 ]
   RETURNS [window: Window.Handle] = {
   IF dims.h = 0 THEN dims.h ← SimpleTextDisplay.systemFontHeight * lines;
   window ← Window.Create [display: NIL, box: [place, dims], parent: parent];
   Create [window, zone, lines];
   Window.InsertIntoTree [window];
   };

  Clear: PUBLIC PROCEDURE [window: Window.Handle] = {
    data: Data ← LocalFind [window];
```

```
    IF window = NIL OR data = NIL THEN { -- convenience for client
      Attention.Clear[]; RETURN};
    data.lines[data.postLine] ← XS.nullReaderBody;
    FOR i: CARDINAL IN [0..data.lines.length) DO
      XS.FreeReaderBytes[r: @data.lines[i], z: data.zone];
      data.lines[i] ← XS.nullReaderBody;
      ENDLOOP;
    XS.ClearWriter[@data.wb];
    data.firstTime ← TRUE; data.postLine ← 0; data.place ← [x: xMargin, y: yMargin];
    Display.White[window, window.EntireBox[]]};

  Create: PUBLIC PROCEDURE [
    window: Window.Handle, zone: UNCOUNTED ZONE ← NIL, lines: CARDINAL ← 10] = {
    data: Data ← NIL;
    place: Window.Place ← [0,0];
    fieldDims: Window.Dims ← [
      w: window.GetBox[].dims.w,
      h: SimpleTextDisplay.systemFontHeight ];
    IF zone = NIL THEN zone ← z;
    IF lines > LineIndex.LAST THEN
      lines ← window.GetBox[].dims.h/SimpleTextDisplay.systemFontHeight;
    IF lines = 0 THEN lines ← 1;
    [] ← Window.SetDisplayProc[window, Repaint];
    data ← zone.NEW [DataObject ← [
      zone: zone,
      lines: zone.NEW[LineSeq[lines]],
      invalid: zone.NEW[InvalidSeq[lines]],
      postLine: 0,
      wb: XS.NewWriterBody[z: zone, maxLength: 80]]];
    FOR i: CARDINAL IN [0..lines) DO
      data.lines[i] ← XS.nullReaderBody;
      ENDLOOP;
    Context.Create [context, data, DestroyContext, window];
    };

  Destroy: PUBLIC PROCEDURE [window: Window.Handle] = {
    Context.Destroy [context, window];
    [] ← Window.SetDisplayProc [window, NIL];
    };

  DestroyContext: PROCEDURE [data: Data, window: Window.Handle] = {
    local: UNCOUNTED ZONE ← data.zone;
    data.lines[data.postLine] ← XS.nullReaderBody;
    FOR i: CARDINAL IN [0..data.lines.length) DO
      XS.FreeReaderBytes[r: @data.lines[i], z: data.zone];
      ENDLOOP;
    XS.FreeWriterBytes[w: @data.wb];
    local.FREE[@data.lines];
    local.FREE[@data.invalid];
    local.FREE[@data];
    };

  IsIt: PUBLIC PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN] =
    {RETURN [ Context.Find[context, window] # NIL ]};

  LocalFind: PROC [w: Window.Handle] RETURNS [data: Data] = {
    data ← Context.Find[context, w];
    RETURN [data];
    };

  validate: BOOLEAN ← TRUE;

  Post: PUBLIC PROCEDURE [
    window: Window.Handle, r: XString.Reader, clear: BOOLEAN ← TRUE] = {
    data: Data ← LocalFind[window];
    rb: XString.ReaderBody;
    IF r = NIL THEN RETURN;
    IF window = NIL OR data = NIL THEN { -- convenience for client
      Attention.Post[r, clear]; RETURN};
    IF clear THEN NewPlaceToPaint[window, data];
    rb ← r↑;
    DO
      PaintLine[window, data, @rb];
      data.firstTime ← FALSE;
      IF XString.Empty[@rb] THEN EXIT;
      NewPlaceToPaint[window, data];
```

```
      ENDLOOP;
    IF validate THEN window.Validate[]};

  NewPlaceToPaint: PROCEDURE [w: Window.Handle, data: Data] = {
    currentLine: LineIndex ← data.postLine;
    dims: Window.Dims = w.GetBox[].dims;
    fontHeight: CARDINAL = SimpleTextDisplay.systemFontHeight;
    SELECT TRUE FROM
      data.firstTime => {
        data.lines[data.postLine ← 0] ← XS.nullReaderBody;
        data.place ← [x: xMargin, y: yMargin];
        RETURN};
      data.lines.length = 1 => {
        XS.ClearWriter[@data.wb];
        data.lines[data.postLine] ← XS.nullReaderBody;
        data.place ← [x: xMargin, y: yMargin + data.postLine*fontHeight]};
      currentLine < data.lines.length - 1 => {
        data.postLine ← currentLine + 1;
        data.lines[currentLine] ←
          XS.CopyToNewReaderBody[XS.ReaderFromWriter[@data.wb], data.zone];
        XS.ClearWriter[@data.wb];
        data.lines[data.postLine] ← XS.nullReaderBody;
        data.place ← [x: xMargin, y: yMargin + data.postLine*fontHeight]}
      ENDCASE => {
        XS.FreeReaderBytes[r: @data.lines[0], z: data.zone];
        FOR i: LineIndex IN [1..currentLine) DO
          data.lines[i-1] ← data.lines[i];
          ENDLOOP;
        Display.Shift[
          window: w,
          box: [[xMargin, fontHeight], dims],
          newPlace: [x: xMargin, y: yMargin]];
        data.lines[currentLine-1] ←
          XS.CopyToNewReaderBody[XS.ReaderFromWriter[@data.wb], data.zone];
        XS.ClearWriter[@data.wb];
        data.lines[data.postLine] ← XS.nullReaderBody;
        data.place ← [x: xMargin, y: yMargin + data.postLine*fontHeight]};
    w.InvalidateBox[[data.place, [dims.w, fontHeight]]]};

  tabChar: XS.Character = XCharSet0.Codes0.tab.ORD;
  tabWidth: CARDINAL = 40;

  PaintLine: PROCEDURE [w: Window.Handle, data: Data, r: XS.Reader] = {
    windowDims: Window.Dims = w.GetBox[].dims;
    PaintProc: SimpleTextDisplay.BufferProc = {
      d: Data = data;
      XS.AppendReader[to: @d.wb, from: string];
      d.lines[d.postLine] ← XS.ReaderFromWriter[@d.wb]↑;
      Display.Bitmap[
        w, [d.place, dims], address, bitsPerLine, Display.paintFlags];
      d.place.x ← d.place.x + dims.w;
      IF result = stop THEN {
        LastChar: XS.MapCharProc = {RETURN[stop: TRUE]};
        stopChar: XS.Character = XS.ReverseMap[string, LastChar];
        IF stopChar = tabChar THEN {
          fontHeight: CARDINAL = SimpleTextDisplay.systemFontHeight;
          newX: CARDINAL ← d.place.x + tabWidth - d.place.x MOD tabWidth;
          Display.White[
            w, [d.place, [w: newX - d.place.x, h: fontHeight]]];
          d.place.x ← newX;
          IF newX < CARDINAL[windowDims.w] THEN RETURN[TRUE]}};
      RETURN[FALSE]};
    [rest: r↑] ← SimpleTextDisplay.StringIntoBuffer[
      string: r, bufferProc: PaintProc, lineWidth: windowDims.w - data.place.x];
    };

  Repaint: PROCEDURE [window: Window.Handle] = {
    windowDims: Window.Dims = window.GetBox[].dims;
    data: Data ← LocalFind [window];
    fontHeight: CARDINAL = SimpleTextDisplay.systemFontHeight;
    MarkBox: PROCEDURE [w: Window.Handle, box: Window.Box] = {
      first, last: CARDINAL;
      lastY: INTEGER = box.place.y + box.dims.h;
      IF box.place.y < yMargin THEN first ← 0
      ELSE first ← CARDINAL[box.place.y-yMargin]/fontHeight;
      IF lastY < yMargin THEN last ← 0
```

```
      ELSE last ← MIN[data.lines.length, CARDINAL[lastY-yMargin]/fontHeight];
      FOR i: LineIndex IN [first..last] DO
        data.invalid[i] ← TRUE;
        ENDLOOP};
    place: Window.Place;
    PaintProc: SimpleTextDisplay.BufferProc = {
      Display.Bitmap[
        window, [place, dims], address, bitsPerLine, Display.paintFlags];
      place.x ← place.x + dims.w;
      IF result = stop THEN {
        LastChar: XS.MapCharProc = {RETURN[stop: TRUE]};
        stopChar: XS.Character = XS.ReverseMap[string, LastChar];
        IF stopChar = tabChar THEN {
          fontHeight: CARDINAL = SimpleTextDisplay.systemFontHeight;
          newX: CARDINAL ← place.x + tabWidth - place.x MOD tabWidth;
          Display.White[
            window, [place, [w: newX - data.place.x, h: fontHeight]]];
          place.x ← newX;
          IF newX < CARDINAL[windowDims.w] THEN RETURN[TRUE]}};
      RETURN[FALSE]};
    FOR i: LineIndex IN [0..data.invalid.length) DO
      data.invalid[i] ← FALSE; ENDLOOP;
    Window.EnumerateInvalidBoxes[window: window, proc: MarkBox];
    FOR i: LineIndex IN [0..data.lines.length) DO
      IF data.invalid[i] THEN {
        place ← [x: xMargin, y: yMargin + i * fontHeight];
        IF ~XS.Empty[@data.lines[i]] THEN
          [] ← SimpleTextDisplay.StringIntoBuffer[
            string: @data.lines[i], bufferProc: PaintProc,
            lineWidth: windowDims.w - xMargin]};
      ENDLOOP};

  XFormatObject: PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [o: XFormat.Object] = {RETURN [[proc: XFormatProc, data: window]]};

  XFormatProc: XFormat.FormatProc = {Post[h.data, r, FALSE]};

  END...
```

```
-- File: Scrollbar.mesa - last edit:
-- Breisacher.ES       23-Aug-85 17:28:04

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Window USING [Handle];

Scrollbar: DEFINITIONS = BEGIN

  -- Types:

  Percent: TYPE = [0..100];
  ScrollFlavor: TYPE = {
    pageFwd, pageBwd, forward, backward, jumpFwd, jumpBwd};
    << jumpFwd, jumpBwd - jump scrolling
       forward, backward - continuous scrolling, unit is arbitrary.
       pageFwd, pageBwd - paging >>
  ThumbFlavor: TYPE = {downClick, track, upClick, enter, exit};
  Type: TYPE = {horizontal, vertical};
  ArrowScrollAction: TYPE = {start, go, stop};
  Where: TYPE = {leftTop, rightBottom, both}; -- which side of window scrollbar goes.

  JumpScrollProc: TYPE = PROC [
    windowToBeScrolled: Window.Handle,
    flavor: ScrollFlavor[jumpFwd..jumpBwd],
    percent: Percent];
    -- percent is relative to window.
  ScrollbarProc: TYPE = PROC [windowToBeScrolled: Window.Handle]
    RETURNS [offset, portion: Percent];
  SingleScrollProc: TYPE = PROC [
    windowToBeScrolled: Window.Handle, flavor: ScrollFlavor[pageFwd..backward],
    arrowScrollAction: ArrowScrollAction ← go];
  ThumbScrollProc: TYPE = PROC [
    windowToBeScrolled: Window.Handle, flavor: ThumbFlavor, m, outOfN: INTEGER];
    -- Percent is not used to provide compatibility with SWS.

  -- Errors

  ErrorCode: TYPE = {alreadyExists, doesNotExist};
  Error: ERROR [code: ErrorCode];

  -- Procedures

  Adjust: PROCEDURE [windowToBeScrolled: Window.Handle];

  Create: PROCEDURE [
    windowToBeScrolled: Window.Handle,
    type: Type ← vertical,
    where: Where ← rightBottom,
    single: SingleScrollProc ← NIL,
    jump: JumpScrollProc ← NIL,
    thumb: ThumbScrollProc ← NIL,
    scrollbar: ScrollbarProc ← NIL,
    zone: UNCOUNTED ZONE ← NIL];

  Destroy: PROC [windowToBeScrolled: Window.Handle, type: Type];

  GetScrollProcs: PROCEDURE [windowToBeScrolled: Window.Handle, type: Type]
    RETURNS [
      single: SingleScrollProc,
      jump:JumpScrollProc,
      thumb: ThumbScrollProc];

  SetSingleScrollProc: PROCEDURE [
    windowToBeScrolled: Window.Handle, type: Type, scroll: SingleScrollProc]
    RETURNS [old: SingleScrollProc];

  SetJumpScrollProc: PROCEDURE [
    windowToBeScrolled: Window.Handle, type: Type, scroll: JumpScrollProc]
    RETURNS [old: JumpScrollProc];

  SetThumbScrollProc: PROCEDURE [
    windowToBeScrolled: Window.Handle, type: Type, scroll: ThumbScrollProc]
    RETURNS [old: ThumbScrollProc];
```

```
  PaintThumbFeedBack: PROCEDURE [
    windowToBeScrolled: Window.Handle,
    offset: Percent,
    portion: Percent ← 0];

  EraseThumbFeedBack: PROCEDURE [windowToBeScrolled: Window.Handle];

  PercentOf: PROCEDURE [v: INTEGER, p: Percent]
    RETURNS [INTEGER];
  -- expresses p in terms of v
  --  example:  m ← PercentOf[OutOfN, offset]

  Percentage: PROCEDURE [part, full: INTEGER]
    RETURNS [Percent];
  -- returns the percentage of part to full
  --  expample: offset ← Percentage[m, OutOfN]

  END...
```

```
-- File: ScrollbarImpl.mesa - last edit:
-- guzik.ES            17-Jan-86 16:38:17
-- Breisacher.ES       27-Aug-85 10:39:58

-- Copyright (C) 1985, 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Context USING [Create, Data, Destroy, Find, Type, UniqueType],
  Cursor USING [Fetch, Object, Store],
  Display,
  Heap USING [Create, Delete],
  Inline,
  Process USING [priorityBackground, SetPriority],
  Scrollbar,
  TIP USING [CallBack, CallBackNotifyProc, ClearManager, NotifyProc, ResultObject, Results, SetManager,
  SetTableAndNotifyProc],
  TIPStar USING [GetTable, NormalTable],
  Window;

ScrollbarImpl: PROGRAM
  IMPORTS Atom, Context, Cursor, Display, Heap, Inline, Process, TIP, TIPStar, Window
  EXPORTS Scrollbar = BEGIN OPEN Scrollbar;

  -- TYPEs

  Client: TYPE = LONG POINTER TO ClientObject;
  ClientObject: TYPE = RECORD [
    horizontal: ScrollbarContext ← NIL,
    vertical: ScrollbarContext ← NIL,
    clientWindow: Window.Handle ← NIL];  -- scrollable window

  ScrollbarContext: TYPE = LONG POINTER TO ScrollbarContextObject;
  ScrollbarContextObject: TYPE = RECORD [
    client: Client ← NIL,
    diamond: BOOLEAN ← FALSE,
    thumbOffset: Percent ← 0,
    thumbPortion: Percent ← 0,
    jump: JumpScrollProc ← NIL,
    thumb: ThumbScrollProc ← NIL,
    type: Type ← vertical,
    scrollbar: ScrollbarProc ← NIL,
    single: SingleScrollProc ← NIL,
    -- need two windows here for where=both, do it later.
    scrollWindow: Window.Handle ← NIL, -- scrollbar window
    where: Where ← rightBottom,
    zone: UNCOUNTED ZONE ← NIL,
    clientZone: BOOLEAN ← FALSE,
    activePiece: Piece ← nil,
    activeBox: Window.Box ← Window.nullBox];

  Piece: TYPE = {nil, grabber, topArrow, plus, minus, bottomArrow,
    thumb, leftArrow, rightArrow, leftMargin, rightMargin,
    spareScroll1, spareScroll2, spareScroll3, spareScroll4};
  ScrollPiece: TYPE = Piece [topArrow..spareScroll4];
  VerticalPiece: TYPE = ScrollPiece [topArrow..thumb];
  HorizontalPiece: TYPE = ScrollPiece [leftArrow..rightMargin];

  MouseAction: TYPE = {down, motion, up, exit, enter};

  Atoms: TYPE = RECORD [
    adjustDown, adjustMotion, exit, enter, stop, open, pointDown, pointMotion, pointUp, adjustUp,
    moveModeDown, copyModeDown, sameAsModeDown, copyModeMotion, moveModeMotion, sameAsModeMotion,
    copyModeUp, moveModeUp, sameAsModeUp, moveModeExit, copyModeExit, sameAsModeExit, moveModeEnter,
    copyModeEnter, sameAsModeEnter: Atom.ATOM ← Atom.null];

    -- from WindowScrollbar2.mesa
  BMRec: TYPE = RECORD [
    w, h: INTEGER,
    wpl: CARDINAL,
    bm: LONG DESCRIPTOR FOR ARRAY OF WORD];

  -- Data

  zone: UNCOUNTED ZONE ← Heap.Create[initial: 1];
```

```
    clientContext: Context.Type ← Context.UniqueType[];
    scrollContext: Context.Type ← Context.UniqueType[];

    atoms: LONG POINTER TO Atoms ← zone.NEW[Atoms];

    savedCursor: LONG POINTER TO Cursor.Object ← zone.NEW[Cursor.Object];

    -- Constants

    noPlace: Window.Place = [-1,-1];

      -- stolen from WindowBasicsImpl:
      insideBorder: INTEGER = 1;
      scrollWidth: INTEGER = 10;
      scrollEdgeTotal: INTEGER = insideBorder+scrollWidth;
      grabberDims: Window.Dims = --[10, 10]--[0, 0];
      rightBorderWidth: INTEGER = 2;
      arrowSize: INTEGER = 34;
      pointerSize: INTEGER = 12;
      teeSize: INTEGER = 16;
      grabberEdgeTotal: INTEGER = --insideBorder+grabberDims.w--0;

    -- error

    Error: PUBLIC SIGNAL [code: ErrorCode] = CODE;

    -- Procedures

    Adjust: PUBLIC PROCEDURE [windowToBeScrolled: Window.Handle] = {
      client: Client ← GetClient [windowToBeScrolled];
      scrollBox: Window.Box ← Window.nullBox;
      windowBox:Window.Box ← windowToBeScrolled.GetBox[];
      IF client.vertical # NIL THEN {
        IF windowBox.dims.h = 0 THEN {
          client.vertical.scrollWindow.SlideAndSize [Window.nullBox];
          IF client.horizontal # NIL THEN
            client.horizontal.scrollWindow.SlideAndSize [Window.nullBox];
          RETURN};
        scrollBox.place.y ← windowBox.place.y;
        windowBox.dims.w ← windowBox.dims.w-scrollEdgeTotal;
        scrollBox.place.x ← windowBox.place.x+windowBox.dims.w;
        scrollBox.dims ← [scrollEdgeTotal, windowBox.dims.h];
        client.vertical.scrollWindow.SlideAndSize [scrollBox];
        client.vertical.scrollWindow.InvalidateBox [
          [[0,0], scrollBox.dims] ] };
      IF client.horizontal # NIL THEN {
        windowBox.dims.h ← windowBox.dims.h-scrollEdgeTotal;
        scrollBox.place.y ← windowBox.place.y+windowBox.dims.h;
        scrollBox.place.x ← windowBox.place.x;
        scrollBox.dims ← [windowBox.dims.w, scrollEdgeTotal];
        client.horizontal.scrollWindow.SlideAndSize [scrollBox];
        client.horizontal.scrollWindow.InvalidateBox [
          [[0,0], scrollBox.dims] ] };
      windowToBeScrolled.SlideAndSize [windowBox];
      };

    Create: PUBLIC PROCEDURE [
      windowToBeScrolled: Window.Handle,
      type: Type ← vertical,
      where: Where ← rightBottom,
      single: SingleScrollProc ← NIL,
      jump: JumpScrollProc ← NIL,
      thumb: ThumbScrollProc ← NIL,
      scrollbar: ScrollbarProc ← NIL,
      zone: UNCOUNTED ZONE ← NIL] = {

      clientZone: BOOLEAN ← zone # NIL;
      me: ScrollbarContext;
      client: Client;
      IF zone = NIL THEN zone ← Heap.Create [initial:1];
      client ← GetOrCreateClient [windowToBeScrolled, zone];
      SELECT type FROM
        vertical =>
          IF client.vertical # NIL THEN ERROR Error [alreadyExists];
        horizontal =>
          IF client.horizontal # NIL THEN ERROR Error [alreadyExists];
```

```
      ENDCASE;
    me ← zone.NEW [ScrollbarContextObject ← [
      client: client,
      jump: jump, thumb: thumb, type: type,
      scrollbar: scrollbar, single: single,
      where: where, zone: zone, clientZone: clientZone,
      scrollWindow: CreateScrollbarWindow [windowToBeScrolled, client, type]
      ]];
    client.clientWindow ← windowToBeScrolled;
    SELECT type FROM
      vertical => client.vertical ← me;
      horizontal => client.horizontal ← me;
      ENDCASE;
    Context.Create [scrollContext, me, DestroyScroll, me.scrollWindow];
    TIP.SetTableAndNotifyProc [window: me.scrollWindow,
      table: TIPStar.NormalTable[], notify: NotifyProc];
    };

  CreateScrollbarWindow: PROCEDURE [viewer: Window.Handle,
    client: Client, type: Type]
    RETURNS [scrollWindow: Window.Handle] = {
    viewerSibling: Window.Handle ← viewer.GetSibling [];
    scrollWindow ← Window.Create [
      display: SELECT type FROM
        horizontal => PaintHorizontal,
        ENDCASE => PaintVertical,
      box: Window.nullBox,
      parent: viewer.GetParent [],
      sibling: SELECT type FROM
        vertical => IF client.horizontal # NIL THEN
          client.horizontal.scrollWindow ELSE viewerSibling,
        ENDCASE => IF client.vertical # NIL THEN
          client.vertical.scrollWindow.GetSibling []
          ELSE viewerSibling];
    SELECT type FROM  -- adjust viewer window geneaology.
      vertical => {
        [] ← viewer.SetSibling [scrollWindow];
        IF client.horizontal # NIL THEN
          [] ← client.horizontal.scrollWindow.SetSibling [viewerSibling];
        };
      horizontal => IF client.vertical # NIL THEN {
          [] ← viewer.SetSibling [client.vertical.scrollWindow];
          [] ← client.vertical.scrollWindow.SetSibling[scrollWindow]}
        ELSE [] ← viewer.SetSibling [scrollWindow];
      ENDCASE;
    };

  Destroy: PUBLIC PROCEDURE [windowToBeScrolled: Window.Handle,
    type: Type] = {
    client: Client ← Context.Find [clientContext, windowToBeScrolled];
    IF client = NIL THEN RETURN; -- already gone
    SELECT type FROM
      vertical =>
        Context.Destroy [scrollContext, client.vertical.scrollWindow];
      horizontal =>
        Context.Destroy [scrollContext, client.horizontal.scrollWindow];
      ENDCASE;
    };

  DestroyClient: PROCEDURE [client: Client, clientWindow: Window.Handle] = {
    zone: UNCOUNTED ZONE = SELECT TRUE FROM
      client.vertical # NIL => client.vertical.zone,
      client.horizontal # NIL => client.horizontal.zone,
      ENDCASE => NIL;
    IF zone = NIL THEN ERROR Error [doesNotExist];
    zone.FREE [@client];
    };

  DestroyScroll: PROCEDURE [me: ScrollbarContext, scrollWindow: Window.Handle] = {
    zone: UNCOUNTED ZONE = me.zone;
    IF me.clientZone THEN zone.FREE [@me] ELSE Heap.Delete [zone];
    };

  EraseThumbFeedBack: PUBLIC PROC [windowToBeScrolled: Window.Handle] = {
    --Paint white into thumbing region.
    client: Client ← GetClient [windowToBeScrolled];
```

```
    me: ScrollbarContext ← client.vertical;
    InternalPaintThumbFeedBack[me.scrollWindow, me.thumbOffset, me.thumbPortion, white];
    me.diamond ← FALSE;
    me.thumbOffset ← 0;
    me.thumbPortion ← 0;
    };

  GetOrCreateClient: PROCEDURE [clientWindow: Window.Handle,
    zone: UNCOUNTED ZONE] RETURNS [client: Client] = {
    client ← Context.Find [clientContext, clientWindow];
    IF client = NIL THEN {
      client ← zone.NEW [ClientObject];
      Context.Create [clientContext, client, DestroyClient, clientWindow];
      };
    };

  GetClient: PROCEDURE [clientWindow: Window.Handle]
    RETURNS [client: Client] = {
    client ← Context.Find [clientContext, clientWindow];
    IF client = NIL THEN ERROR Error [doesNotExist];
    };

  GetScrollbar: PROCEDURE [scrollWindow: Window.Handle]
    RETURNS [s: ScrollbarContext] = {
    s ← Context.Find [scrollContext, scrollWindow];
    IF s = NIL THEN ERROR Error [doesNotExist];
    };

  InternalPaintThumbFeedBack: PROCEDURE [
    scrollBar: Window.Handle,
    offset: Percent,
    portion: Percent,
    color: {black, white} ] = {
    -- turn the percentages into locations in the current thumbing region
    diamond: BMRec ← [w: 10, h: 9, wpl: 1, bm: DESCRIPTOR[ndBits]];
    ndBits: ARRAY [0..9) OF WORD ← [
      004000B, 016000B, 037000B, 077400B, 177600B, 077400B, 037000B, 016000B,
      004000B];
    -- determine the h of the thumb region
    thumbH: INTEGER ← scrollBar.GetBox.dims.h - rightBorderWidth -
      grabberDims.h - (2*arrowSize) - pointerSize - teeSize -
      (5*insideBorder);
    -- then express offset as a percentage of the thumb height
    place: Window.Place ← [insideBorder, insideBorder + arrowSize
     + insideBorder + teeSize
     + MIN[PercentOf[thumbH, offset], thumbH-(diamond.h+2)]];
    IF color = black THEN
      PaintBitmap[scrollBar, place, @diamond]
    ELSE Display.White[scrollBar, [place, [scrollWidth, diamond.h]]];
    };

  NotifyProc: TIP.NotifyProc = {
    OPEN atoms;
    place: Window.Place ← noPlace;
    me: ScrollbarContext ← GetScrollbar [window];
    FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
      WITH z: input SELECT FROM
        coords => place ← z.place;
        atom =>
          BEGIN
          piece: Piece ← nil;
          box: Window.Box ← Window.nullBox;
          clear: BOOLEAN ← FALSE;
          [piece, box] ← ResolveMouse [window, place];
          SELECT z.a FROM
            pointDown, adjustDown, copyModeDown,
            moveModeDown, sameAsModeDown => SELECT piece FROM
              nil => NULL;
              IN ScrollPiece =>
                clear ← ScrollNotify [window, piece, box, down, place];
              grabber => NULL; -- later, maybe
              ENDCASE;
            pointMotion, adjustMotion, copyModeMotion,
            moveModeMotion, sameAsModeMotion =>
              BEGIN
              IF piece # me.activePiece THEN {
```

```
            clear ← ScrollNotify [window, me.activePiece, me.activeBox, exit, place];
            clear ← ScrollNotify [window, piece, box, enter, place];
            };
          SELECT piece FROM
            nil => NULL;
            IN ScrollPiece =>
              clear ← ScrollNotify [window, piece, box, motion, place];
            grabber => NULL; -- later, maybe
            ENDCASE;
        END;
      pointUp, adjustUp, copyModeUp,
      moveModeUp, sameAsModeUp =>
        BEGIN
        SELECT piece FROM
          nil => NULL;
          IN ScrollPiece =>
            clear ← ScrollNotify [window, piece, box, up, place];
          grabber => NULL; -- later, maybe
          ENDCASE;
        END;
      moveModeExit, copyModeExit, sameAsModeExit, exit =>
        BEGIN
        -- Enter and exit have no coords.
        IF me.activePiece # nil THEN
          clear ← ScrollNotify [window, me.activePiece, me.activeBox, exit, place];
        END;
      ENDCASE;
    IF clear THEN {piece ← nil; box ← Window.nullBox};
    me.activePiece ← piece;
    me.activeBox ← box;
    END; -- atom
  ENDCASE;
ENDLOOP;
};

NotifyFeedback: PROCEDURE [scrollWindow: Window.Handle,
  piece: Piece, box: Window.Box, mouseAction: MouseAction] = {
  IF piece # thumb AND mouseAction # motion THEN
    Display.Invert [scrollWindow, box];
  };

PaintBitmap: PROCEDURE [window: Window.Handle, place: Window.Place,
  bmPtr: LONG POINTER TO BMRec] = {
  Display.Bitmap[
    window: window,
    box: [place, [bmPtr.w, bmPtr.h]],
    bitmapBitWidth: bmPtr.wpl*16, address: [@bmPtr.bm[0], 0, 0],
    flags: Display.paintFlags];
  };

PaintHorizontal: PROCEDURE [window: Window.Handle] =
  BEGIN
  scrollBox: Window.Box ← window.GetBox [];
  sd: Window.Dims = scrollBox.dims;
  ts: INTEGER = teeSize;
  as: INTEGER = arrowSize;
  ib: INTEGER = insideBorder;
  ge: INTEGER = grabberEdgeTotal;
  gd: Window.Dims = grabberDims;
  newLeftTee: BMRec ← [w: 7, h: 6, wpl: 1, bm: DESCRIPTOR[nltBits]];
    nltBits: ARRAY [0..6) OF WORD ← [
    140603B, 140451B, 177000B, 177400B, 140472B, 140752B];
  newRightArrow: BMRec ← [w: 12, h: 6, wpl: 1, bm: DESCRIPTOR[nraBits]];
    nraBits: ARRAY [0..6) OF WORD ← [
    001403B, 000711B, 177760B, 177760B, 000717B, 001412B];
  newLeftArrow: BMRec ← [w: 12, h: 6, wpl: 1, bm: DESCRIPTOR[nlaBits]];
    nlaBits: ARRAY [0..6) OF WORD ← [
    006003B, 034011B, 177760B, 177760B, 034017B, 006012B];
  newRightTee: BMRec ← [w: 7, h: 6, wpl: 1, bm: DESCRIPTOR[nrtBits]];
    nrtBits: ARRAY [0..6) OF WORD ← [
    003554B, 003104B, 177567B, 177101B, 003162B, 003567B];

  Display.White[ window: window, box: [[0,0], sd]];
  Display.Black[  -- inside border
    window: window, box: [[0, 0], [sd.w, ib]]];
  Display.Black[  -- line at right of control point
```

```
      window: window, box: [[gd.w, 0], [ib, sd.h]]];
    Display.Black[  -- line at right of left tee
      window: window, box: [[ge+ts, 0], [ib, sd.h]]];
    Display.Black[  -- line at left of right tee
      window: window, box: [[sd.w-ts-ib, 0], [ib, sd.h]]];
    --WindowGrabber.Paint[window, [[0, ib], gd], lowerLeft];
    PaintBitmap[window,
      [ge+(ts-newLeftTee.w)/2,
        ib+(sd.h-newLeftTee.h)/2],
      @newLeftTee];
    PaintBitmap[window,
      [ge+as, ib+(sd.h-newRightArrow.h)/2],
      @newRightArrow];
    PaintBitmap[window,
      [sd.w-as-newLeftArrow.w,
        ib+(sd.h-newLeftArrow.h)/2],
      @newLeftArrow];
    PaintBitmap[window,
      [sd.w-newRightTee.w-(ts-newRightTee.w)/2,
        ib+(sd.h-newRightTee.h)/2],
      @newRightTee];
    END;  -- PaintHorizontal

  PaintThumbFeedBack: PUBLIC PROC [windowToBeScrolled: Window.Handle,
    offset: Percent, portion: Percent ← 0] = {
    client: Client ← GetClient [windowToBeScrolled];
    me: ScrollbarContext ← client.vertical;
    me.diamond ← TRUE;
    me.thumbOffset ← offset;
    me.thumbPortion ← portion;
    InternalPaintThumbFeedBack[me.scrollWindow, me.thumbOffset, me.thumbPortion, black];
    };

  PaintVertical: PROCEDURE [window: Window.Handle] =
    BEGIN
    scrollBox: Window.Box ← window.GetBox [];
    context: ScrollbarContext ← GetScrollbar [window];
    sd: Window.Dims = scrollBox.dims;
    ts: INTEGER = teeSize;
    as: INTEGER = arrowSize;
    ib: INTEGER = insideBorder;
    gd: Window.Dims = grabberDims;
    ge: INTEGER = grabberEdgeTotal;

    newDownArrow: BMRec ← [w: 6, h: 12, wpl: 1, bm: DESCRIPTOR[ndaBits]];
      ndaBits: ARRAY [0..12) OF WORD ← [
      031554B, 030104B, 031567B, 031101B, 031162B, 031567B, 132544B, 176555B,
      075440B, 074440B, 031440B, 030454B];
    newMinus: BMRec ← [w: 6, h: 2, wpl: 1, bm: DESCRIPTOR[nmBits]];
      nmBits: ARRAY [0..2) OF WORD ← [177054B, 176061B];
    newPlus: BMRec ← [w: 6, h: 6, wpl: 1, bm: DESCRIPTOR[npBits]];
      npBits: ARRAY [0..6) OF WORD ← [
      031135B, 030415B, 177145B, 177504B, 031567B, 031101B];
    newUpArrow: BMRec ← [w: 6, h: 12, wpl: 1, bm: DESCRIPTOR[nuaBits]];
      nuaBits: ARRAY [0..12) OF WORD ← [
      031554B, 030104B, 075567B, 075101B, 177162B, 133567B, 030544B, 030555B,
      031440B, 030440B, 031440B, 030454B];
    upArrow, minus, plus, downArrow: LONG POINTER TO BMRec ← NIL;

    Display.White[window: window, box: [[0,0], sd]];
    Display.Black[window, [[0, 0], [ib, sd.h]]]; -- inside border for scrollbar
    Display.Black [window,
      [[ib, sd.h-ge], [gd.w, ib]]]; -- edge between grabber and bottom arrow
    Display.Black[window, [[ib, as], [sd.w-ib, ib]]];
    Display.Black[window, [[ib, as+ts+ib], [sd.w-ib, ib]]];
    Display.Black[window, [[ib, sd.h-ge-as-ib-ts-ib], [sd.w-ib, ib]]];
    Display.Black[window, [[ib, sd.h-ge-as-ib], [sd.w-ib, ib]]];

    upArrow ← @newUpArrow; minus ← @newMinus;
    plus ← @newPlus; downArrow ← @newDownArrow;
    PaintBitmap[window,
      [ib+(sd.w-downArrow.w)/2, (as-downArrow.h)/2],
      downArrow];
    PaintBitmap[window,
      [ib+(sd.w-minus.w)/2, as+ib+(ts-minus.h)/2],
      minus];
```

```
    IF context.diamond THEN
      InternalPaintThumbFeedBack[window, context.thumbOffset, context.thumbPortion, black];

    PaintBitmap[window,
      [ib+(sd.w-plus.w)/2, sd.h-ge-as-ib-ts+(ts-plus.h)/2],
      plus];
    PaintBitmap[window,
      [ib+(sd.w-upArrow.w)/2, sd.h-ge-as+(as-upArrow.h)/2],
      upArrow];
    --WindowGrabber.Paint[window, [[ib, sd.h-gd.h], gd], lowerRight];
    END;

  Percentage: PUBLIC PROC [part, full: INTEGER] RETURNS [Percent] = {
    RETURN [ SELECT TRUE FROM
        full <= 0 => 0,
        part <= 0 => 0,
        part >= full => 100,
        ENDCASE => Inline.LongDiv[Inline.LongMult[part, 100], full]] };

  PercentOf: PUBLIC PROC [v: INTEGER, p: Percent] RETURNS [INTEGER] = {
    RETURN [Inline.LongDiv[Inline.LongMult[v, p] + 50, 100]] };

  ResolveMouse: PROCEDURE [scrollWindow: Window.Handle, place: Window.Place]
    RETURNS [piece: Piece, box: Window.Box ← Window.nullBox] = {
    me: ScrollbarContext ← GetScrollbar [scrollWindow];
    as: INTEGER = arrowSize;
    ts: INTEGER = teeSize;
    ib: INTEGER = insideBorder;
    sw: INTEGER = scrollWidth;
    ge: INTEGER = grabberEdgeTotal;
    sd: Window.Dims = Window.GetBox [scrollWindow].dims;
    gd: Window.Dims = grabberDims;
    th: INTEGER = sd.h - rightBorderWidth - gd.h - (2*as) - pointerSize - ts - (5*ib);

    IF place = noPlace THEN RETURN [nil, Window.nullBox];
    SELECT me.type FROM
      vertical => SELECT place.y FROM
        IN [0..as) => RETURN [piece: topArrow, box: [[ib, 0], [sw, as]]];
        IN [as..as+ib+ts) => RETURN[piece: minus, box: [[ib, as+ib], [sw, ts]]];
        IN [as+ib+ts..sd.h-ge-as-ib-ts) =>
          RETURN[piece: thumb, box: [[ib, as+ib+ts], [sw, th]]];
        IN [sd.h-ge-as-ib-ts..sd.h-ge-as) =>
          RETURN[piece: plus, box: [[ib, sd.h-ge-as-ib-ts], [sw, ts]]];
        IN [sd.h-ge-as..sd.h-ge) =>
          RETURN[piece: bottomArrow, box:[[ib, sd.h-as-ge], [sw, as]]];
        IN [sd.h-ge..sd.h) =>
          RETURN[piece: grabber, box: [[ib, sd.h-ge-ib], gd]];
        ENDCASE;
      horizontal => SELECT place.x FROM
        IN [0..ge) =>
          RETURN[piece: grabber, box: [[0, ib], gd]];
        IN [ge..ge+ts) =>
          RETURN[piece: leftMargin, box: [[ge, ib], [ts, sw]]];
        IN [ge+ts..ge+(sd.w-ge)/2) =>
          RETURN[piece: leftArrow, box: [[ge+ts, ib], [(sd.w-ge)/2-ts, sw]]];
        IN [ge+(sd.w-ge)/2..sd.w-ts) =>
          RETURN[piece: rightArrow, box: [[ge+(sd.w-ge)/2, ib], [(sd.w-ge)/2-ts, sw]]];
        IN [sd.w-ts..sd.w) =>
          RETURN[piece: rightMargin, box: [[sd.w-ts, ib], [ts, sw]]];
        ENDCASE;
      ENDCASE;
    };

  ScrollNotify: PROCEDURE [scrollWindow: Window.Handle,
    piece: Piece, box: Window.Box, mouseAction: MouseAction,
    place: Window.Place] RETURNS [clear: BOOLEAN ← FALSE] = {
    me: ScrollbarContext ← GetScrollbar [scrollWindow];
    scrolling: BOOLEAN ← FALSE;
    flavor: ScrollFlavor;

    ContinuousScroll: PROC = {
      Process.SetPriority [Process.priorityBackgrounc];
      me.single [windowToBeScrolled: me.client.clientWindow, flavor: flavor,
        arrowScrollAction: start];
      WHILE scrolling DO
```

```
        me.single [windowToBeScrolled: me.client.clientWindow, flavor: flavor,
          arrowScrollAction: go];
        ENDLOOP;
      me.single [windowToBeScrolled: me.client.clientWindow,
        flavor: flavor, arrowScrollAction: stop];
      };

    CallBackNotify: TIP.CallBackNotifyProc = {
      OPEN atoms;
      FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
        WITH z: input SELECT FROM
          atom => SELECT z.a FROM
            pointUp, adjustUp, copyModeUp, moveModeUp, sameAsModeUp,
            moveModeExit, copyModeExit, sameAsModeExit, exit => {
              scrolling ← FALSE;
              NotifyFeedback [scrollWindow, piece, box, exit];
              RETURN [done: TRUE] };
            ENDCASE;
          ENDCASE;
        ENDLOOP;
      RETURN [done: FALSE] };

    DoContinuous: PROCEDURE = {
      process: PROCESS;
      IF mouseAction # down THEN RETURN;
      IF me.single = NIL THEN RETURN;
      scrolling ← TRUE;
      process ← FORK ContinuousScroll[];
      TIP.CallBack [scrollWindow, TIPStar.NormalTable [], CallBackNotify];
      JOIN process;
      clear ← TRUE;
      };

    IF piece IN VerticalPiece AND me.type # vertical THEN ERROR;
    IF piece IN HorizontalPiece AND me.type # horizontal THEN ERROR;
    NotifyFeedback [scrollWindow, piece, box, mouseAction];
    IF mouseAction = motion AND piece # thumb THEN RETURN;
    IF mouseAction = up THEN clear ← TRUE;
    SELECT piece FROM
      nil => NULL;
      topArrow => {
        -- continuous single backward
        flavor ← backward;
        DoContinuous []};
      plus => IF mouseAction = up THEN {
        -- page forward
        IF me.single = NIL THEN RETURN;
        me.single [windowToBeScrolled: me.client.clientWindow,
          flavor: pageFwd, arrowScrollAction: go]};
      minus => IF mouseAction = up THEN {
        -- page backward
        IF me.single = NIL THEN RETURN;
        me.single [windowToBeScrolled: me.client.clientWindow,
          flavor: pageBwd, arrowScrollAction: go]};
      bottomArrow => {
        -- continuous single forward
        flavor ← forward;
        DoContinuous []};
      thumb => {
        -- thumb
        dap: INTEGER = arrowSize + teeSize;
        IF me.thumb = NIL THEN RETURN;
        Cursor.Fetch[savedCursor];
        -- Paint sideways caret here
        me.thumb [windowToBeScrolled: me.client.clientWindow,
          flavor: SELECT mouseAction FROM
            down => downClick,
            motion => track,
            up => upClick,
            enter => enter,
            ENDCASE => exit,
          m:  MAX[0, MIN[place.y-dap, box.dims.h]],
          outOfN: box.dims.h];
        };
      leftArrow => {
        -- continuous single backward
```

```
        flavor ← backward;
        DoContinuous []};
      rightArrow => {
        -- continuous single forward
        flavor ← forward;
        DoContinuous []};
      leftMargin => IF mouseAction = up THEN {
        -- page backward
        IF me.single = NIL THEN RETURN;
        me.single [windowToBeScrolled: me.client.clientWindow,
          flavor: pageBwd, arrowScrollAction: go]};
      rightMargin => IF mouseAction = up THEN {
        -- page backward
        IF me.single = NIL THEN RETURN;
        me.single [windowToBeScrolled: me.client.clientWindow,
          flavor: pageFwd, arrowScrollAction: go]};
      ENDCASE;
    };

  -- Init code

  InitAtoms: PROCEDURE = {
    OPEN atoms;
    adjustDown ← Atom.MakeAtom["AdjustDown"L];
    adjustMotion ← Atom.MakeAtom["AdjustMotion"L];
    exit ← Atom.MakeAtom["Exit"L];
    enter ← Atom.MakeAtom["Enter"L];
    stop ← Atom.MakeAtom["Stop"L];
    open ← Atom.MakeAtom["Open"L];
    pointDown ← Atom.MakeAtom["PointDown"L];
    pointMotion ← Atom.MakeAtom["PointMotion"L];
    pointUp ← Atom.MakeAtom["PointUp"L];
    adjustUp ← Atom.MakeAtom["AdjustUp"L];
    copyModeDown ← Atom.MakeAtom["CopyModeDown"L];
    moveModeDown ← Atom.MakeAtom["MoveModeDown"L];
    sameAsModeDown ← Atom.MakeAtom["SameAsModeDown"L];
    copyModeMotion ← Atom.MakeAtom["CopyModeMotion"L];
    moveModeMotion ← Atom.MakeAtom["MoveModeMotion"L];
    sameAsModeMotion ← Atom.MakeAtom["SameAsModeMotion"L];
    copyModeUp ← Atom.MakeAtom["CopyModeUp"L];
    moveModeUp ← Atom.MakeAtom["MoveModeUp"L];
    sameAsModeUp ← Atom.MakeAtom["SameAsModeUp"L];
    moveModeExit ← Atom.MakeAtom["MoveModeExit"L];
    copyModeExit ← Atom.MakeAtom["CopyModeExit"L];
    sameAsModeExit ← Atom.MakeAtom["SameAsModeExit"L];
    moveModeEnter ← Atom.MakeAtom["MoveModeEnter"L];
    copyModeEnter ← Atom.MakeAtom["CopyModeEnter"L];
    sameAsModeEnter ← Atom.MakeAtom["SameAsModeEnter"L];
    };

  -- Main code

  InitAtoms[];

  END...
```

```
-- File: VPMCommandsImpl.mesa - Last edit:
-- ANg:OSBU North:Xerox 27-May-88 15:39:37
-- JGS                 16-Apr-86 16:05:28

-- Copyright (C) 1985, 1986, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, GetProp, MakeAtom, RefPair, null],
  Auth USING [
    CallError, CallProblem, ChangeMyPasswords, ConversationHandle, CreateSimpleKey,
    CreateStrongKey, DeleteSimpleKey, DeleteStrongKey, HashSimplePassword,
    IdentityHandle, Key, PasswordStringToKey],
  CH USING [
    AddGroupMember, AddGroupProperty, AddSelf, Buffer, ChangeValueProperty,
    ConversationHandle, DeleteGroupMember, DeleteSelf, DeserializeFromRhs,
    FreeConversationHandle, FreeRhs, LookupAliasesOfName, LookupDistinguishedName,
    LookupGroupProperty, LookupValueProperty, MakeConversationHandle, MakeRhs,
    maxBufferSize, Name, NameStreamProc, ReturnCode, SerializeIntoRhs,
    unspecified],
  CHEntries USING [DescribePrimary, DescribeUserData, UserData],
  CHPIDs USING [members, user, userData, userGroup],
  Courier USING [Error, Free, Parameters],
  Heap USING [Error],
  LogWindow USING[Append],
  MCHNameExtras USING [
    EnumerateProblem, FreeNames, NameList, ParseNameList, TooManySeparators,
    UnfoundName],
  MoreCH USING [
    AddPropertyAccessMember, DeletePropertyAccessMember, LookupPropertyAccess],
  NSFile USING [Error],
  NSName USING [EquivalentNames, InitNameStore, Name, NameStore, String],
  NSString USING [FreeString, nullString, String],
  ServicesErrorMessage USING [
    MsgFromCHError, MsgFromCourierError, MsgFromNSFileError],
  Space USING [InsufficientSpace],
  TIP,
  Volume USING [InsufficientSpace],
  VPMMessages USING [Handle, Key],
  VPMPrivate USING [Data, NotBusy, UserOrGroup, Which],
  XChar USING [Character, not],
  XCharSet0 USING [Make],
  XFormat USING [CR, Decimal, Handle,  ReaderBody],
  XMessage USING [ ComposeOneToFormatHandle, ComposeToFormatHandle, Get, Handle, Compose, ComposeOne],
  XString;

VPMCommandsImpl: PROGRAM
  IMPORTS
    Atom, Auth, CH, CHEntries,  Courier, Heap, LogWindow, MCHNameExtras, MoreCH, NSFile,
    NSName, NSString, ServicesErrorMessage, Space,TIP, Volume, VPMMessages, VPMPrivate,
    XCharSet0, XFormat, XMessage, XString
  EXPORTS VPMPrivate =
  BEGIN OPEN XS: XString, VPMPrivate;

  -- TYPEs

  Data: TYPE = VPMPrivate.Data;

  identityHandle, nsNameAtom, currentUser: Atom.ATOM ← Atom.null;
  msgH: XMessage.Handle = VPMMessages.Handle[];

  Semicolon: XChar.Character = XCharSet0.Make [semicolon];
  BlankSpace: XChar.Character = XCharSet0.Make[space];

  InfoProc: TYPE = PROC [
    data: Data, ch: CH.ConversationHandle, nsName: NSName.Name];
  GroupNameProc: TYPE = PROC [
    data: Data, ch: CH.ConversationHandle, nsGroup, nsName: NSName.Name];

<< This proc will "umbrella" all the work that will retrieve and display information from
clearinghouse. This shelter error that were raised.
>>

  Umbrella: PROC [data: Data, work: PROC] = {
    work[
      ! ABORTED => {
<< This is brut force method for displaying stop and a newline after stop key has been pressed>>
```

```
        rb: XS.ReaderBody ← XS.FromSTRING["  Stopped  "];
         LogWindow.Append[ window: data.logW, r: @rb];
         rb← XS.FromSTRING["\n"];
        LogWindow.Append[ window: data.logW, r: @rb];
          CONTINUE;};
          Auth.CallError => Error[data, authError];
        Courier.Error => {
          data.msgXFH.CR[];
          ServicesErrorMessage.MsgFromCourierError[errorCode, data.msgXFH];
          ToLog[data, errorOccured];
          CONTINUE};
        NSFile.Error => {
          data.msgXFH.CR[];
          ServicesErrorMessage.MsgFromNSFileError[error, data.msgXFH];
          ToLog[data, errorOccured];
          CONTINUE}; Space.InsufficientSpace => Error[data, notEnoughMemory];
        Volume.InsufficientSpace => Error[data, notEnoughDisk];
        Heap.Error => Error[data, programBug]]};

<< Retrieve defined set of summary information of a user or group.
>>

  Summary: PUBLIC PROC [data: Data, name: XS.ReaderBody, ug: UserOrGroup] = {

<< Use to find summary information for both individual user or group.
   User summary info :
      user Remark, file Service description.
   Group summary info:
      group Repark, number of members, list of owners of DL, list of members of DL, list of friends of
      DL.
>>

    SummaryInfo: InfoProc = {
      Done: PROC = {IF buffer # NIL THEN CH.FreeRhs[buffer, data.zone]};

      buffer: CH.Buffer ← CH.MakeRhs[CH.maxBufferSize, data.zone];
      first: BOOLEAN ← TRUE;
      WriteNames: CH.NameStreamProc = {
        <<IF first THEN first ← FALSE;
        ELSE ToLog[data, listSeparator];>>
        OneWithNameToXFH[data.logXFH, threePartName, currentName,data,FALSE,TRUE]};
      howManyMembers: CARDINAL ← 0;
      MemberCount: CH.NameStreamProc = {howManyMembers ← howManyMembers + 1; };
      ok: CH.ReturnCode;
      userData: CHEntries.UserData ← [0, NIL];
      params: Courier.Parameters;
        {  -- scope for enable & exits
        ENABLE UNWIND => Done[];
        OneWithNameToXFH[data.logXFH, summaryIntro, nsName,data];

        <<  If user name # propertyIDNotFound Then
              Do summary on user.
        >>
        ok ← CH.LookupValueProperty[ch, nsName, CHPIDs.user, buffer, NIL]; -- userRemark in buffer
        IF ok.code # propertyIDNotFound THEN
          BEGIN  -- This is a user
            IF ok.code # done THEN
              CHError[data, ok];  -- raises ABORTED
            ToLog[data, userRemark];
            BufferToLog[data, buffer];
            ok ← CH.LookupValueProperty[ch, nsName, CHPIDs.userData, buffer, NIL]; -- userData in
            buffer
            IF ok.code # done THEN
              CHError[data, ok];  -- raises ABORTED
            params ← [@userData, CHEntries.DescribeUserData]; --has to separate userData.lastNameindex
            and fileService.
            IF ~CH.DeserializeFromRhs[params, data.zone, buffer] THEN
              Error[data, deserializedFailed];
            OneWithNameToXFH[data.logXFH, fileService, userData.fileService,data];-- only want
            fileService.
            Courier.Free[params, data.zone]; -- Free any data in serialization.
          END;
         << If group name # propertyIDNotFound Then
              Do summary on group.
         >>
        ok ← CH.LookupValueProperty[ch, nsName, CHPIDs.userGroup, buffer, NIL];-- groupRemark in
```

```
        buffer.
        IF ok.code # propertyIDNotFound THEN
          BEGIN  -- This is a group
            IF ok.code # done THEN
              CHError[data, ok];  -- raises ABORTED
            ToLog[data, groupRemark];
            BufferToLog[data, buffer];
            ok ← CH.LookupGroupProperty[ch, nsName, CHPIDs.members, MemberCount, NIL];--count number of
            members
             SELECT ok.code FROM
              done, propertyIDNotFound =>
               {
                ToLog[data, numberOfMembers];
                data.logXFH.Decimal[howManyMembers]-- print number of members
                };
              ENDCASE => CHError[data, ok];  -- raises ABORTED

            ToLog[data, ownersHeader];
            first ← TRUE;
            ok ← MoreCH.LookupPropertyAccess[
              ch, nsName, CHPIDs.members, administrators, WriteNames, NIL];-- if any owner found, print
              with WriteNames proc.
            SELECT ok.code FROM
              done => IF first THEN ToLog[data, none];
              propertyIDNotFound => ToLog[data, none]
              ENDCASE => CHError[data, ok];  -- raises ABORTED

            ToLog[data, friendsHeader];
            first ← TRUE;
            ok ← MoreCH.LookupPropertyAccess[
              ch, nsName, CHPIDs.members, selfControllers, WriteNames, NIL]; --if any friend found,
              print with WriteNames proc.
            SELECT ok.code FROM
              done => IF first THEN ToLog[data, none];
              propertyIDNotFound => ToLog[data, none]
              ENDCASE => CHError[data, ok];  -- raises ABORTED
          END;
        };  --ELBANE
      Done[]};


    Work: PROC = {
      OneToLog[data, summaryStart, @name];
      DoList[data, @name, ug, SummaryInfo];
      };

        -- Summary Main

    Umbrella[data, Work];
    XS.FreeReaderBytes[@name, data.zone ! Heap.Error => CONTINUE];
    NotBusy[data];
    RETURN;
    }; -- END Summary

  Matches: PUBLIC PROC [data: Data, name: XS.ReaderBody, ug: UserOrGroup] = {
    first: BOOLEAN ← TRUE;
    -- If it's the first name of the list then don't put listSeparator infront of it.
    MatchesInfo: InfoProc = {
      IF first THEN
        first ← FALSE;
      <<ELSE
        ToLog[data, listSeparator];>>
      OneWithNameToXFH[data.logXFH, threePartName, nsName,data,FALSE, TRUE];
    };

    Work: PROC = {
      OneToLog[data, matchesIntro, @name];
      DoList[data, @name, ug, MatchesInfo];
      IF first THEN ToLog[data, none];
     };
    Umbrella[data, Work];
    XS.FreeReaderBytes[@name, data.zone ! Heap.Error => CONTINUE];
    NotBusy[data];
    RETURN};

<< Find all members belonging to the group.
```

```
    >>
    Members: PUBLIC PROC [data: Data, name: XS.ReaderBody] = {
      first: BOOLEAN ← TRUE;

      MembersInfo: InfoProc = {
        WriteNames: CH.NameStreamProc = {
         << IF first THEN
            first ← FALSE
          ELSE
           ToLog[data, listSeparator];>>
          OneWithNameToXFH[data.logXFH, threePartName, currentName,data,FALSE,TRUE]
        };

        ok: CH.ReturnCode;
        OneWithNameToXFH[data.logXFH, membersIntro, nsName,data];
        ToLog[data, membersHeader];
        ok ← CH.LookupGroupProperty[ch, nsName, CHPIDs.members, WriteNames, NIL];--find all members that
        matches nsNames.  Print it using the format of WriteNames.
        SELECT ok.code FROM
          done => IF first THEN ToLog[data, none];
          propertyIDNotFound => ToLog[data, none];
          ENDCASE => CHError[data, ok]
      };

      Work: PROC = {
        OneToLog[data, membersStart, @name];
        DoList[data, @name, group, MembersInfo];
        IF first THEN ToLog[data, none]};
      Umbrella[data, Work];
      XS.FreeReaderBytes[@name, data.zone ! Heap.Error => CONTINUE];
      NotBusy[data];
      RETURN};

  << Find aliases belonging to user or group.
  >>
    Aliases: PUBLIC PROC [data: Data, name: XS.ReaderBody, ug: UserOrGroup] = {

      AliasInfo: InfoProc = {
        first: BOOLEAN ← TRUE;

        WriteNames: CH.NameStreamProc = {
          <<IF first THEN
           first ← FALSE;
          ELSE
           list Separator is added in OneWithNameToXFH
           ToLog[data, listSeparator];>>
            OneWithNameToXFH[data.logXFH, threePartName, currentName,data,FALSE,TRUE]
        };

        ok: CH.ReturnCode;
        OneWithNameToXFH[data.logXFH, distinguishedNameHeader, nsName,data];
        ToLog[data, aliasHeader];
        ok ← CH.LookupAliasesOfName[ch, nsName, WriteNames, NIL];-- find all aliases belonging to nsName.
        SELECT ok.code FROM
          done => IF first THEN ToLog[data, none];
          propertyIDNotFound => ToLog[data, none];
          ENDCASE => CHError[data, ok];
      };

      Work: PROC = {
        OneToLog[data, aliasIntro, @name]; DoList[data, @name, ug, AliasInfo]};
      Umbrella[data, Work];
      XS.FreeReaderBytes[@name, data.zone];
      NotBusy[data];
      RETURN
    }; -- end Aliases.

    AddSelf: PUBLIC PROC [data: Data, name: XS.ReaderBody] = {

      AddSelfInfo: InfoProc = {
        rc: CH.ReturnCode;
        OneWithNameToXFH[data.logXFH, addSelfIntro, nsName,data];
        rc ← CH.AddSelf[ch, nsName, CHPIDs.members, NIL]; -- add self to nsName as member.
        IF rc.code = propertyIDNotFound THEN {  -- add property & retry
          rc ← CH.AddGroupProperty[ch, nsName, CHPIDs.members, NIL, NIL];
          IF rc.code # done THEN CHError[data, rc];
```

```
        rc ← CH.AddSelf[ch, nsName, CHPIDs.members, NIL]};
      SELECT rc.code FROM
        noChange => ToLog[data, alreadyMember];
        done => ToLog[data, ok];
        propertyIDNotFound => ToLog[data, none];
        overflowOfDataBase => Error[data, dataBaseFull];
        ENDCASE => CHError[data, rc]
    };

    Work: PROC = {
      OneToLog[data, addSelfStart, @name];
      DoList[data, @name, group, AddSelfInfo]
    };

    Umbrella[data, Work];
    XS.FreeReaderBytes[@name, data.zone ! Heap.Error => CONTINUE];
    NotBusy[data];
    RETURN
  };


  RemoveSelf: PUBLIC PROC [data: Data, name: XS.ReaderBody] = {

    RemoveSelfInfo: InfoProc = {
      rc: CH.ReturnCode;
      OneWithNameToXFH[data.logXFH, removeSelfIntro, nsName,data];
      rc ← CH.DeleteSelf[ch, nsName, CHPIDs.members, NIL];-- remove self as member of group
      SELECT rc.code FROM
        noSuchLocal, noSuchDomain, noSuchOrg, propertyIDNotFound, noChange =>
          ToLog[data, notAMember];
        done => ToLog[data, ok];
        overflowOfDataBase => Error[data, dataBaseFull];
        ENDCASE => CHError[data, rc]
    };

    Work: PROC = {DoList[data, @name, group, RemoveSelfInfo]};
    Umbrella[data, Work];
    XS.FreeReaderBytes[@name, data.zone ! Heap.Error => CONTINUE];
    NotBusy[data];
    RETURN
  };--end removeself


  SetRemark: PUBLIC PROC [data: Data, group, remark: XS.ReaderBody] = {
    nsRemark: NSString.String ← XS.NSStringFromReader[@remark, data.zone];

    SetRemarkInfo: InfoProc = {
      rc: CH.ReturnCode;
      b: CH.Buffer;
      OneWithNameToXFH[data.logXFH, setRemarkIntro, nsName,data];
      b ← CH.SerializeIntoRhs[[@nsRemark, CHEntries.DescribePrimary], data.zone];
      rc ← CH.ChangeValueProperty[ch, nsName, CHPIDs.userGroup, b, NIL];-- only sa can change remark,
      should give error message otherwise.
      SELECT rc.code FROM
        done => ToLog[data, ok];
        overflowOfDataBase => Error[data, dataBaseFull];
        ENDCASE => CHError[data, rc]
    };

    Work: PROC = {
      OneToLog[data, setRemarkStart, @group];
      DoList[data, @group, group, SetRemarkInfo]
    };

    Umbrella[data, Work];
    NSString.FreeString[data.zone, nsRemark ! Heap.Error => CONTINUE];
    XS.FreeReaderBytes[@remark, data.zone ! Heap.Error => CONTINUE];
    XS.FreeReaderBytes[@group, data.zone ! Heap.Error => CONTINUE];
    NotBusy[data];
    RETURN
  };


  SetPassword: PUBLIC PROC [data: Data, individual, password: XS.ReaderBody] = {
    nsPassword: NSString.String ← XS.NSStringFromReader[@password, data.zone];
```

```
    -- I'm not sure what is going on?
        SetPasswordInfo: InfoProc = {
          namePair: Atom.RefPair = Atom.GetProp[currentUser, nsNameAtom];
          defaultName: NSName.Name = IF namePair = NIL THEN NIL ELSE namePair.value;
          identPair: Atom.RefPair = Atom.GetProp[currentUser, identityHandle];
          identity: Auth.IdentityHandle ←
            IF identPair = NIL THEN NIL ELSE identPair.value;
          name1, name2: NSName.NameStore;
          rc: CH.ReturnCode;
          NSName.InitNameStore[@name1];
          NSName.InitNameStore[@name2];
          OneWithNameToXFH[data.logXFH, setPasswordIntro, nsName,data];
          rc ← CH.LookupDistinguishedName[ch, nsName, @name1.record];
          IF rc.code # done THEN CHError[data, rc];
          rc ← CH.LookupDistinguishedName[ch, defaultName, @name2.record];
          IF rc.code # done THEN CHError[data, rc];
          IF NSName.EquivalentNames[@name1.record, @name2.record] THEN
            Auth.ChangeMyPasswords[identity, nsPassword, NIL, TRUE, TRUE]
          ELSE {  -- changing someone else's
            Auth.DeleteStrongKey[identity, nsName!
              Auth.CallError =>
               {IF reason = strongKeyDoesNotExist THEN
                  CONTINUE
                ELSE Error[data, accessRightsInsufficient];-- raises ABORTED
                }
            ];
            Auth.CreateStrongKey[
              identity, nsName, Auth.PasswordStringToKey[nsPassword]];

            Auth.DeleteSimpleKey[
              identity, nsName !
              Auth.CallError => {
                IF reason = simpleKeyDoesNotExist THEN
                 CONTINUE
                ELSE
                 Error[data, accessRightsInsufficient];-- raises ABORTED
                }
            ];
            Auth.CreateSimpleKey[
              identity, nsName, Auth.HashSimplePassword[nsPassword]]
           };
          ToLog[data, ok];
        }; -- END SummaryInfoProc

        ToLog[data, setPasswordStart];
        DoList[data, @individual, user, SetPasswordInfo ! ABORTED => CONTINUE];
        NSString.FreeString[data.zone, nsPassword];
        XS.FreeReaderBytes[@individual, data.zone ! Heap.Error => CONTINUE];
        XS.FreeReaderBytes[@password, data.zone ! Heap.Error => CONTINUE];
        NotBusy[data];
        RETURN
      };


    <<Adding name to group either as member, friend or owner
     Where does it check if name has already been added??>>

      AddName: PUBLIC PROC [data: Data, group, nameList: XS.ReaderBody, which: Which] =
        {

        Work: PROC = {
          proc: GroupNameProc =
            SELECT which FROM
              member => AddMember,
              friends => AddFriend,
              ENDCASE => AddOwner;
          OneToLog[data, addNamesStart, @group];
          DoGroupsAndNames[data, @group, @nameList, proc, FALSE]
        };

        AddMember: GroupNameProc = {
          rc: CH.ReturnCode;
          NameAndGroupToXFH[data.logXFH, addingMember, nsName, nsGroup,data];
          rc ← CH.AddGroupMember[ch, nsName, nsGroup, CHPIDs.members, NIL];--add name as member of group
          IF rc.code = propertyIDNotFound THEN {  -- add property & retry
            rc ← CH.AddGroupProperty[ch, nsGroup, CHPIDs.members, NIL, NIL];
```

```
        IF rc.code # done THEN CHError[data, rc];
        rc ← CH.AddGroupMember[ch, nsName, nsGroup, CHPIDs.members, NIL]};
      SELECT rc.code FROM
        done => ToLog[data, ok];
        noChange => ToLog[data, alreadyMember];
        overflowOfDataBase => Error[data, dataBaseFull];
        ENDCASE => CHError[data, rc]
    };

    AddFriend: GroupNameProc = {
      rc: CH.ReturnCode;
      NameAndGroupToXFH[data.logXFH, addingFriend, nsName, nsGroup,data];
      rc ← MoreCH.AddPropertyAccessMember[
        ch, nsName, nsGroup, CHPIDs.members, selfControllers, NIL];-- add nsName to access list that
        gives rights of individuals to add themselve to or remove themselves from  group.

      IF rc.code = propertyIDNotFound THEN {  -- add property & retry
        rc ← CH.AddGroupProperty[ch, nsGroup, CHPIDs.members, NIL, NIL];
        SELECT rc.code FROM
          done => {};
          overflowOfDataBase => Error[data, dataBaseFull];
          ENDCASE => CHError[data, rc];
        rc ← MoreCH.AddPropertyAccessMember[
          ch, nsName, nsGroup, CHPIDs.members, selfControllers, NIL]};
      SELECT rc.code FROM
        done => ToLog[data, ok];
        noChange => ToLog[data, alreadyFriend];
        overflowOfDataBase => Error[data, dataBaseFull];
        ENDCASE => CHError[data, rc]
    };

    AddOwner: GroupNameProc = {
      rc: CH.ReturnCode;
      NameAndGroupToXFH[data.logXFH, addingOwner, nsName, nsGroup ,data];
      rc ← MoreCH.AddPropertyAccessMember[
        ch, nsName, nsGroup, CHPIDs.members, administrators, NIL]; -- add nsName to acces list that
        gives rights of individuals to modify objects of the database.

      IF rc.code = propertyIDNotFound THEN {  -- add property & retry
        rc ← CH.AddGroupProperty[ch, nsGroup, CHPIDs.members, NIL, NIL];
        SELECT rc.code FROM
          done => {};
          overflowOfDataBase => Error[data, dataBaseFull];
          ENDCASE => CHError[data, rc];
        rc ← MoreCH.AddPropertyAccessMember[
          ch, nsName, nsGroup, CHPIDs.members, administrators, NIL]};
      SELECT rc.code FROM
        done => ToLog[data, ok];
        noChange => ToLog[data, alreadyOwner];
        overflowOfDataBase => Error[data, dataBaseFull];
        ENDCASE => CHError[data, rc]
    };

    Umbrella[data, Work];
    XS.FreeReaderBytes[@nameList, data.zone ! Heap.Error => CONTINUE];
    XS.FreeReaderBytes[@group, data.zone ! Heap.Error => CONTINUE];
    NotBusy[data];
    RETURN
  };

<<removing name from group either as member, friend or owner
>>

  RemoveName: PUBLIC PROC [data: Data, group, nameList: XS.ReaderBody, which: Which] = {

    Work: PROC = {
      proc: GroupNameProc =
        SELECT which FROM
          member => RemoveMember,
          friends => RemoveFriend,
          ENDCASE => RemoveOwner;
      OneToLog[data, removeNamesStart, @group];
      DoGroupsAndNames[data, @group, @nameList, proc,FALSE]
    };

    RemoveMember: GroupNameProc = {
```

```
        rc: CH.ReturnCode;
        NameAndGroupToXFH[data.logXFH, removingMember, nsName, nsGroup,data];
        rc ← CH.DeleteGroupMember[ch, nsName, nsGroup, CHPIDs.members, NIL]; -- removing member from
        group.

        SELECT rc.code FROM
          done => ToLog[data, ok];
          noSuchLocal, noSuchDomain, noSuchOrg, propertyIDNotFound, noChange =>
            ToLog[data, notAMember];
          overflowOfDataBase => Error[data, dataBaseFull];
          ENDCASE => CHError[data, rc]
      };

      RemoveFriend: GroupNameProc = {
        rc: CH.ReturnCode;
        NameAndGroupToXFH[data.logXFH, removingFriend, nsName, nsGroup,data];
        rc ← MoreCH.DeletePropertyAccessMember[
          ch, nsName, nsGroup, CHPIDs.members, selfControllers, NIL]; -- removing friend form access
          list.
        SELECT rc.code FROM
          done => ToLog[data, ok];
          noSuchLocal, noSuchDomain, noSuchOrg, propertyIDNotFound, noChange =>
            ToLog[data, notAMember];
          overflowOfDataBase => Error[data, dataBaseFull];
          ENDCASE => CHError[data, rc]
      };

      RemoveOwner: GroupNameProc = {
        rc: CH.ReturnCode;
        NameAndGroupToXFH[data.logXFH, removingOwner, nsName, nsGroup,data];
        rc ← MoreCH.DeletePropertyAccessMember[
          ch, nsName, nsGroup, CHPIDs.members, administrators, NIL]; -- removing owner from access list
        SELECT rc.code FROM
          done => ToLog[data, ok];
          noSuchLocal, noSuchDomain, noSuchOrg, propertyIDNotFound, noChange =>
            ToLog[data, notAMember];
          overflowOfDataBase => Error[data, dataBaseFull];
          ENDCASE => CHError[data, rc]
      };

  -- Main RemoveName

      Umbrella[data, Work];
      XS.FreeReaderBytes[@nameList, data.zone ! Heap.Error => CONTINUE];
      XS.FreeReaderBytes[@group, data.zone ! Heap.Error => CONTINUE];
      NotBusy[data];
      RETURN
    };-- END RemoveName


  << Call when adding or removing names from group. This works like DoList. Make connection to clearing
  house. Get list of names associated with name field. Get list of names associated with group field.
  Loop calling proc passed in.
  >>
    DoGroupsAndNames: PROC [
      data: Data, group, names: XS.Reader, proc: GroupNameProc, resolveAsterisk: BOOLEAN] = {
      nameList: MCHNameExtras.NameList ← NIL;
      groupList: MCHNameExtras.NameList ← NIL;
      ch: CH.ConversationHandle ← [NIL, NIL];

      Done: PROC = {
        IF nameList # NIL THEN MCHNameExtras.FreeNames[nameList, data.zone];
        IF groupList # NIL THEN MCHNameExtras.FreeNames[groupList, data.zone];
        IF ch.conversation # NIL THEN CH.FreeConversationHandle[@ch, data.zone]
      };

       -- Main

      IF XS.Empty[group] THEN Error[data, fillInGroup];
      IF XS.Empty[names] THEN Error[data, fillInNameList];
      BEGIN
        ENABLE UNWIND => Done[];
        ch ← MakeCh[data];  -- if problem, raises ABORTED after posting msg
        groupList ← GetNameList[data, ch, group, group,resolveAsterisk];

        IF groupList = NIL OR groupList.LENGTH = 0 THEN Error[data, noGroupMatched];
```

```
        nameList ← GetNameList[data, ch, names, any, resolveAsterisk];
        IF nameList = NIL OR nameList.LENGTH = 0 THEN Error[data, noGroupMatched];

        FOR j: CARDINAL IN [0..groupList.LENGTH) DO
          FOR i: CARDINAL IN [0..nameList.LENGTH) DO
            proc[data: data, ch: ch, nsGroup: @groupList[j], nsName: @nameList[i]]
            ENDLOOP;
          ENDLOOP;
      END;  -- ELBANE;
      ToLog[data, done];
      Done[]}; -- End of DoGroupsAndNames

  <<  Get list of name that associate with " names ".  Do display using proc passed in.
  >>
    DoList: PROC [data: Data, names: XS.Reader, ug: UserOrGroup, proc: InfoProc] =    {
        nameList: MCHNameExtras.NameList ← NIL;
        ch: CH.ConversationHandle ← [NIL, NIL];

        Done: PROC = {
          IF nameList # NIL THEN MCHNameExtras.FreeNames[nameList, data.zone];
          IF ch.conversation # NIL THEN CH.FreeConversationHandle[@ch, data.zone]
        };

         -- DoList Main

        IF XS.Empty[names] THEN
          Error[data, IF ug = user THEN fillInUser ELSE fillInGroup];
        BEGIN
          ENABLE UNWIND => Done[];

          ch ← MakeCh[data];  -- if problem, raises ABORTED after posting msg
          nameList ← GetNameList[data, ch, names, IF ug = user THEN user ELSE group,TRUE];
          IF nameList = NIL OR nameList.LENGTH = 0 THEN
            Error[data, IF ug = user THEN noUserMatched ELSE noGroupMatched];
          -- for every name in name list call proc for display.
          FOR i: CARDINAL IN [0..nameList.LENGTH) DO
            proc[data, ch, @nameList[i]] ENDLOOP;
        END;  -- ELBANE;
        ToLog[data, done];
        Done[]
      };  -- End of DoList

   <<MakeCh:
     Make clearinghouse connection. Return Clearinghouse handle.
    >>
   MakeCh: PROC [data: Data] RETURNS [ch: CH.ConversationHandle] = {
      pair: Atom.RefPair = Atom.GetProp[currentUser, identityHandle];
      identity: Auth.IdentityHandle ← IF pair = NIL THEN NIL ELSE pair.value;
      ae: Auth.CallProblem;
      handleOk: BOOLEAN;
      IF identity = NIL THEN Error[data, mustLoginIn];  -- Error raises ABORTED
      [ch, handleOk, ae] ← CH.MakeConversationHandle[identity, data.zone];
      IF ~handleOk THEN Error[data, authProblem];
      RETURN};

    UserGroupOrAny: TYPE = {user, group, any};
  <<GetNameList:
     Given Clearinghouse handle, get list of names from names,uga.   Most of the time will not reslove
     astericks,  neither local or remote. Operation done here assumes authenication has been done on
     users.
  >>

  <<3/11 - ANG - the following modification was made to  GetNameList.
     resolveAlias, expandPatterns,validateNames will be set TRUE only if names (string for Name list )
     does not contain any instance of "*".

   >>

    GetNameList: PROC [
      data: Data, ch: CH.ConversationHandle, names: XS.Reader, uga: UserGroupOrAny, resolveAsterisk :
      BOOLEAN]
      RETURNS [nameList: MCHNameExtras.NameList] = {
      temp: XS.ReaderBody ← XS.Dereference[names]; << added 3/11>>
      nss: NSString.String = XS.NSStringFromReader[names, data.zone];
      pair: Atom.RefPair = Atom.GetProp[currentUser, nsNameAtom];
      defaults: NSName.Name = IF pair = NIL THEN NIL ELSE pair.value;
```

```
      asterisk: XChar.Character = XCharSet0.Make[asterisk];
      <<XS.ScanForCharacter might raise InvalidEncoding error>>
      resolveAliases: BOOLEAN ← TRUE;
      expandPatterns: BOOLEAN ← TRUE;
      validateNames: BOOLEAN ← TRUE;
      validatePatterns: BOOLEAN ← TRUE;
      front: XS.ReaderBody;
      breakChar: XChar.Character;

      [breakChar, front] ← XS.ScanForCharacter[
    r: @temp, char: asterisk, option: ignore];

        IF breakChar # XChar.not AND ~resolveAsterisk THEN {
        -- We found a wildcard char.
        -- So, don't resolve aliases, don't validate name
          resolveAliases ← FALSE;
          expandPatterns ← FALSE;
          validateNames ← FALSE;
          validatePatterns ← FALSE};


      nameList ← MCHNameExtras.ParseNameList[
        namesString: nss, validatePatterns: validatePatterns, resolveAliases: resolveAliases,
        expandPatterns: expandPatterns, validateNames: validateNames, ch: ch, defaults: defaults,
        zone: data.zone,
        pid:
        (SELECT uga FROM
           user => CHPIDs.user,
           group => CHPIDs.userGroup,
           ENDCASE => CH.unspecified) !
        UNWIND => NSString.FreeString[data.zone, nss];
        MCHNameExtras.TooManySeparators => {  -- name is arg
          rbName: XString.ReaderBody ← XS.FromNSString[name];
          rbMsg: XString.ReaderBody ← GetMsg[invalidName];
          XMessage.ComposeOneToFormatHandle[@rbMsg, data.msgXFH, @rbName];
          RESUME
          ;
          };
        MCHNameExtras.EnumerateProblem => {  -- rc & name are args
          SELECT rc.code FROM
            noSuchLocal, noSuchDomain, noSuchOrg =>
              OneWithNameToXFH[
                data.msgXFH, IF uga = user THEN notAUser ELSE notAGroup, name,data];
            ENDCASE => {
              data.msgXFH.CR[];
              ServicesErrorMessage.MsgFromCHError[rc, data.msgXFH]};
          RESUME
          };
        MCHNameExtras.UnfoundName => {
          IF uga = any THEN RESUME;
          OneWithNameToXFH[
            xfh: data.msgXFH, name: name,
            key: IF uga = user THEN noUserMatches ELSE noGroupMatches, data: data];
          RESUME [FALSE]}; ];
      NSString.FreeString[data.zone, nss]};

  << raise error then abort.
  >>
    Error: PUBLIC PROC [data: Data, key: VPMMessages.Key] = {
      data.msgXFH.CR[];
      data.msgXFH.ReaderBody[msgH.Get[key.ORD]];
      ToLog[data, errorOccured];
      ERROR ABORTED};

    GetMsg: PROC [key: VPMMessages.Key] RETURNS [XS.ReaderBody] = INLINE {
      RETURN msgH.Get[key.ORD]};

    CHError: PUBLIC PROC [data: Data, rc: CH.ReturnCode] = {
      data.msgXFH.CR[];
      ServicesErrorMessage.MsgFromCHError[rc, data.msgXFH];
      ToLog[data, errorOccured];
      ERROR ABORTED};

  <<  Display a message in log window.  Log window is in key.

    ToLog: PUBLIC PROC [data: Data, key: VPMMessages.Key] = {
```

```
      data.logXFH.ReaderBody[msgH.Get[key.ORD]]};
  >>

    ToLog: PUBLIC PROC [data: Data, key: VPMMessages.Key] = {

       rb : XString.ReaderBody ← msgH.Get[key.ORD];
      IF TIP.UserAbort[data.logW] THEN
        ERROR ABORTED;
         LogWindow.Append[ window: data.logW, r: @rb];
    };

  <<  Compose a message with key and a reader and display in log window.>>

    OneToLog: PUBLIC PROC [data: Data, key: VPMMessages.Key, r: XS.Reader] = {
       wb: XS.WriterBody ← XS.NewWriterBody[2*XS.ByteLength[r], data.zone];
      r2 : XS.Reader;
      rb: XS.ReaderBody ← msgH.Get[key.ORD];
      <<XMessage.ComposeOneToFormatHandle[@rb, data.logXFH, r]>>
     IF TIP.UserAbort[data.logW] THEN
        ERROR ABORTED;
      XMessage.ComposeOne[@rb,@wb,r];
      r2 ← XS.ReaderFromWriter[@wb];
      LogWindow.Append[data.logW,r2];
      XS.FreeWriterBytes[@wb];
       };



  <<  display data in buffer in log window.>>
    BufferToLog: PROC [data: Data, b: CH.Buffer] = {
        strbuf: NSString.String ← NSString.nullString;
      rb2: XS.ReaderBody ← XS.nullReaderBody;
      params: Courier.Parameters ← [@strbuf, CHEntries.DescribePrimary];
      IF TIP.UserAbort[data.logW] THEN
        ERROR ABORTED;
      IF ~CH.DeserializeFromRhs[params, data.zone, b] THEN
        Error[data, deserializedFailed];
     << I don't know why Mesa language designer ever consented to concocted something as confusing as
     interface dot notation. Who ever suggested this should be shot!!!!
      data.logXFH.NSString[strbuf];
      XFormat.NSString [data.logXFH, strbuf];>>
      rb2 ← XS.FromNSString[strbuf];
      LogWindow.Append[data.logW, @rb2];        Courier.Free[params, data.zone]};


  <<  Display message with the 3part name of the group/individual to log or message window.
  >>
    OneWithNameToXFH: PROC [
      xfh: XFormat.Handle, key: VPMMessages.Key, name: NSName.Name, data: Data, endLine: BOOLEAN ← TRUE,
      addSeparator: BOOLEAN ← FALSE] = {
          r2: XS.Reader;
      wb :XS.WriterBody;
      rbName: ARRAY [0..3) OF XString.ReaderBody ← [
        XS.FromNSString[name.local], XS.FromNSString[name.domain], XS.FromNSString[
        name.org]];
      rbMsg: XS.ReaderBody ← GetMsg[key];
      IF TIP.UserAbort[data.logW] THEN
        ERROR ABORTED;

      IF xfh = data.logXFH THEN
        BEGIN
          wb  ← XS.NewWriterBody[2*XS.ByteLength[@rbMsg], data.zone];
          XMessage.Compose[@rbMsg,@wb,DESCRIPTOR[rbName]];
          IF addSeparator THEN
            XS.AppendChar[@wb, Semicolon];
            XS.AppendChar[@wb, BlankSpace];
          r2 ← XS.ReaderFromWriter[@wb];
          LogWindow.Append[data.logW, r2];
          XS.FreeWriterBytes[@wb];
        END
      ELSE
      XMessage.ComposeToFormatHandle[@rbMsg, xfh, DESCRIPTOR[rbName]]};



  << Display message with 3 part name of group and individual to log or message window.
```

```
>>
    NameAndGroupToXFH: PROC [
     xfh: XFormat.Handle, key: VPMMessages.Key, name, group: NSName.Name, data: Data] = {
         r2: XS.Reader;
     wb:XS.WriterBody;
     rbNames: ARRAY [0..6) OF XS.ReaderBody ← [
       XS.FromNSString[name.local], XS.FromNSString[name.domain], XS.FromNSString[
       name.org], XS.FromNSString[group.local], XS.FromNSString[group.domain],
       XS.FromNSString[group.org]];
     rbMsg: XS.ReaderBody ← GetMsg[key];
     IF TIP.UserAbort[data.logW] THEN
       ERROR ABORTED;

     IF xfh = data.logXFH THEN
       BEGIN
         wb← XS.NewWriterBody[2*XS.ByteLength[@rbMsg], data.zone];
         XMessage.Compose[@rbMsg,@wb,DESCRIPTOR[rbNames]];
                 r2 ← XS.ReaderFromWriter[@wb];
         LogWindow.Append[data.logW, r2];
         XS.FreeWriterBytes[@wb];
       END
     ELSE
     XMessage.ComposeToFormatHandle[@rbMsg,xfh,DESCRIPTOR[rbNames]]};



  InitAtoms: PROCEDURE = {
    currentUser ← Atom.MakeAtom["CurrentUser"L];
    identityHandle ← Atom.MakeAtom["IdentityHandle"L];
    nsNameAtom ← Atom.MakeAtom["NSName"L];
    };

  InitAtoms[];

  END.
```

```
-- File: VPMMessageFileImpl.mesa - Last edit:
-- JGS                  1-Nov-85  9:51:19

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  ApplicationFolder USING [FindDescriptionFile, FromName],
  BWSFileTypes USING [systemFileCatalog],
  Catalog USING [Open],
  Heap USING [systemZone],
  NSFile USING [
    Close, Error, Find, GetReference, Handle, nullHandle, nullReference,
    OpenByReference, Reference],
  NSString USING [FreeString, String],
  OptionFile USING [GetStringValue],
  VPMMessages,
  XMessage USING [
    DestroyMsgsProc, FreeMsgDomainsStorage, Handle, MessagesFromReference,
    MsgDomains],
  XString USING [FromSTRING, NSStringFromReader, Reader, ReaderBody];

VPMMessageFileImpl: PROGRAM
  IMPORTS
    ApplicationFolder, Catalog, Heap, NSFile, NSString, OptionFile,
    XMessage, XString
  EXPORTS VPMMessages =
  BEGIN

  h: XMessage.Handle ← NIL;

  zone: UNCOUNTED ZONE ← Heap.systemZone;

  DeleteMessages: XMessage.DestroyMsgsProc = BEGIN END;

  Handle: PUBLIC PROCEDURE RETURNS [XMessage.Handle] = {RETURN[h]};

  Init: PROCEDURE = {
    msgDomains: XMessage.MsgDomains ← XMessage.MessagesFromReference [
      file: GetMsgFile[], clientData: NIL, proc: DeleteMessages];
    h ← msgDomains[0].handle;
    XMessage.FreeMsgDomainsStorage[msgDomains]};

  GetMsgFile: PROCEDURE RETURNS [file: NSFile.Reference] = {
    internalName: XString.ReaderBody ← XString.FromSTRING["VPMaintain"L];
    messageFile: XString.ReaderBody ← XString.FromSTRING["MessageFile"L];
    folder: NSFile.Reference ← ApplicationFolder.FromName[@internalName];
    handle: NSFile.Handle ← TRASH;
    FileFromName: PROCEDURE [value: XString.Reader] = {
      nsName: NSString.String ← XString.NSStringFromReader [r: value, z: zone];
      msgFile: NSFile.Handle ← NSFile.nullHandle;
      msgFile ← NSFile.Find [
        directory: handle,
        scope: [filter: [matches[attribute: [name[nsName]]]]] !
          UNWIND => NSString.FreeString[z: zone, s: nsName];
          NSFile.Error => {msgFile ← NSFile.nullHandle; CONTINUE}];
      IF msgFile = NSFile.nullHandle THEN ERROR; -- No message file!
      file ← NSFile.GetReference[msgFile];
      NSFile.Close[msgFile];
      NSString.FreeString [z: zone, s: nsName]};
    IF folder = NSFile.nullReference THEN {
      name: XString.ReaderBody ← XString.FromSTRING ["VPMaintain.*.Messages"L];
      handle ← Catalog.Open [BWSFileTypes.systemFileCatalog];
      FileFromName [@name]}
    ELSE {
      adf: NSFile.Reference ← NSFile.nullReference;
      handle ← NSFile.OpenByReference [folder];
      adf ← ApplicationFolder.FindDescriptionFile[handle];
      OptionFile.GetStringValue[section: @internalName,
        entry: @messageFile, callBack: FileFromName, file: adf]};
    NSFile.Close[handle]};

  Init [];

END.
```

```
-- File: VPMMessages.mesa - Last edit:
-- ANg:OSBU North:Xerox 10-May-88 13:00:04
-- JGS                 16-Apr-86 16:07:58

-- Copyright (C) 1985, 1986, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  XMessage USING [Handle];

VPMMessages: DEFINITIONS = {

  -- Messages

  Key: TYPE = {
    accessRightsInsufficient, applicationName, another, level, levelChoices, group,
    summary, matches, members, aliases, addSelf, removeSelf, nameList, add,
    remove, which, whichChoices, set, remark, individual, password,
    busyTryAgain, summaryIntro, userRemark, fileService, deserializedFailed,
    groupRemark, numberOfMembers, authProblem, fillInUser, fillInGroup,
    mustLoginIn, noUserMatched, noGroupMatched, done, invalidName, notAUser,
    notAGroup, noUserMatches, noGroupMatches, listSeparator, membersIntro,
    membersHeader, none, copyLog, matchesIntro, aliasIntro,
    distinguishedNameHeader, aliasHeader, addSelfIntro, alreadyMember, ok,
    dataBaseFull, removeSelfIntro, notAMember, threePartName, resetLog,
    dropLog, setRemarkIntro, setPasswordIntro, logFileName,
    notEnoughMemory, notEnoughDisk, programBug, addingMember, addingFriend,
    addingOwner, alreadyFriend, alreadyOwner, removingMember, removingFriend,
    removingOwner, notAFriend, notAOwner, fillInNameList, authError,
    friendsHeader, ownersHeader, summaryStart, membersStart, addSelfStart,
    removeSelfStart, setRemarkStart, setPasswordStart, addNamesStart,
    removeNamesStart, errorOccured, inUse};

  Handle: PROCEDURE RETURNS [h: XMessage.Handle];

  }...
```

```
-- File: VPMMessagesImpl.mesa - last edit:
-- ANg:OSBU North:Xerox 10-May-88 12:49:21
-- JGS                 17-Apr-86 10:20:39

-- Copyright (C) 1985, 1986, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  VPMMessages USING [Key],
  XMessage USING [AllocateMessages, Handle, MsgEntry, RegisterMessages],
  XString USING [FromSTRING];

VPMMessagesImpl: PROGRAM
  IMPORTS XMessage, XString
  EXPORTS VPMMessages =
  BEGIN

  h: XMessage.Handle ← NIL;

  Handle: PUBLIC PROCEDURE RETURNS [XMessage.Handle] = {RETURN[h]};

  InitFromArray: PROCEDURE = {
    msgArray: ARRAY VPMMessages.Key OF XMessage.MsgEntry ← [
      applicationName: [
        msgKey: VPMMessages.Key.applicationName.ORD,
        msg: XString.FromSTRING["VP Maintain"L],
        type: menuItem,
        translationNote: "application name"L,
        translatable: TRUE,
        id: 0],
      another: [
        msgKey: VPMMessages.Key.another.ORD,
        msg: XString.FromSTRING["Another"L],
        type: menuItem,
        translationNote: "command to get another window"L,
        translatable: TRUE,
        id: 1],
      level: [
        msgKey: VPMMessages.Key.level.ORD,
        msg: XString.FromSTRING["Level"L],
        type: pSheetItem,
        translationNote: "choice tag for level of use"L,
        translatable: TRUE,
        id: 2],
      levelChoices: [
        msgKey: VPMMessages.Key.levelChoices.ORD,
        msg: XString.FromSTRING["Normal:0@Owner:1"L],
        type: pSheetItem,
        translationNote: ""L,
        translatable: TRUE,
        id: 3],
      group: [
        msgKey: VPMMessages.Key.group.ORD,
        msg: XString.FromSTRING["Group"L],
        type: pSheetItem,
        translationNote: "Tag for group text item"L,
        translatable: TRUE,
        id: 4],
      summary: [
        msgKey: VPMMessages.Key.summary.ORD,
        msg: XString.FromSTRING["Summary"L],
        type: pSheetItem,
        translationNote: "Summary command name"L,
        translatable: TRUE,
        id: 5],
      matches: [
        msgKey: VPMMessages.Key.matches.ORD,
        msg: XString.FromSTRING["Matches"L],
        type: pSheetItem,
        translationNote: "Matches command name"L,
        translatable: TRUE,
        id: 6],
      members: [
        msgKey: VPMMessages.Key.members.ORD,
        msg: XString.FromSTRING["Members"L],
        type: pSheetItem,
        translationNote: "Members command name"L,
```

```
      translatable: TRUE,
      id: 7],
    aliases: [
      msgKey: VPMMessages.Key.aliases.ORD,
      msg: XString.FromSTRING["Aliases"L],
      type: pSheetItem,
      translationNote: "Aliases command name"L,
      translatable: TRUE,
      id: 8],
    addSelf: [
      msgKey: VPMMessages.Key.addSelf.ORD,
      msg: XString.FromSTRING["Add Self"L],
      type: pSheetItem,
      translationNote: "Add Self command name"L,
      translatable: TRUE,
      id: 9],
    removeSelf: [
      msgKey: VPMMessages.Key.removeSelf.ORD,
      msg: XString.FromSTRING["Remove Self"L],
      type: pSheetItem,
      translationNote: "Remove Self command name"L,
      translatable: TRUE,
      id: 10],
    nameList: [
      msgKey: VPMMessages.Key.nameList.ORD,
      msg: XString.FromSTRING["Name List"L],
      type: pSheetItem,
      translationNote: "Name List text item tag"L,
      translatable: TRUE,
      id: 11],
    add: [
      msgKey: VPMMessages.Key.add.ORD,
      msg: XString.FromSTRING["Add"L],
      type: userMsg,
      translationNote: "Add command name"L,
      translatable: TRUE,
      id: 12],
    remove: [
      msgKey: VPMMessages.Key.remove.ORD,
      msg: XString.FromSTRING["Remove"L],
      type: pSheetItem,
      translationNote: "Remove command name"L,
      translatable: TRUE,
      id: 13],
    which: [
      msgKey: VPMMessages.Key.which.ORD,
      msg: XString.FromSTRING["Which"L],
      type: pSheetItem,
      translationNote: "Choice item tag for which to be added or removed"L,
      translatable: TRUE,
      id: 14],
    whichChoices: [
      msgKey: VPMMessages.Key.whichChoices.ORD,
      msg: XString.FromSTRING["Members:0@Friends:1@Owners:2"L],
      type: pSheetItem,
      translationNote: "choices for which choice"L,
      translatable: TRUE,
      id: 15],
    set: [
      msgKey: VPMMessages.Key.set.ORD,
      msg: XString.FromSTRING["Set"L],
      type: pSheetItem,
      translationNote: "Set command name"L,
      translatable: TRUE,
      id: 16],
    remark: [
      msgKey: VPMMessages.Key.remark.ORD,
      msg: XString.FromSTRING["Remark"L],
      type: pSheetItem,
      translationNote: "Remark text item tag"L,
      translatable: TRUE,
      id: 17],
    individual: [
      msgKey: VPMMessages.Key.individual.ORD,
      msg: XString.FromSTRING["Individual"L],
      type: pSheetItem,
```

```
      translationNote: "Individual text item tag"L,
      translatable: TRUE,
      id: 18],
    password: [
      msgKey: VPMMessages.Key.password.ORD,
      msg: XString.FromSTRING["Password"L],
      type: pSheetItem,
      translationNote: "Password text item tag"L,
      translatable: TRUE,
      id: 19],
    busyTryAgain: [
      msgKey: VPMMessages.Key.busyTryAgain.ORD,
      msg: XString.FromSTRING["\NThis window is busy. Try again when the current command finishes"L],
      type: errorMsg,
      translationNote: "starts with newline character"L,
      translatable: TRUE,
      id: 20],
    summaryIntro: [
      msgKey: VPMMessages.Key.summaryIntro.ORD,
      msg: XString.FromSTRING["\nSummary of <1>:<2>:<3>..."L],
      type: userMsg,
      translationNote: "first part of summary command. Begins with newline character."L,
      translatable: TRUE,
      id: 21],
    userRemark: [
      msgKey: VPMMessages.Key.userRemark.ORD,
      msg: XString.FromSTRING["\nUser remark: "L],
      type: userMsg,
      translationNote: "User remark field. Remark contents follows. Begins with newline character."L,
      translatable: TRUE,
      id: 22],
    fileService: [
      msgKey: VPMMessages.Key.fileService.ORD,
      msg: XString.FromSTRING["\nFile service: <1>:<2>:<3>"L],
      type: userMsg,
      translationNote: "File service field. Begins with newline character"L,
      translatable: TRUE,
      id: 23],
    deserializedFailed: [
      msgKey: VPMMessages.Key.deserializedFailed.ORD,
      msg: XString.FromSTRING["\nDeserialize failed"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 24],
    groupRemark: [
      msgKey: VPMMessages.Key.groupRemark.ORD,
      msg: XString.FromSTRING["\nGroup remark: "L],
      type: userMsg,
      translationNote: "Group remark field. Remark contents follows. Begins with newline character"L,
      translatable: TRUE,
      id: 25],
    numberOfMembers: [
      msgKey: VPMMessages.Key.numberOfMembers.ORD,
      msg: XString.FromSTRING["\nNumber of members: "L],
      type: userMsg,
      translationNote: "Number of members field. Numeric value follows. Begins with newline
      character"L,
      translatable: TRUE,
      id: 26],
    authProblem: [
      msgKey: VPMMessages.Key.authProblem.ORD,
      msg: XString.FromSTRING["\nProblem with authentication service, unable to finish command"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 27],
    fillInUser: [
      msgKey: VPMMessages.Key.fillInUser.ORD,
      msg: XString.FromSTRING["\nIndividual text item must be filled in"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 28],
    fillInGroup: [
      msgKey: VPMMessages.Key.fillInGroup.ORD,
```

```
      msg: XString.FromSTRING["\nGroup text item must be filled in"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 29],
    mustLoginIn: [
      msgKey: VPMMessages.Key.mustLoginIn.ORD,
      msg: XString.FromSTRING["\nYou must be logged in to use VP Maintain"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 30],
    noUserMatched: [
      msgKey: VPMMessages.Key.noUserMatched.ORD,
      msg: XString.FromSTRING["\nNo user matched"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 31],
    noGroupMatched: [
      msgKey: VPMMessages.Key.noGroupMatched.ORD,
      msg: XString.FromSTRING["\nNo group matched"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 32],
    done: [
      msgKey: VPMMessages.Key.done.ORD,
      msg: XString.FromSTRING["\NDone\N"L],
      type: userMsg,
      translationNote: "Termination message for commands. Begins and ends with new line character"L,
      translatable: TRUE,
      id: 33],
    invalidName: [
      msgKey: VPMMessages.Key.invalidName.ORD,
      msg: XString.FromSTRING["\N<> is an invalid name"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 34],
    notAUser: [
      msgKey: VPMMessages.Key.notAUser.ORD,
      msg: XString.FromSTRING["\N<1>:<2>:<3> is not a user"L],
      type: errorMsg,
      translationNote: "Templates are for local, domain and organization"L,
      translatable: TRUE,
      id: 35],
    notAGroup: [
      msgKey: VPMMessages.Key.notAGroup.ORD,
      msg: XString.FromSTRING["\N<1>:<2>:<3> is not a group"L],
      type: errorMsg,
      translationNote: "Templates are for local, domain and organization"L,
      translatable: TRUE,
      id: 36],
    noUserMatches: [
      msgKey: VPMMessages.Key.noUserMatches.ORD,
      msg: XString.FromSTRING["\NNo user matches <1>:<2>:<3>"L],
      type: errorMsg,
      translationNote: "Templates are for local, domain and organization"L,
      translatable: TRUE,
      id: 37],
    noGroupMatches: [
      msgKey: VPMMessages.Key.noGroupMatches.ORD,
      msg: XString.FromSTRING["\NNo group matches <1>:<2>:<3>"L],
      type: errorMsg,
      translationNote: "Templates are for local, domain and organization"L,
      translatable: TRUE,
      id: 38],
    listSeparator: [
      msgKey: VPMMessages.Key.listSeparator.ORD,
      msg: XString.FromSTRING[", "L],
      type: userMsg,
      translationNote: "separator between names in list of names"L,
      translatable: TRUE,
      id: 39],
    membersIntro: [
```

```
      msgKey: VPMMessages.Key.membersIntro.ORD,
      msg: XString.FromSTRING["\NMembers of <1>:<2>:<3>..."L],
      type: userMsg,
      translationNote: "first part of members command. Begins with newline character."L,
      translatable: TRUE,
      id: 40],
    membersHeader: [
      msgKey: VPMMessages.Key.membersHeader.ORD,
      msg: XString.FromSTRING["\NMembers: "L],
      type: userMsg,
      translationNote: "first part of members list. Begins with newline character. list of three part
      names follows"L,
      translatable: TRUE,
      id: 41],
    none: [
      msgKey: VPMMessages.Key.none.ORD,
      msg: XString.FromSTRING["none"L],
      type: userMsg,
      translationNote: "Templates are for local, domain and organization"L,
      translatable: TRUE,
      id: 42],
    copyLog: [
      msgKey: VPMMessages.Key.copyLog.ORD,
      msg: XString.FromSTRING["Copy Log"L],
      type: menuItem,
      translationNote: ""L,
      translatable: TRUE,
      id: 43],
    matchesIntro: [
      msgKey: VPMMessages.Key.matchesIntro.ORD,
      msg: XString.FromSTRING["\NNames matching <>..."L],
      type: userMsg,
      translationNote: "first part of matches list. Begins with newline character. list of three part
      names follows"L,
      translatable: TRUE,
      id: 44],
    aliasIntro: [
      msgKey: VPMMessages.Key.aliasIntro.ORD,
      msg: XString.FromSTRING["\NAliases for <>..."L],
      type: userMsg,
      translationNote: "first part of alias command. Begins with newline character."L,
      translatable: TRUE,
      id: 45],
    distinguishedNameHeader: [
      msgKey: VPMMessages.Key.distinguishedNameHeader.ORD,
      msg: XString.FromSTRING["\NDistinguished Name: <1>:<2>:<3>"L],
      type: userMsg,
      translationNote: "Distinguished name header. Begins with newline character."L,
      translatable: TRUE,
      id: 46],
    aliasHeader: [
      msgKey: VPMMessages.Key.aliasHeader.ORD,
      msg: XString.FromSTRING["\NAliases: "L],
      type: userMsg,
      translationNote: "first part of alias list. Begins with newline character. list of three part
      names follows"L,
      translatable: TRUE,
      id: 47],
    addSelfIntro: [
      msgKey: VPMMessages.Key.addSelfIntro.ORD,
      msg: XString.FromSTRING["\NAdding self to <1>:<2>:<3>..."L],
      type: userMsg,
      translationNote: "intro to add self command. Begins with newline character."L,
      translatable: TRUE,
      id: 48],
    alreadyMember: [
      msgKey: VPMMessages.Key.alreadyMember.ORD,
      msg: XString.FromSTRING["already a member"L],
      type: userMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 49],
    ok: [
      msgKey: VPMMessages.Key.ok.ORD,
      msg: XString.FromSTRING["ok"L],
      type: userMsg,
```

```
    translationNote: ""L,
    translatable: TRUE,
    id: 50],
  dataBaseFull: [
    msgKey: VPMMessages.Key.dataBaseFull.ORD,
    msg: XString.FromSTRING["\nClearinghouse database full. Contact system administrator"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 51],
  removeSelfIntro: [
    msgKey: VPMMessages.Key.removeSelfIntro.ORD,
    msg: XString.FromSTRING["\NRemoving self from <1>:<2>:<3>..."L],
    type: userMsg,
    translationNote: "intro to remove self command. Begins with newline character"L,
    translatable: TRUE,
    id: 52],
  notAMember: [
    msgKey: VPMMessages.Key.notAMember.ORD,
    msg: XString.FromSTRING["not a member"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 74],
  threePartName: [
    msgKey: VPMMessages.Key.threePartName.ORD,
    msg: XString.FromSTRING["<1>:<2>:<3>"L],
    type: userMsg,
    translationNote: "templates are local, domain, organization"L,
    translatable: TRUE,
    id: 53],
  resetLog: [
    msgKey: VPMMessages.Key.resetLog.ORD,
    msg: XString.FromSTRING["Reset Log"L],
    type: menuItem,
    translationNote: ""L,
    translatable: TRUE,
    id: 54],
  dropLog: [
    msgKey: VPMMessages.Key.dropLog.ORD,
    msg: XString.FromSTRING["\NSelect destination for log file"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 55],
  setRemarkIntro: [
    msgKey: VPMMessages.Key.setRemarkIntro.ORD,
    msg: XString.FromSTRING["\NSetting remark for <1>:<2>:<3>..."L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 56],
  setPasswordIntro: [
    msgKey: VPMMessages.Key.setPasswordIntro.ORD,
    msg: XString.FromSTRING["\NSetting password for <1>:<2>:<3>..."L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 57],
  logFileName: [
    msgKey: VPMMessages.Key.logFileName.ORD,
    msg: XString.FromSTRING["VPMaintain.log"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 59],
  notEnoughMemory: [
    msgKey: VPMMessages.Key.notEnoughMemory.ORD,
    msg: XString.FromSTRING["\NOperation failed. Not enough program memory"L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 60],
  notEnoughDisk: [
    msgKey: VPMMessages.Key.notEnoughDisk.ORD,
    msg: XString.FromSTRING["\NOperation failed. Not enough disk pages available"L],
```

```
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 61],
  programBug: [
    msgKey: VPMMessages.Key.programBug.ORD,
    msg: XString.FromSTRING["\NOperation failed due to program error"L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 62],
  addingMember: [
    msgKey: VPMMessages.Key.addingMember.ORD,
    msg: XString.FromSTRING["\NAdding <1>:<2>:<3> as member of group <4>:<5>:<6>..."L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 63],
  addingFriend: [
    msgKey: VPMMessages.Key.addingFriend.ORD,
    msg: XString.FromSTRING["\NAdding <1>:<2>:<3> as friend of group <4>:<5>:<6>..."L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 64],
  addingOwner: [
    msgKey: VPMMessages.Key.addingOwner.ORD,
    msg: XString.FromSTRING["\NAdding <1>:<2>:<3> as owner of group <4>:<5>:<6>..."L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 65],
  alreadyFriend: [
    msgKey: VPMMessages.Key.alreadyFriend.ORD,
    msg: XString.FromSTRING["already a friend"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 66],
  alreadyOwner: [
    msgKey: VPMMessages.Key.alreadyOwner.ORD,
    msg: XString.FromSTRING["already a owner"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 67],
  removingMember: [
    msgKey: VPMMessages.Key.removingMember.ORD,
    msg: XString.FromSTRING["\NRemoving <1>:<2>:<3> as member of group <4>:<5>:<6>..."L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 68],
  removingFriend: [
    msgKey: VPMMessages.Key.removingFriend.ORD,
    msg: XString.FromSTRING["\NRemoving <1>:<2>:<3> as friend of group <4>:<5>:<6>..."L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 69],
  removingOwner: [
    msgKey: VPMMessages.Key.removingFriend.ORD,
    msg: XString.FromSTRING["\NRemoving <1>:<2>:<3> as owner of group <4>:<5>:<6>..."L],
    type: errorMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 70],
  notAFriend: [
    msgKey: VPMMessages.Key.notAFriend.ORD,
    msg: XString.FromSTRING["not a friend"L],
    type: userMsg,
    translationNote: ""L,
    translatable: TRUE,
    id: 71],
  notAOwner: [
    msgKey: VPMMessages.Key.notAOwner.ORD,
```

```
      msg: XString.FromSTRING["not a owner"L],
      type: userMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 72],
    fillInNameList: [
      msgKey: VPMMessages.Key.fillInNameList.ORD,
      msg: XString.FromSTRING["\nName List text item must be filled in"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 73],
    authError: [
      msgKey: VPMMessages.Key.authError.ORD,
      msg: XString.FromSTRING["\nOperation failed due to authentication service error"L],
      type: errorMsg,
      translationNote: ""L,
      translatable: TRUE,
      id: 75],
    friendsHeader: [
      msgKey: VPMMessages.Key.friendsHeader.ORD,
      msg: XString.FromSTRING["\nFriends: "L],
      type: userMsg,
      translationNote: "Friends of list header. Begins with newline character. list of three part
      names follows"L,
      translatable: TRUE,
      id: 76],
    ownersHeader: [
      msgKey: VPMMessages.Key.ownersHeader.ORD,
      msg: XString.FromSTRING["\nOwners: "L],
      type: userMsg,
      translationNote: "Owners of list header. Begins with newline character. list of three part
      names follows"L,
      translatable: TRUE,
      id: 77],
    summaryStart: [
      msgKey: VPMMessages.Key.summaryStart.ORD,
      msg: XString.FromSTRING["\nSummary of <> ..."L],
      type: userMsg,
      translationNote: "Beginning of summary command. Begins with newline character. Arg is group or
      name being summarized"L,
      translatable: TRUE,
      id: 78],
    membersStart: [
      msgKey: VPMMessages.Key.membersStart.ORD,
      msg: XString.FromSTRING["\nMembers of <> ..."L],
      type: userMsg,
      translationNote: "Beginning of members command. Begins with newline character. Arg is group(s)
      with members"L,
      translatable: TRUE,
      id: 79],
    addSelfStart: [
      msgKey: VPMMessages.Key.addSelfStart.ORD,
      msg: XString.FromSTRING["\nAdding self to <> ..."L],
      type: userMsg,
      translationNote: "Beginning of add self command. Begins with newline character. Arg is group(s)
      adding to"L,
      translatable: TRUE,
      id: 80],
    removeSelfStart: [
      msgKey: VPMMessages.Key.removeSelfStart.ORD,
      msg: XString.FromSTRING["\nRemoving self from <> ..."L],
      type: userMsg,
      translationNote: "Beginning of remove self command. Begins with newline character. Arg is
      group(s) removing from"L,
      translatable: TRUE,
      id: 81],
    setRemarkStart: [
      msgKey: VPMMessages.Key.setRemarkStart.ORD,
      msg: XString.FromSTRING["\nSetting remark to <> ..."L],
      type: userMsg,
      translationNote: "Beginning of set remark command. Begins with newline character. Arg is
      remark"L,
      translatable: TRUE,
      id: 82],
    setPasswordStart: [
```

```
        msgKey: VPMMessages.Key.setPasswordStart.ORD,
        msg: XString.FromSTRING["\nSetting password..."L],
        type: userMsg,
        translationNote: "Beginning of set Password command. Begins with newline character"L,
        translatable: TRUE,
        id: 83],
      addNamesStart: [
        msgKey: VPMMessages.Key.addNamesStart.ORD,
        msg: XString.FromSTRING["\nAdding names to <>..."L],
        type: userMsg,
        translationNote: "Beginning of add names command. Begins with newline character. Arg is
        group(s) adding to"L,
        translatable: TRUE,
        id: 84],
      removeNamesStart: [
        msgKey: VPMMessages.Key.removeNamesStart.ORD,
        msg: XString.FromSTRING["\nRemoving names from <>..."L],
        type: userMsg,
        translationNote: "Beginning of remove names command. Begins with newline character. Arg is
        group(s) removing from"L,
        translatable: TRUE,
        id: 85],
      errorOccured: [
        msgKey: VPMMessages.Key.errorOccured.ORD,
        msg: XString.FromSTRING["\n\NError occured. Check message area for details\N"L],
        type: userMsg,
        translationNote: ""L,
        translatable: TRUE,
        id: 86],
      inUse: [
        msgKey: VPMMessages.Key.inUse.ORD,
        msg: XString.FromSTRING["In Use"L],
        type: pSheetItem,
        translationNote: "Text for inuse indicator"L,
        translatable: TRUE,
        id: 87],
      accessRightsInsufficient: [
        msgKey: VPMMessages.Key.accessRightsInsufficient.ORD,
        msg: XString.FromSTRING["Access Rights Insufficient"L],
        type: userMsg,
        translationNote: "insufficient access to change passwd"L,
        translatable: TRUE,
        id: 88]
      <<,
      : [
        msgKey: VPMMessages.Key.USEAGAINTOREPLACETHISSTRING.ORD,
        msg: XString.FromSTRING[""L],
        type: ,
        translationNote: ""L,
        translatable: TRUE,
        id: ],
>>
      ];

    h ← XMessage.AllocateMessages["VP Maintain"L, VPMMessages.Key.LAST.ORD.SUCC, NIL, NIL];
    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};

InitFromArray[];

END....

LOG.
```

```
-- File: VPMPrivate.mesa - Last edit:
-- ANg:OSBU North:Xerox 12-Apr-88 14:40:31
-- JGS                 17-Apr-86 10:11:55

-- Copyright (C) 1985, 1986, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY

  NSFile USING [Handle, Reference, nullHandle , nullReference],
  NSFileStream USING [Handle],
  VPMMessages USING [Key],
  Window USING [Handle],
  XFormat USING [Handle, Object],
  XString USING [nullReaderBody, Reader, ReaderBody];

VPMPrivate: DEFINITIONS = BEGIN OPEN XS: XString;

  -- TYPEs

  Data: TYPE = LONG POINTER TO DataObject;

  DataObject: TYPE = RECORD [
    busy, done, inverted: BOOLEAN ← FALSE,
    logXFH, msgXFH: XFormat.Handle ← NIL,
    body, logW, msgW, inuseW: Window.Handle ← NIL,
    cond: CONDITION,
    zone: UNCOUNTED ZONE ← NIL,
    rb: XString.ReaderBody ← XS.nullReaderBody,
    logStream: NSFileStream.Handle ← [NIL],
    logXFO, msgXFO, logWXFO, logSXFO: XFormat.Object ← [NIL],
    file: NSFile.Handle ← NSFile.nullHandle,
    fileRef: NSFile.Reference ← NSFile.nullReference,
    promptMode: BOOLEAN ← TRUE,
    keepLog: BOOLEAN ←TRUE,
    scratchPadZone: UNCOUNTED ZONE ← NIL];

  Busy: PROC [data: Data] RETURNS [BOOLEAN];
  NotBusy: PROC [data: Data];

  Which: TYPE = {member, friends, owners};

  UserOrGroup: TYPE = {user, group};

  Summary: PROC [data: Data, name: XS.ReaderBody, ug: UserOrGroup];
  Matches: PROC [data: Data, name: XS.ReaderBody, ug: UserOrGroup];
  Members: PROC [data: Data, name: XS.ReaderBody];
  Aliases: PROC [data: Data, name: XS.ReaderBody, ug: UserOrGroup];
  AddSelf: PROC [data: Data, name: XS.ReaderBody];
  RemoveSelf: PROC [data: Data, name: XS.ReaderBody];
  AddName: PROC [data: Data, group, nameList: XS.ReaderBody, which: Which];
  RemoveName: PROC [
    data: Data, group, nameList: XS.ReaderBody, which: Which];
  SetRemark: PROC [data: Data, group, remark: XS.ReaderBody];
  SetPassword: PROC [data: Data, individual, password: XS.ReaderBody];

  Error: PROC [data: Data, key: VPMMessages.Key];
  OneToLog: PROC [data: Data, key: VPMMessages.Key, r: XS.Reader];

  END...
```

```
-- File: VPMShellImpl.mesa - Last edit:
-- ANg:OSBU North:Xerox 15-Jun-88 16:30:42
-- JGS                 17-Apr-86 11:07:49

-- Copyright (C) 1985, 1986, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, GetProp, MakeAtom, null, RefPair],
  Catalog,
  Containee USING [
    Data, DataHandle, Error, GenericProc, GetImplementation, Implementation,
    SetImplementation],
  Context USING [Create, Data, Find, Type, UniqueType],
  Cursor USING [StoreCharacter,GetInfo, Set, Type],
  Divider,
  Directory,
  FormWindow USING [
    AppendItem, AppendLine, ChoiceChangeProc, ChoiceItems, CommandProc, Create,
    GetChoiceItemValue, GetTextItemValue, LayoutProc, Line, MakeChoiceItem,
    MakeCommandItem, MakeItemsProc, MakeTextItem, MakeWindowItem, noTabStop,
    SetTabStops, SetVisibility, TabStops, Visibility],
  FormWindowMessageParse USING [FreeChoiceItems, ParseChoiceItemMessage],
  Heap USING [Expand, Create],
  LogWindow,
  MenuData USING [CreateItem, CreateMenu, ItemHandle, MenuHandle, MenuProc],
  MessageWindow USING [Clear, Create, XFormatObject],
  NSAssignedTypes USING [tText],
  NSFile,
  NSFileStream,
  NSString USING [FreeString, String],
  Selection USING [
    ActOnProc, ConvertProc, Error, FreeContext, nullValue,  ValueCopyMoveProc,
    ValueFreeProc, ValueProcs],
  SimpleTextDisplay USING [MeasureString],
  SimpleTextFont USING [
    FontNotFound, MappedFont, MappedFontHandle, MappedDefaultFont],
  StarWindowShell USING [
    Create, CreateBody, GetBody, GetZone, Handle, IsCloseLegalProc,     SetRegularCommands],
  StarFileTypes,
  StarWindowShellExtra2 USING [SetPreferredInteriorDims],
  Stream,
  TIP USING [NotifyProc],
  TIPStar USING [SetMode],
  VPMMessages USING [Handle, Key],
  VPMPrivate USING [
    AddName, AddSelf, Aliases, Data, DataObject, Matches, Members, RemoveName,
    RemoveSelf, SetPassword, SetRemark, Summary, Which],
  Window USING [Dims, Handle],
  XFormat USING [FormatProc, Handle, ReaderBody],
  XMessage USING [Get, Handle],
  XString USING [
    Context, NSStringFromReader, Reader, ReaderBody, FromSTRING];


VPMShellImpl: MONITOR
  IMPORTS
    Atom, Catalog, Containee, Context, Cursor, Divider, Directory, FormWindow,
    Heap, FormWindowMessageParse, LogWindow, MenuData, MessageWindow, NSFile,
    NSFileStream, NSString, Selection, SimpleTextDisplay, SimpleTextFont,
    StarWindowShell, StarWindowShellExtra2, Stream,  TIPStar, VPMMessages, TIP,
    VPMPrivate, XFormat, XMessage, XString
  EXPORTS VPMPrivate =
  BEGIN OPEN FW: FormWindow, XS: XString;

  -- TYPEs

  Data: TYPE = VPMPrivate.Data;
  NewMaintainObject: TYPE = RECORD [
    maintainFile: NSFile.Handle, zone: UNCOUNTED ZONE];
  NewMaintainData: TYPE = LONG POINTER TO NewMaintainObject;

  << This code orignally came from CalculatorOps.   >>
  -- Constants


  bodyWindowDims: Window.Dims = [500, 700];
```

```
  maintainIconFileType: NSFile.Type = 4431;

  -- Data

  context: Context.Type ← Context.UniqueType[];
  maintainZone: UNCOUNTED ZONE ← Heap.Create[initial: 1, increment: 1];
  xmh: XMessage.Handle = VPMMessages.Handle[];
  newImpl, oldImpl: Containee.Implementation;
  dividerData: Containee.Data ← [Divider.notAFile];
  newSelectionValueProcs: LONG POINTER TO Selection.ValueProcs = maintainZone.NEW[
    Selection.ValueProcs];

  -- Procedures

  IsCloseLegal: ENTRY StarWindowShell.IsCloseLegalProc =
    BEGIN
    ENABLE UNWIND => NULL;
    data: Data = GetContext[StarWindowShell.GetBody[sws]];
    IF data = NIL OR ~data.busy THEN
      BEGIN
      fh: NSFile.Handle ← NSFile.OpenByReference[data.fileRef];
      fileStream: NSFileStream.Handle ← NSFileStream.Create[
        file: fh, options: [signalEndOfStream: TRUE]];

      LogWindow.SaveLog[fileStream, data.logW];
      Stream.SendNow[fileStream];
      NSFileStream.SetLength[fileStream, Stream.GetPosition[fileStream]];
      Stream.Delete[fileStream];

      RETURN[TRUE];
      END;
    data.msgXFH.ReaderBody[GetMessageRB[busyTryAgain]];
    RETURN[FALSE];
    END;

  DestroyContext: PROC [data: Data, window: Window.Handle] = {
   << data.logStream.Delete[]>>};

  GetContext: PROC [body: Window.Handle] RETURNS [data: Data] = {
    data ← Context.Find[context, body];
    IF data = NIL THEN ERROR;  -- just in case.
    RETURN[data];
    };

  Init: PROC =
    BEGIN
    name: XString.ReaderBody ← GetMessageRB[applicationName];

    -- First, get any existing implementation for this icon
    oldImpl ← newImpl ← Containee.GetImplementation[maintainIconFileType];
    newImpl.genericProc ← MaintainGenericProc;
    newImpl.convertProc ← Convert;
    [] ← Containee.SetImplementation[maintainIconFileType, newImpl];

    Divider.AddEntry[
      handle: Directory.GetDividerHandle[officeAids], type: maintainIconFileType,
      label: @name, data: @dividerData, convertProc: DividerConvertProc];

    <<Attention.AddMenuItem [
      MenuData.CreateItem [zone: NIL, name: @name, proc: MenuProc] ];>>
    END;  -- of Init


  MaintainGenericProc: Containee.GenericProc =
    BEGIN
    SELECT atom FROM
      open => {
        IF data.reference = Divider.notAFile THEN {
          msg: XString.ReaderBody ← XS.FromSTRING["Icon can not be open"L];
          Containee.Error[@msg];
          };

        IF data.reference.service.systemElement #
        NSFile.localSystemElement THEN {
          msg: XString.ReaderBody ← XS.FromSTRING["Icon can not be open"L];
          Containee.Error [@msg]};
```

```
        RETURN[OpenTheShell[data]];
        }
      ENDCASE =>
        RETURN oldImpl.genericProc[atom, data, changeProc, changeProcData];

    -- Note that I call the old implementation for any atoms that I don't want to handle. In most
    cases, the old implementation will be the default implementation supplied by ContaineeImpl, which
    will probably display an appropriate message to the user.
    END;  -- of MaintainGenericProc



  OpenTheShell: PROC [uniqueFileData: Containee.DataHandle]
    RETURNS [shell: StarWindowShell.Handle] = {
      resetLog: XString.ReaderBody ← GetMessageRB[resetLog];
    name: XString.ReaderBody ← GetMessageRB[applicationName];
    data: Data;
    z: UNCOUNTED ZONE;
    items: ARRAY [0..1) OF MenuData.ItemHandle;
    myMenu: MenuData.MenuHandle;
    body: Window.Handle;



    -- Create the shell that we will be using and the body inside it.
    shell ← StarWindowShell.Create[name: @name, isCloseLegalProc: IsCloseLegal];
    body ← StarWindowShell.CreateBody[sws: shell, box: [[0, 0], bodyWindowDims]];

    -- Get the zone/heap where we will be storing our data
    -- We know it should be at least 15 pages, so expand to that size.
    z ← StarWindowShell.GetZone[shell];
    Heap.Expand[z, 15];

    -- Now create the menu items which will appear on the shell.
    items ← [
      MenuData.CreateItem[zone: z, name: @resetLog, proc: ResetLogProc]
      ];

    -- Now attach the menus to the shell.
    myMenu ← MenuData.CreateMenu[zone: z, title: NIL, array: DESCRIPTOR[items]];
    StarWindowShell.SetRegularCommands[sws: shell, commands: myMenu];


    -- Now allocate the data we need for this particular window and associate it with this window.
    data ← z.NEW[VPMPrivate.DataObject ← [body: body, zone: z, done: TRUE]];
    -- Allocate data and "hang it off the body window" by using Context.Create.
    <<data.logStream ← GetLogStream[data];>>
    data.logW ← NIL;
    data.keepLog ← TRUE;
    data.fileRef ← uniqueFileData.reference;
    data.logXFH ← @data.logXFO;
    data.msgXFH ← @data.msgXFO;
    <<data.logSXFO ← XFormat.StreamObject[data.logStream];>>
    data.logXFO ← [proc: LogFormatProc, data: data];
    Context.Create[type: context, data: data, proc: DestroyContext, window: body];

    -- Now create the window inside the shell where we will layout our commands and log windows.
    FW.Create[
      window: body, makeItemsProc: MakeItems, layoutProc: DoLayout, zone: z,
      clientData: data];

    -- Tell the system how big we want our shell to be
    StarWindowShellExtra2.SetPreferredInteriorDims[shell, bodyWindowDims];

    -- Now return the shell so it can be made visible.
    RETURN[shell]};

  <<GetLogStream: PROC [data: Data] RETURNS [stream: NSFileStream.Handle] = {
    file: NSFile.Handle;
    rb: XS.ReaderBody ← GetMessageRB[logFileName];
    name: NSString.String = XS.NSStringFromReader[z: data.zone, r: @rb];
    attributes: ARRAY [0..3) OF NSFile.Attribute ← [
      [name[name]], [sizeInBytes[4096]], [type[NSAssignedTypes.tText]]];
    file ← NSFile.Create[
      directory: NSFile.nullHandle, attributes: DESCRIPTOR[attributes] !
```

```
        UNWIND => NSString.FreeString[data.zone, name]];
      NSString.FreeString[data.zone, name];
      RETURN NSFileStream.Create[file]};
  >>

    InitLog: PROC[data:Data] =
    BEGIN
      --open file pointed by data.fileref
      -- if length of file non-zero then copy file to log.
       size: LONG CARDINAL;
       attributes: NSFile.AttributesRecord;
       fileStream: NSFileStream.Handle;
       fh: NSFile.Handle ← NSFile.OpenByReference [data.fileRef];


       NSFile.GetAttributes[
        file: fh,
        selections: [
        [sizeInBytes: TRUE]],
        attributes: @attributes];

        size ← attributes.sizeInBytes;

        NSFile.ClearAttributes[@attributes];

       IF size = 0 THEN
        BEGIN
          NSFile.Close [fh !NSFile.Error => CONTINUE];
          RETURN;
        END;

       fileStream ← NSFileStream.Create [file: fh, options: [signalEndOfStream: TRUE]];

        [] ← LogWindow.BuildLogWindowFromFile [fileStream, data.logW, data.zone];

       Stream.Delete [fileStream];
       END;


    LogFormatProc: XFormat.FormatProc =
      BEGIN
      data: Data = h.data;

      LogWindow.Append[data.logW, r];

      --XFormat.Reader[h: @data.logWXFO, r: r];
      -- XFormat.Reader[h: @data.logSXFO, r: r];
      END;

    GetMessageRB: PROC [key: VPMMessages.Key] RETURNS [rb: XS.ReaderBody] = {
      RETURN[xmh.Get[key.ORD]]};


    SetCursor: TIP.NotifyProc = {
      Cursor.StoreCharacter[
        Containee.GetImplementation[NSAssignedTypes.tText].smallPictureProc[
        type: NSAssignedTypes.tText, normalOrReference: normal]]};

    ResetLogProc: MenuData.MenuProc = {
      body: Window.Handle = StarWindowShell.GetBody[sws: [window]];
      data: Data ← GetContext[body];

      IF Busy[data] THEN {
        data.msgXFH.ReaderBody[GetMessageRB[busyTryAgain]]; RETURN};
      BEGIN
      ENABLE UNWIND => NotBusy[data];
      MessageWindow.Clear[data.logW];

      -- use logWindow impl to clear window.
      LogWindow.ClearLog[data.logW]

      <<data.logStream.SetLength[0];
      data.logXFO.context ← XS.vanillaContext;>>

      END;  -- ELBANE
      NotBusy[data]};
```

```
  ActOn: Selection.ActOnProc = {
    SELECT action FROM
      clear => {[] ← TIPStar.SetMode[normal]; cleared ← TRUE};
      save => [] ← TIPStar.SetMode[normal];
      restore => [] ← TIPStar.SetMode[move];
      ENDCASE};

  DividerConvertProc: Divider.ConvertProc =
    <<data: LONG POINTER, PATTERN: CH.Pattern, target: Selection.Target, zone: UNCONUTED ZONE, info:
    Selection.ConversionInfo ← [convert[]]]>>
    BEGIN
    cd: Containee.DataHandle ← data;

    <<The DividerConvertProc is used to handle copying/moving/deleting inside a Divider.   If such
    activity was done for the desktop, it would use Selection.ConvertProc.

>>
    WITH i: info SELECT FROM

      << The first time this procedure gets call is to query if the transformtion that needs to be done
      is possible. It assumes you could of selected multiple items from the divider. If you only select
      one, i.query.LENGTH =1. VPMaintain does not handle multiple items, it returns "impossible" for
      i.query[c].enumeration.
>>
      query =>
        FOR c: CARDINAL IN [0..i.query.LENGTH) DO
          i.query[c].difficulty ←
            IF i.query[c].enumeration THEN impossible
            ELSE  -- not an enumeration
              SELECT i.query[c].target FROM file => easy, ENDCASE => impossible;
          ENDLOOP;

      enumeration => RETURN[Selection.nullValue];

      convert =>
        SELECT target FROM
          file =>
            BEGIN

            -- create temporary file catalogs to store in catalog.
            tempFileCatalog: NSFile.Type ← StarFileTypes.tempFileCatalog;
            label: XString.ReaderBody ← XS.FromSTRING["VP Maintain"L];
            nsName: NSString.String ← XString.NSStringFromReader[
              @label, maintainZone];
            fileRef: LONG POINTER TO NSFile.Reference ← zone.NEW[NSFile.Reference];

            -- Now create Handle to the catalog.
            tempCatalogHandle: NSFile.Handle ← Catalog.Open[tempFileCatalog];


            -- Create Attribute list for the catalog entry
            maintainAttrList: ARRAY [0..4) OF NSFile.Attribute ← [
              [name[nsName]], [type[maintainIconFileType]], [isDirectory[FALSE]], [
              sizeInBytes[0]]];

            -- Now take the Handle and the Attribute list, and create a File Handle.
            maintainFileHandle: NSFile.Handle ← NSFile.Create[
              tempCatalogHandle, DESCRIPTOR[maintainAttrList]];

            -- Remember the file handle of the file we just
            -- created. We will need it in CopyMoveNew.
            -- Also, don't bother closing it here.
            newmaintainData: NewMaintainData ← zone.NEW[
              NewMaintainObject ← [maintainFileHandle, zone]];

            -- Now get the reference to the FileHand e
            fileRef↑ ← NSFile.GetReference[maintainFileHandle];
            NSString.FreeString[maintainZone, nsName];

            << Some where later in time, you no longer need newMaintainData. Register these two call
            back proc and clean up later>>
            newSelectionValueProcs↑ ← [FreeAndDeleteNew, CopyMoveNew];

            << reset it to the correct free and copyMove procs
            since both proc gets altered in CopyMoveNew. >>
```

```
            RETURN[
              [
                value: fileRef, ops: newSelectionValueProcs,
                context: newmaintainData]];
            END;
          ENDCASE => RETURN[Selection.nullValue];
      ENDCASE;


    END;  -- of DividerConvertProc

  << Free and delete data space that was used as temporary storage for setting up entry to Catalog.
This is used as a call back procedure. >>
  FreeAndDeleteNew: Selection.ValueFreeProc =
    BEGIN
    newMaintainData: NewMaintainData ← v.context;
    zone: UNCOUNTED ZONE ← newMaintainData.zone;

    zone.FREE[@v.value];
    NSFile.Delete[newMaintainData.maintainFile];
    zone.FREE[@newMaintainData];
    END;  -- FreeAndDeleteNew

  << Free and delete data space that was used as temporary storage for setting up entry to Catalog.
This is used as a call back procedure. >>
  FreeNew: Selection.ValueFreeProc =
    BEGIN
    newMaintainData: NewMaintainData ← v.context;
    zone: UNCOUNTED ZONE ← newMaintainData.zone;

    zone.FREE[@v.value];
    zone.FREE[@newMaintainData];
    END;  -- FreeNew


  <<This procedure shold release (or perhaps simply turn over control of) any resources that were
  allocated by the ConvertProc to produce the orginal converted value. Values passed in is from
  DividerConvertProc.
>>
  CopyMoveNew: Selection.ValueCopyMoveProc =
    BEGIN
    newMaintainData: NewMaintainData = v.context;
    dRef: LONG POINTER TO NSFile.Reference = LOOPHOLE[data];
    dest: NSFile.Handle ← NSFile.OpenByReference[dRef↑];

    SELECT op FROM
      copy =>
        BEGIN
        NSFile.Move[newMaintainData.maintainFile, dest];
        NSFile.Close[newMaintainData.maintainFile];
        NSFile.Close[dest];
        v.ops.free ← FreeNew;
        << need to change ValueFreeProc so we don't delete the fileHandle once we moved it >>
        v.ops.copyMove ← NIL
        << done as a safeguard just incase copyMove proc called more than once for
         every call to the ConvertProc>>
        END;

      move => BEGIN NSFile.Close[dest]; Selection.Error[invalidOperation]; END;

      ENDCASE => NULL;

    END;  -- CopyMoveNew


<<  This is used to obtain the value of the selection. It determines what Targets the selection can be
converted .
>>
  Convert: Selection.ConvertProc = {
    d: Data = data;
    WITH i: info SELECT FROM
      -- determines the difficulty of the conversion
      query =>
        FOR c: CARDINAL IN [0..LENGTH[i.query]) DO
          i.query[c].difficulty ←
```

```
            SELECT i.query[c].target FROM
              file, fileType,interpressMaster => easy,
              ENDCASE => impossible;
          ENDLOOP;

        -- doing the convertion for target types it supports
        convert =>
         SELECT target FROM
           fileType => RETURN[[zone.NEW[NSFile.Type ← NSAssignedTypes.tText]]];
            file =>
             RETURN[
               [
                 value: zone.NEW[
                 NSFile.Reference ← NSFile.GetReference[
                 NSFileStream.FileFromStream[d.logStream]]], ops: @copyOps,
                 context: zone.NEW[CopyContext ← [zone, d]]]];
           ENDCASE;

      enumeration =>
        <<
        -- $$$$ We don't think we need this code.
        SELECT target FROM
          fileType =>
            [] ← i.proc[[zone.NEW[NSFile.Type ← NSAssignedTypes.tText]]];
          file =>
            [] ← i.proc[[
              value: zone.NEW[NSFile.Reference ← NSFile.GetReference[
                NSFileStream.FileFromStream[d.logStream]]],
              ops: @copyOps, context: zone.NEW[CopyContext ← [zone, d]]]];
          ENDCASE; >>
        RETURN[Selection.nullValue];

      ENDCASE;
    RETURN[Selection.nullValue]};

  CopyContext: TYPE = RECORD [zone: UNCOUNTED ZONE, d: Data];
  copyOps: Selection.ValueProcs ← [CopyFree, Copy];

  CopyFree: Selection.ValueFreeProc <<[v: ValueHandle]>>  = {
    context: LONG POINTER TO CopyContext = v.context;
    context.zone.FREE[@v.value];  -- free the Refererce
    Selection.FreeContext[v, context.zone]};

  Copy: Selection.ValueCopyMoveProc <<[v, op, data]>>  = {
    ctxt: LONG POINTER TO CopyContext = v.context;
    zone: UNCOUNTED ZONE = ctxt.zone;  -- copy to outlive zone.FREE[@ctxt]
    d: Data = ctxt.d;
    destRef: LONG POINTER TO NSFile.Reference = data;
    destHandle: NSFile.Handle;
    log: NSFile.Handle = NSFileStream.FileFromStream[d.logStream];
    copy: NSFile.Handle;
    IF op = move OR destRef = NIL THEN ERROR Selection.Error[invalidOperation];
    destHandle ← NSFile.OpenByReference[destRef↑, [access: [add: TRUE]]];
    copy ← NSFile.Copy[
      log, destHandle !
      NSFile.Error =>
        WITH error SELECT FROM
          access => IF problem = fileOpen THEN CONTINUE ELSE REJECT;
          -- workaround for Filing bug: move from self to self raises error if name
          -- is very long (50+ characters); see ProductSoftware AR 14467, 85/4/17
          ENDCASE => REJECT; UNWIND => NSFile.Close[destHandle]];
    NSFile.Close[destHandle];
    LOOPHOLE[v.value, LONG POINTER TO NSFile.Reference]↑ ← NSFile.GetReference[
      copy];
    NSFile.Close[copy];
    Selection.FreeContext[v, zone];
    v.ops ← NIL;
    };

  Items: TYPE = {
    msg, level, inuse, group, gSummary, gMatches, gMembers, gAliases, gAddSelf,
    gRemoveSelf, gNameList, gAdd, gRemove, gWhich, gSet, gRemark, individual,
    iSummary, iMatches, iAliases, iSet, iPassword, log};

  Busy: PUBLIC ENTRY PROC [data: Data] RETURNS [BOOLEAN] = {
    IF data.busy THEN RETURN[TRUE]; data.busy ← TRUE; RETURN[FALSE]};
```

```
NotBusy: PUBLIC ENTRY PROC [data: Data] = {data.busy ← FALSE};

GroupCommandProc: FW.CommandProc = {

  data: Data ← GetContext[window];
  group: XString.ReaderBody;
   oldCursor : Cursor.Type ← Cursor.GetInfo[].type;
  IF Busy[data] THEN {
data.msgXFH.ReaderBody[GetMessageRB[busyTryAgain]]; RETURN};
     Cursor.Set[hourGlass];
  MessageWindow.Clear[data.msgW];
  group ← FW.GetTextItemValue[window, Items.group.ORD, data.zone];

  SELECT item FROM
    Items.gSummary.ORD =>
      VPMPrivate.Summary[data, group, group];
    Items.gMatches.ORD =>
       VPMPrivate.Matches[data, group, group];
    Items.gMembers.ORD =>
      VPMPrivate.Members[data, group];
    Items.gAliases.ORD =>
       VPMPrivate.Aliases[data, group, group];
    Items.gAddSelf.ORD =>
      VPMPrivate.AddSelf[data, group];
    Items.gRemoveSelf.ORD =>
      VPMPrivate.RemoveSelf[data, group];
    Items.gAdd.ORD => {
      nameList: XString.ReaderBody ← FW.GetTextItemValue[
        window, Items.gNameList.ORD, data.zone];
      which: VPMPrivate.Which = VAL[
        FW.GetChoiceItemValue[window, Items.gWhich.ORD]];
        VPMPrivate.AddName[data, group, nameList, which];
      };
    Items.gRemove.ORD => {
      nameList: XString.ReaderBody ← FW.GetTextItemValue[
        window, Items.gNameList.ORD, data.zone];
      which: VPMPrivate.Which = VAL[
        FW.GetChoiceItemValue[window, Items.gWhich.ORD]];
      VPMPrivate.RemoveName[data, group, nameList, which];
      };
    Items.gSet.ORD => {
      remark: XString.ReaderBody ← FW.GetTextItemValue[
        window, Items.gRemark.ORD, data.zone];
      VPMPrivate.SetRemark[data, group, remark];
      };
    ENDCASE;
    Cursor.Set[oldCursor];
    };

IndividualCommandProc: FW.CommandProc = {

  data: Data ← GetContext[window];
  individual: XString.ReaderBody;
  oldCursor: Cursor.Type ← Cursor.GetInfo[].type;
  IF Busy[data] THEN {
    data.msgXFH.ReaderBody[GetMessageRB[busyTryAgain]]; RETURN};
 Cursor.Set[hourGlass];
  MessageWindow.Clear[data.msgW];
  individual ← FW.GetTextItemValue[window, Items.individual.ORD, data.zone];
      SELECT item FROM
  Items.iSummary.ORD =>
   VPMPrivate.Summary[data, individual, user];
        Items.iMatches.ORD =>
    VPMPrivate.Matches[data, individual, user];
    Items.iAliases.ORD =>
    VPMPrivate.Aliases[data, individual, user];
    Items.iSet.ORD => {
      password: XString.ReaderBody ← FW.GetTextItemValue[
        window, Items.iPassword.ORD, data.zone];
      VPMPrivate.SetPassword[data, individual, password];
     };
    ENDCASE;
    Cursor.Set[oldCursor];
         };
```

```
    LevelChangeProc: FW.ChoiceChangeProc = {
      visibility: FW.Visibility = IF newValue = 0 THEN invisible ELSE visible;
      IF newValue = oldValue THEN RETURN;
      FW.SetVisibility[window, Items.gNameList.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.gAdd.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.gRemove.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.gWhich.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.gSet.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.gRemark.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.iSet.ORD, visibility, FALSE];
      FW.SetVisibility[window, Items.iPassword.ORD, visibility, TRUE];
      RETURN};

    MakeItems: FormWindow.MakeItemsProc = {
      data: Data = clientData;
      rb: XS.ReaderBody;
      mh: XMessage.Handle = VPMMessages.Handle[];
      zone: UNCOUNTED ZONE = data.zone;
      LocalMessage: PROC [key: VPMMessages.Key] RETURNS [XS.Reader] = {
        rb ← mh.Get[key.ORD]; RETURN[@rb]};
      data.msgW ← FormWindow.MakeWindowItem[
        window: window, myKey: Items.msg.ORD, size: [492, 60]];
      MessageWindow.Create[window: data.msgW, zone: zone, lines: CARDINAL.LAST];
      data.msgXFO ← MessageWindow.XFormatObject[data.msgW];

      BEGIN
      choices: FW.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[
        LocalMessage[levelChoices], zone];
      FormWindow.MakeChoiceItem[
        window: window, myKey: Items.level.ORD, tag: LocalMessage[level],
        values: choices, initChoice: 0, changeProc: LevelChangeProc];
      FormWindowMessageParse.FreeChoiceItems[choices, zone];
      END;

      FormWindow.MakeTextItem[
        window: window, myKey: Items.group.ORD, tag: LocalMessage[group],
        width: TextItemWidth[@rb, 0, 4]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gSummary.ORD, commandProc: GroupCommandProc,
        commandName: LocalMessage[summary]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gMatches.ORD, commandProc: GroupCommandProc,
        commandName: LocalMessage[matches]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gMembers.ORD, commandProc: GroupCommandProc,
        commandName: LocalMessage[members]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gAliases.ORD, commandProc: GroupCommandProc,
        commandName: LocalMessage[aliases]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gAddSelf.ORD, commandProc: GroupCommandProc,
        commandName: LocalMessage[addSelf]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gRemoveSelf.ORD, commandProc: GroupCommandProc,
        commandName: LocalMessage[removeSelf]];

      BEGIN
      r: XS.Reader ← LocalMessage[nameList];
      FormWindow.MakeTextItem[
        window: window, myKey: Items.gNameList.ORD, visibility: invisible, tag: r,
        width: TextItemWidth[r, 1, 0]];
      END;

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gAdd.ORD, visibility: invisible,
        commandProc: GroupCommandProc, commandName: LocalMessage[add]];

      FormWindow.MakeCommandItem[
        window: window, myKey: Items.gRemove.ORD, visibility: invisible,
        commandProc: GroupCommandProc, commandName: LocalMessage[remove]];
```

```
    BEGIN
    choices: FW.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[
      LocalMessage[whichChoices], zone];
    FormWindow.MakeChoiceItem[
      window: window, myKey: Items.gWhich.ORD, tag: LocalMessage[which],
      visibility: invisible, values: choices, initChoice: 0];
    FormWindowMessageParse.FreeChoiceItems[choices, zone];
    END;

    FormWindow.MakeCommandItem[
      window: window, myKey: Items.gSet.ORD, visibility: invisible,
      commandProc: GroupCommandProc, commandName: LocalMessage[set]];

    BEGIN
    setRb: XS.ReaderBody ← LocalMessage[set]↑;
    r: XS.Reader ← LocalMessage[remark];
    FormWindow.MakeTextItem[
      window: window, visibility: invisible, myKey: Items.gRemark.ORD, tag: r,
      width: TextItemWidth[r, 1, 8] - CommandItemWidth[@setRb]];
    END;

    BEGIN
    r: XS.Reader ← LocalMessage[individual];
    pair: Atom.RefPair = Atom.GetProp[currentUser, fullUserName];
    FormWindow.MakeTextItem[
      window: window, myKey: Items.individual.ORD, tag: r,
      width: TextItemWidth[r, 0, 4],
      initString: IF pair = NIL THEN NIL ELSE pair.value];
    END;

    FormWindow.MakeCommandItem[
      window: window, myKey: Items.iSummary.ORD,
      commandProc: IndividualCommandProc, commandName: LocalMessage[summary]];

    FormWindow.MakeCommandItem[
      window: window, myKey: Items.iMatches.ORD,
      commandProc: IndividualCommandProc, commandName: LocalMessage[matches]];

    FormWindow.MakeCommandItem[
      window: window, myKey: Items.iAliases.ORD,
      commandProc: IndividualCommandProc, commandName: LocalMessage[aliases]];

    FormWindow.MakeCommandItem[
      window: window, myKey: Items.iSet.ORD, visibility: invisible,
      commandProc: IndividualCommandProc, commandName: LocalMessage[set]];

    BEGIN
    FormWindow.MakeTextItem[
      window: window, myKey: Items.iPassword.ORD, visibility: invisible,
      tag: LocalMessage[password], width: 100, passwordFeedback: TRUE];
    END;


    << New window impl
>>
    BEGIN
    lwFont: SimpleTextFont.MappedFontHandle ← SimpleTextFont.MappedDefaultFont[];
    logWindowItem: Window.Handle ← FormWindow.MakeWindowItem[
      window: window, myKey: Items.log.ORD, size: [492, 400]];

    LogWindow.Create[
      window: logWindowItem,
      zone: data.zone,
      bodyWindowDims:,
      horizontalScrollbars:,
      verticalScrollbars:,
      font: lwFont,
      lwFull: LogWindowFull];

     data.logW ← logWindowItem;
    -- Open log stream if one exsist else create new stream.

      InitLog[data];

    data.logWXFO ← LogWindow.XFormatObject[data.logW ← logWindowItem];
```

```
    END;

    << replaced the old window with new impl.
    BEGIN
    wh: Window.Handle ← FormWindow.MakeWindowItem [
      window: window, myKey: Items.log.ORD, size: [492, 400]];
    MessageWindow.Create[window: wh, zone: zone, lines: CARDINAL.LAST];
    data.logWXFO ← MessageWindow.XFormatObject[data.logW ← wh];
    END;
>>


    };

  SetLogWindowFont: PROC =
    BEGIN
    fontName: XString.ReaderBody ← XString.FromSTRING["AStarTerminal12.NovaFont"L];
    lwFont: SimpleTextFont.MappedFontHandle ← SimpleTextFont.MappedFont[
      @fontName ! SimpleTextFont.FontNotFound => RESUME ];
    END;

  -- Set up 3 dummy procedure to satisfy calls.
  LogWindowFull: LogWindow.LogWindowFullProc = {
  message : XString.ReaderBody ← XS.FromSTRING["Storage for log window is full. Please Reset Log and
  start over."L];
   Containee.Error[@message];
  };


  tabStopInterval: CARDINAL = 32;

  DoLayout: FormWindow.LayoutProc = {
    leadingMargin: CARDINAL = 5;
    line: FormWindow.Line;
    -- set the tabs for FormWindow
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];
    FormWindow.SetTabStops[window: window, tabStops: tabChoice];
    -- Line 1
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.msg.ORD, line: line, preMargin: 4];
    line ← FormWindow.AppendLine[window, 4];
    FormWindow.AppendItem[
      window: window, item: Items.level.ORD, line: line, tabStop: 3];
    line ← FormWindow.AppendLine[window, 4];
    FormWindow.AppendItem[
      window: window, item: Items.group.ORD, line: line, preMargin: 4];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.gSummary.ORD, line: line, tabStop: 1];
    FormWindow.AppendItem[
      window: window, item: Items.gMatches.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    FormWindow.AppendItem[
      window: window, item: Items.gMembers.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    FormWindow.AppendItem[
      window: window, item: Items.gAliases.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.gAddSelf.ORD, line: line, tabStop: 1];
    FormWindow.AppendItem[
      window: window, item: Items.gRemoveSelf.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.gNameList.ORD, line: line, tabStop: 1];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.gAdd.ORD, line: line, tabStop: 1];
    FormWindow.AppendItem[
      window: window, item: Items.gRemove.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    FormWindow.AppendItem[
```

```
      window: window, item: Items.gWhich.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.gSet.ORD, line: line, tabStop: 1];
    FormWindow.AppendItem[
      window: window, item: Items.gRemark.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    line ← FormWindow.AppendLine[window, 4];
    FormWindow.AppendItem[
      window: window, item: Items.individual.ORD, line: line, tabStop: 0,
      preMargin: 4];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.iSummary.ORD, line: line, tabStop: 1];
    FormWindow.AppendItem[
      window: window, item: Items.iMatches.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    FormWindow.AppendItem[
      window: window, item: Items.iAliases.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    line ← FormWindow.AppendLine[window, 2];
    FormWindow.AppendItem[
      window: window, item: Items.iSet.ORD, line: line, tabStop: 1];
    FormWindow.AppendItem[
      window: window, item: Items.iPassword.ORD, line: line,
      tabStop: FormWindow.noTabStop, preMargin: 8];
    line ← FormWindow.AppendLine[window, 4];
    FormWindow.AppendItem[
      window: window, item: Items.log.ORD, line: line, tabStop: 0, preMargin: 4];
    };

  TextItemWidth: PROC [tag: XS.Reader, tabStop, preMargin: CARDINAL]
    RETURNS [CARDINAL] = {
    tagWidth: CARDINAL = SimpleTextDisplay.MeasureString[tag].width;
    RETURN[
      bodyWindowDims.w - tagWidth - (tabStop * tabStopInterval) - preMargin - 8]};

  CommandItemWidth: PROC [tag: XS.Reader] RETURNS [CARDINAL] = {
    tagWidth: CARDINAL = SimpleTextDisplay.MeasureString[tag].width;
    RETURN[tagWidth + 16]};

  currentUser, fullUserName, open: Atom.ATOM ← Atom.null;

  InitAtoms: PROCEDURE = {
    currentUser ← Atom.MakeAtom["CurrentUser"L];
    fullUserName ← Atom.MakeAtom["FullUserName"L];
    open ← Atom.MakeAtom["Open"L];
    };

  -- Main line code

  InitAtoms[];
  Init[];

  END.
```

```
-- NameFinderTool.mesa
--  4-Oct-89 13:59:05

DIRECTORY
  Attention,
  Context,
  FormWindow,
  Heap,
  MenuData,
  StarWindowShell,
  TIP,
  Window,
  XString;

NameFinderTool: PROGRAM
  IMPORTS Attention, Context, FormWindow, Heap, MenuData, StarWindowShell, TIP, Window, XString = {

  -- TYPEs

  Items: TYPE = {matchwords, matchcase, find, stop, help, nametofind, results};

  Data: TYPE = LONG POINTER TO DataObject;
  DataObject: TYPE = RECORD [
    -- Record structure for the tool data
    -- Fill in with the variables used by tool
  ];


  -- Constants and Data

  formWindowDims: Window.Dims ← [500, 750];
  shellDims: Window.Dims = [500, 750];                    -- display size of tool
  sampleString: XString.ReaderBody ← XString.FromSTRING["NameFinderTool"L];

  tabStopInterval: CARDINAL = 50;

  context: Context.Type ← Context.UniqueType[];

-- Procedures

  Init: PROCEDURE = {
    Attention.AddMenuItem [
      MenuData.CreateItem [
        zone: Heap.systemZone,
        name: @sampleString,
        proc: MakeShell] ];
    };

  MakeShell: MenuData.MenuProc = {
    shell: StarWindowShell.Handle = StarWindowShell.Create [
      name: @sampleString,
      scrollData: [displayHorizontal: FALSE, displayVertical: FALSE]];
    formWindow: Window.Handle ← StarWindowShell.CreateBody [
      sws: shell,
      box: [[0,0], formWindowDims] ];
    FormWindow.Create [
      window: formWindow,
      makeItemsProc: MakeItems,
      layoutProc: DoLayout];
    StarWindowShell.SetPreferredDims [shell, shellDims];
    StarWindowShell.Push [shell];
    };

  MakeItems: FormWindow.MakeItemsProc = {
    fwz: UNCOUNTED ZONE = FormWindow.GetZone[window];

    BEGIN
    rb: XString.ReaderBody ← XString.FromSTRING["Match words"L];
    FormWindow.MakeBooleanItem [
      window: window,
      myKey: Items.matchwords.ORD,
      initBoolean: TRUE,
      label: [string[rb]] ];
    END;

    BEGIN
```

```
    rb: XString.ReaderBody ← XString.FromSTRING["Match case"L];
    FormWindow.MakeBooleanItem [
      window: window,
      myKey: Items.matchcase.ORD,
      initBoolean: TRUE,
      label: [string[rb]] ];
    END;

    BEGIN
    rb: XString.ReaderBody ← XString.FromSTRING["Find"L];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: Items.find.ORD,
      commandProc: FindName,
      commandName: @rb];
    END;

    BEGIN
    rb: XString.ReaderBody ← XString.FromSTRING["Stop"L];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: Items.stop.ORD,
      commandProc: AbortFind,
      commandName: @rb];
    END;

    BEGIN
    rb: XString.ReaderBody ← XString.FromSTRING["Help"L];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: Items.help.ORD,
      commandProc: ,
      commandName: @rb];
    END;

    BEGIN
    tag: XString.ReaderBody ← XString.FromSTRING["Name to find"L];
    FormWindow.MakeTextItem [
      window: window,
      myKey: Items.nametofind.ORD,
      tag: @tag,
      width: 300 ];
    END;

    BEGIN
    tag: XString.ReaderBody ← XString.FromSTRING[""L];
    wh: Window.Handle ← FormWindow.MakeWindowItem [
      window: window,
      myKey: Items.results.ORD,
      tag: @tag,
      size: [400, 500] ];
                -- These bufferProcs must be written!
    [] ← Window.SetDisplayProc[wh, Display];
    [] ← TIP.SetNotifyProc[wh, Notify];
    END;

    };

  DoLayout: FormWindow.LayoutProc = {
    lineLeading: CARDINAL = 6;
    topMargin: CARDINAL = 16;
    line: FormWindow.Line;

        -- set the tabs for FormWindow
    tabChoice: fixed FormWindow.TabStops = [fixed[ tabStopInterval]];
    FormWindow.SetTabStops[window: window, tabStops: tabChoice];
    -- Line 1
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: topMargin];
    FormWindow.AppendItem [
      window: window,
      item: Items.matchwords.ORD,
      line: line,
      tabStop: 16 / tabStopInterval,
      preMargin: 16 MOD tabStopInterval];
```

```
    FormWindow.AppendItem [
      window: window,
      item: Items.matchcase.ORD,
      line: line,
      tabStop: 128 / tabStopInterval,
      preMargin: 128 MOD tabStopInterval];
    -- Line 2
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    FormWindow.AppendItem [
      window: window,
      item: Items.find.ORD,
      line: line,
      tabStop: 16 / tabStopInterval,
      preMargin: 16 MOD tabStopInterval];
    FormWindow.AppendItem [
      window: window,
      item: Items.stop.ORD,
      line: line,
      tabStop: 64 / tabStopInterval,
      preMargin: 64 MOD tabStopInterval];
    FormWindow.AppendItem [
      window: window,
      item: Items.help.ORD,
      line: line,
      tabStop: 120 / tabStopInterval,
      preMargin: 120 MOD tabStopInterval];
    -- Line 3
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    FormWindow.AppendItem [
      window: window,
      item: Items.nametofind.ORD,
      line: line,
      tabStop: 22 / tabStopInterval,
      preMargin: 22 MOD tabStopInterval];
    -- Line 4
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    FormWindow.AppendItem [
      window: window,
      item: Items.results.ORD,
      line: line,
      tabStop: 17 / tabStopInterval,
      preMargin: 17 MOD tabStopInterval];
    -- Line 5
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    -- Line 6
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    -- Line 7
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    -- Line 8
    line ← FormWindow.AppendLine [
      window: window,
      spaceAboveLine: lineLeading];
    };

  LocalFind: PROC [fw: Window.Handle] RETURNS [mydata: Data] = {
    mydata ← Context.Find[context, fw];
    IF mydata = NIL THEN ERROR;
    RETURN [mydata];
    };

  FindName: FormWindow.CommandProc = { };

  AbortFind: FormWindow.CommandProc = { };
```

```
    Help: FormWindow.CommandProc = { };
    Display: Window.DisplayProc = { };
    Notify: TIP.NotifyProc = { };
    -- Mainline code
    Init[];
    }...
```

```
-- PublicCommands.mesa
-- Trow  4-Oct-89 13:11:55

DIRECTORY Stream, System;

PublicCommands: DEFINITIONS
 = {

  BindHandle: TYPE = LONG POINTER TO READONLY BindObject;
  BindObject: TYPE;

  Status: TYPE = INTEGER;

  Grep: PROCEDURE [
    bH: BindHandle, matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: LONG STRING, file: LONG STRING,
    output: Stream.Handle]
    RETURNS [status: Status];

  GetEOLConvention: PROCEDURE [
    bH: BindHandle]
    RETURNS [eolConvention: LONG STRING];

  FreeGetEOLConventionResults: PROCEDURE[bH: BindHandle, eolConvention: LONG STRING];

  ServiceError: ERROR [
    bH: BindHandle, problem: ServiceProblem];

  ServiceProblem: TYPE = MACHINE DEPENDENT {cannotAuthenticate(0), serviceFull(1),
  serviceUnavailable(2), notPublic(3), (CARDINAL.LAST - 1)};

  TransferError: ERROR [
    bH: BindHandle, problem: TransferProblem];

  TransferProblem: TYPE = MACHINE DEPENDENT {aborted(0), (CARDINAL.LAST - 1)};

  RemoteBind: PROCEDURE [
    host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
    RETURNS[bH: BindHandle];

  RemoteUnbind: PROCEDURE[bH: BindHandle] RETURNS [nil: BindHandle];



  }.
```

```
-- PublicCommandsClientImpl.mesa
-- Trow  4-Oct-89 13:22:18

DIRECTORY Courier, Heap, System, PublicCommands, PublicCommandsCourier, Stream, XStream;

PublicCommandsClientImpl: PROGRAM
  IMPORTS Heap, Courier, PublicCommandsCourier, XStream
  EXPORTS PublicCommands = {

  BindHandle: TYPE = LONG POINTER TO READONLY BindObject;
  BindObject: PUBLIC TYPE = Courier.Object;

  Grep: PUBLIC PROCEDURE [
    bH: BindHandle, matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: LONG STRING, file: LONG STRING,
    output: Stream.Handle]
    RETURNS [status: PublicCommands.Status]= {
    args: PublicCommandsCourier.GrepArgs ← [matchCase, matchWords, pattern, file,
    XStream.Make[[stream[sH: output]]]];
    res: PublicCommandsCourier.GrepRes;
    DoCourierCall[
      cH: bH, procedureNumber: PublicCommandsCourier.Grep,
      arguments: [@args, PublicCommandsCourier.DescribeGrepArgs],
      results: [@res, PublicCommandsCourier.DescribeGrepRes],
      streamCheckoutProc: XStream.UserCheckout!
        UNWIND => XStream.Destroy[args.output]];
    [status] ← res;
    XStream.Destroy[args.output];
    };

  GetEOLConvention: PUBLIC PROCEDURE [
    bH: BindHandle]
    RETURNS [eolConvention: LONG STRING]= {
    res: PublicCommandsCourier.GetEOLConventionRes;
    DoCourierCall[
      cH: bH, procedureNumber: PublicCommandsCourier.GetEOLConvention,
      results: [@res, PublicCommandsCourier.DescribeGetEOLConventionRes],
      streamCheckoutProc: XStream.UserCheckout!
        UNWIND => NULL];
    [eolConvention] ← res;
    };

  FreeGetEOLConventionResults: PUBLIC PROCEDURE[
    bH: BindHandle, eolConvention: LONG STRING] = {
    res: PublicCommandsCourier.GetEOLConventionRes;
    res.eolConvention ← eolConvention;
    Courier.Free[[@res, PublicCommandsCourier.DescribeGetEOLConventionRes], bH.zone];
    };

  ServiceError: PUBLIC ERROR [
    bH: BindHandle, problem: PublicCommands.ServiceProblem]= CODE;

  TransferError: PUBLIC ERROR [
    bH: BindHandle, problem: PublicCommands.TransferProblem]= CODE;


  RemoteBind: PUBLIC PROCEDURE[
    host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
    RETURNS[bH: BindHandle] = {
    IF zone = NIL THEN zone ← Heap.systemZone;
    bH ← Courier.Create[
      remote: host, programNumber: PublicCommandsCourier.programNumber,
    versionNumber: PublicCommandsCourier.version, zone: zone, classOfService: transactional];
    };

  RemoteUnbind: PUBLIC PROCEDURE[
    bH: BindHandle] RETURNS [nil: BindHandle] = {
    nil ← NIL;
    IF bH # NIL THEN Courier.Delete[bH];
    };


  DoCourierCall: PROCEDURE[
    cH: Courier.Handle, procedureNumber: CARDINAL,
    arguments: Courier.Parameters ← Courier.nullParameters,
    results: Courier.Parameters ← Courier.nullParameters,
    streamCheckoutProc: PROCEDURE [cH: Courier.Handle] ← NIL] = {
```

```
    ENABLE {
      Courier.RemoteErrorSignalled => {
        SELECT errorNumber FROM
          100 => DoServiceError[cH, arguments];
          101 => DoTransferError[cH, arguments];
          ENDCASE;
        };
      Courier.Error => NULL;
      };
    [] ← Courier.Call[
      cH: cH, procedureNumber: procedureNumber, arguments: arguments,
      results: results, streamCheckoutProc: streamCheckoutProc];
    };

  DoServiceError: PROCEDURE[
    bH: BindHandle, arguments: Courier.Arguments] = {
    args: PublicCommandsCourier.ServiceErrorArgs;
    arguments[[@args, PublicCommandsCourier.DescribeServiceErrorArgs]];
    ERROR ServiceError[bH, args.problem];
    };

  DoTransferError: PROCEDURE[
    bH: BindHandle, arguments: Courier.Arguments] = {
    args: PublicCommandsCourier.TransferErrorArgs;
    arguments[[@args, PublicCommandsCourier.DescribeTransferErrorArgs]];
    ERROR TransferError[bH, args.problem];
    };



  }.
```

```
-- PublicCommandsCourier.mesa
-- Trow  4-Oct-89 13:12:29

DIRECTORY Courier, PublicCommands, XStream;

PublicCommandsCourier: DEFINITIONS
 = {

  programNumber: LONG CARDINAL = 2220;
  version: CARDINAL = 1;
  DescribeStatus: Courier.Description;
  Grep: CARDINAL = 1;
  GetEOLConvention: CARDINAL = 2;
  ServiceError: CARDINAL = 100;
  DescribeServiceProblem: Courier.Description;
  TransferError: CARDINAL = 101;
  DescribeTransferProblem: Courier.Description;
  GrepArgs: TYPE = RECORD[matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: LONG STRING, file: LONG
  STRING, output: XStream.Handle];
  DescribeGrepArgs: Courier.Description;

  GrepRes: TYPE = RECORD[status: PublicCommands.Status];
  DescribeGrepRes: Courier.Description;

  GetEOLConventionRes: TYPE = RECORD[eolConvention: LONG STRING];
  DescribeGetEOLConventionRes: Courier.Description;

  ServiceErrorArgs: TYPE = RECORD[problem: PublicCommands.ServiceProblem];
  DescribeServiceErrorArgs: Courier.Description;

  TransferErrorArgs: TYPE = RECORD[problem: PublicCommands.TransferProblem];
  DescribeTransferErrorArgs: Courier.Description;


  }.
```

```
-- PublicCommandsDescription.mesa
-- Trow  4-Oct-89 13:20:27

DIRECTORY Courier, PublicCommandsCourier, PublicCommands, XStream;

PublicCommandsDescription: PROGRAM
  IMPORTS XStream
  EXPORTS PublicCommandsCourier = PUBLIC {

  DescribeStatus: Courier.Description = {
    p: LONG POINTER TO PublicCommands.Status = notes.noteSize[
      SIZE[PublicCommands.Status]];
    };

  DescribeGrepArgs: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.GrepArgs = notes.noteSize[
      SIZE[PublicCommandsCourier.GrepArgs]];
    notes.noteString[@p.pattern];
    notes.noteString[@p.file];
    notes.noteParameters[@p.output, XStream.DescribeSink];
    };

  DescribeGrepRes: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.GrepRes = notes.noteSize[
      SIZE[PublicCommandsCourier.GrepRes]];
    notes.noteParameters[@p.status, DescribeStatus];
    };

  DescribeGetEOLConventionRes: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.GetEOLConventionRes = notes.noteSize[
      SIZE[PublicCommandsCourier.GetEOLConventionRes]];
    notes.noteString[@p.eolConvention];
    };

  DescribeServiceErrorArgs: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.ServiceErrorArgs = notes.noteSize[
      SIZE[PublicCommandsCourier.ServiceErrorArgs]];
    notes.noteParameters[@p.problem, DescribeServiceProblem];
    };

  DescribeServiceProblem: Courier.Description = {
    p: LONG POINTER TO PublicCommands.ServiceProblem = notes.noteSize[
      SIZE[PublicCommands.ServiceProblem]];
    };

  DescribeTransferErrorArgs: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.TransferErrorArgs = notes.noteSize[
      SIZE[PublicCommandsCourier.TransferErrorArgs]];
    notes.noteParameters[@p.problem, DescribeTransferProblem];
    };

  DescribeTransferProblem: Courier.Description = {
    p: LONG POINTER TO PublicCommands.TransferProblem = notes.noteSize[
      SIZE[PublicCommands.TransferProblem]];
    };



  }.
```

```
-- PublicCommandsServerImpl.mesa
-- Trow  4-Oct-89 13:24:44

DIRECTORY Courier, Heap, System, PublicCommands, PublicCommandsCourier, Stream, XStream;

PublicCommandsServerImpl: PROGRAM
  IMPORTS Heap, Courier, PublicCommands, PublicCommandsCourier, Stream, XStream
  EXPORTS PublicCommands = {

  BindHandle: TYPE = LONG POINTER TO READONLY BindObject;
  BindObject: PUBLIC TYPE = Courier.Object;


  Dispatcher: Courier.Dispatcher = {
    ENABLE {
      PublicCommands.ServiceError => GOTO error100;
      PublicCommands.TransferError => GOTO error101;
      };
    SELECT procedureNumber FROM
      1 => DoGrep[cH, arguments, results];
      2 => DoGetEOLConvention[cH, arguments, results];
      ENDCASE => ERROR Courier.NoSuchProcedureNumber;
    EXITS
      error100 => {
        args: PublicCommandsCourier.ServiceErrorArgs ← [problem];
        Courier.SignalRemoteError[100, [@args, PublicCommandsCourier.DescribeServiceErrorArgs]];
        };
      error101 => {
        args: PublicCommandsCourier.TransferErrorArgs ← [problem];
        Courier.SignalRemoteError[101, [@args, PublicCommandsCourier.DescribeTransferErrorArgs]];
        };
    };

  DoGrep: PROCEDURE[
    cH: Courier.Handle, arguments: Courier.Arguments, results: Courier.Results] = {
    args: PublicCommandsCourier.GrepArgs;
    res: PublicCommandsCourier.GrepRes;
    GrepBulkData: PROCEDURE[xH: XStream.Handle] = {
      xstream: Stream.Handle ← XStream.Create[xH];
      [res.status] ← PublicCommands.Grep[NIL, args.matchCase, args.matchWords, args.pattern, args.file,
      xstream!UNWIND => Stream.Delete[xstream]];
      Stream.Delete[xstream];
      };
    arguments[[@args, PublicCommandsCourier.DescribeGrepArgs]];
    XStream.ServerCheckout[cH, [proc[GrepBulkData]]];
    [] ← results[[@res, PublicCommandsCourier.DescribeGrepRes]];
    Courier.Free[[@args, PublicCommandsCourier.DescribeGrepArgs], cH.zone];
    };

  DoGetEOLConvention: PROCEDURE[
    cH: Courier.Handle, arguments: Courier.Arguments, results: Courier.Results] = {
    res: PublicCommandsCourier.GetEOLConventionRes;
    arguments[];
    [res.eolConvention] ← PublicCommands.GetEOLConvention[NIL];
    [] ← results[[@res, PublicCommandsCourier.DescribeGetEOLConventionRes]];
    PublicCommands.FreeGetEOLConventionResults[bH: NIL, eolConvention: res.eolConvention];
    };


  started: BOOLEAN ← FALSE;
  RemoteBind: PUBLIC PROCEDURE [
    host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
    RETURNS[bH: BindHandle] = {
    bH ← NIL;
    IF zone = NIL THEN zone ← Heap.systemZone;
    IF started THEN RETURN;
    Courier.ExportRemoteProgram[
      programNumber: PublicCommandsCourier.programNumber,
      versionRange: [PublicCommandsCourier.version,PublicCommandsCourier.version],
      dispatcher: Dispatcher, serviceName: "PublicCommands"L,
      zone: zone, classOfService: transactional];
    started ← TRUE;
  };

  RemoteUnbind: PUBLIC PROCEDURE[
    bH: BindHandle] RETURNS [nil: BindHandle] = {
```

```
    nil ← NIL;
    IF ~started THEN RETURN;
    Courier.UnexportRemoteProgram[
      programNumber: PublicCommandsCourier.programNumber,
      versionRange: [PublicCommandsCourier.version,PublicCommandsCourier.version]];
    started ← FALSE;
    };


}.
```

```
-- File: CvAsciiDataImpl.mesa
-- Last Revised by: Erickson                    24-Nov-87 16:54:21
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Courier
    USING [Description, DeserializeParameters, Error, Free,
    Parameters, SerializeParameters],
  Converter
    USING [CreateClientFile, CvData, DestroyProc, FindClientFile],
  ConverterMsg
    USING [Get, kvpDocument],
  CvAscii
    USING [Common, CommonData, CommonObj,
    GetMessage, Owners, Problem, TextIDs],
  Environment
    USING [bytesPerPage],
  Heap
    USING [Create, Delete],
  NSFile
    USING [Delete, Error, Handle, nullReference, OpenByReference],
  NSFileStream
    USING [Create, GetLength, Handle, SetLength],
  Stream
    USING [Delete, InvalidOperation],
  Window
    USING [Handle],
  XString
    USING [CopyToNewReaderBody, DescribeReaderBody, nullReaderBody, ReaderBody];

<<
-- OVERVIEW:

Data and filed data procedures
>>

CvAsciiDataImpl: PROGRAM
IMPORTS
  Converter, ConverterMsg, Courier, CvAscii, Heap,
  NSFile, NSFileStream, Stream, XString
EXPORTS
  CvAscii =
BEGIN

-- CONSTANTS

keyBits: Key = 2707974433;    --/* never change this value! */

currentVersion: Version = 1;  --/* change this value if you alter the filed data format */
-- History of Versions (update each time version number changes)
-- 18-Mar-87 11:48:29 -  1 - First version

-- TYPES

Key: TYPE = LONG CARDINAL;

Version: TYPE = INTEGER;

-- PUBLIC PROCEDURES

CreateCommon: PUBLIC PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: CvAscii.Owners] RETURNS [my:
CvAscii.Common] = {
    z: UNCOUNTED ZONE ← Heap.Create[initial: 16, increment: 28];

    my ← z.NEW[CvAscii.CommonData ← [
      cvData: cvData,
      options: options,
      window: window,
      owner: owner,
      ref: NSFile.nullReference,
      textRb: ALL[XString.nullReaderBody],
      text: ALL[NIL],
      z: z]];

    --/* find client file */
    BEGIN
      ENABLE UNWIND => Heap.Delete[z];
    prefix: XString.ReaderBody ← CvAscii.GetMessage[prefix];
    src: XString.ReaderBody ← CvAscii.GetMessage[asciiSrcDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.FindClientFile[
      cvData: cvData,
      srcFormat: @src,
```

```
        destFormat: @dst,
        prefix: @prefix];

      IF my.ref = NSFile.nullReference THEN
          {
          --/* file never created, so initialize */
          InitFiledData[my];  --/* fills in my.ref */
          };

      --/* read data */
      BEGIN
        ENABLE CvAscii.Problem =>
          {
          file: NSFile.Handle ← NSFile.OpenByReference[my.ref];
          aToVMeta: XString.ReaderBody ← CvAscii.GetMessage[aToVDfltMeta];
          meta: XString.ReaderBody ← CvAscii.GetMessage[dfltMeta];
          char: XString.ReaderBody ← CvAscii.GetMessage[dfltChar];
          --/* get rid of old file, reinitialize */
          NSFile.Delete[file];
          InitFiledData[my];
          my.textRb ← [
            paraEndsWith: aToVMeta,
            atovReplaceUnknown: char,
            endLine: meta,
            endPara: meta,
            vtoaReplaceUnknown: char,
            replaceOffice: char,
            spare0: char,
            spare1: char,
            spare2: char];
          CONTINUE;
          };

      LoadFiledData[my];
      END;
      END;
};


DestroyCommon: PUBLIC Converter.DestroyProc = {
<< = PROCEDURE [instance: LONG POINTER];
>>
    my: CvAscii.Common ← instance;
    z: UNCOUNTED ZONE;

    IF my = NIL THEN RETURN;
    z ← my.z;
    Heap.Delete[z];
};


InitFiledData: PUBLIC PROC [my: CvAscii.Common] = {
    myObj: CvAscii.CommonData;
    aToVMeta: XString.ReaderBody ← CvAscii.GetMessage[aToVDfltMeta];
    meta: XString.ReaderBody ← CvAscii.GetMessage[dfltMeta];
    char: XString.ReaderBody ← CvAscii.GetMessage[dfltChar];

    --/* make dummy filed data */
    myObj.f ← [];
    myObj.textRb ← [
      paraEndsWith: aToVMeta,
      atovReplaceUnknown: char,
      endLine: meta,
      endPara: meta,
      vtoaReplaceUnknown: char,
      replaceOffice: char,
      spare0: char,
      spare1: char,
      spare2: char];

    --/* create client file */
    BEGIN
    prefix: XString.ReaderBody ← CvAscii.GetMessage[prefix];
    src: XString.ReaderBody ← CvAscii.GetMessage[asciiSrcDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.CreateClientFile[
      cvData: my.cvData,
      srcFormat: @src,
      destFormat: @dst,
      prefix: @prefix];
    END;

    myObj.ref ← my.ref;
    myObj.z ← my.z;
    myObj.owner ← backstop;  --/* let StoreFiledData know we are initializing */
    --/* store */
    StoreFiledData[@myObj];
};


LoadFiledData: PUBLIC PROC [my: CvAscii.Common] = {
    sh: NSFileStream.Handle ← [NIL];
    file: NSFile.Handle;
    parms: Courier.Parameters;
    tz: UNCOUNTED ZONE ← NIL;
```

```
    --/* read filed data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND =>
            {
            Stream.Delete[sh];
            IF tz # NIL THEN Heap.Delete[tz];
            };
        };

    --/* open data file */
    file ← NSFile.OpenByReference[my.ref];

    --/* open read stream on data file */
    sh ← NSFileStream.Create[file: file, closeOnDelete: TRUE];

    --/* create temporary zone for disjoint data */
    tz ← Heap.Create[(NSFileStream.GetLength[sh]/Environment.bytesPerPage) + 2];

    --/* read key */
    BEGIN
    key: Key;

    parms ← [location: @key, description: DescribeKey];
    Courier.DeserializeParameters[parms, sh, tz];
    IF key # keyBits THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        SIGNAL CvAscii.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read version */
    BEGIN
    ver: Version;

    parms ← [location: @ver, description: DescribeVersion];
    Courier.DeserializeParameters[parms, sh, tz];
    IF ver # currentVersion THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        SIGNAL CvAscii.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read commonObj */
    parms ← [location: @my.f, description: DescribeCommonObj];
    Courier.DeserializeParameters[parms, sh, tz];

    --/* read paraEndsWith */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[paraEndsWith] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read atovReplaceUnknown */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[atovReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endLine */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endLine] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endPara */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endPara] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read vtoaReplaceUnknown */
    BEGIN
```

IF sh # NSFileStream.Handle[NIL] THEN

NSFile.Close[file];
sh ← [NIL];

NSFile.Close[file];
sh ← [NIL];

```
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[vtoaReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read replaceOffice */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[replaceOffice] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* skip spares */
<<
    THROUGH [0..3) DO
        rb: XString.ReaderBody;
        parms ← [location: @rb, description: XString.DescribeReaderBody];
        Courier.DeserializeParameters[parms, sh, tz];
        Courier.Free[parms, tz];
    ENDLOOP;
>>

    END;

    --/* clean up */
    Stream.Delete[sh];
    Heap.Delete[tz];
};



--=============================================================================
-- StoreFiledData
-- * This is tricky, since common data is used.  This routine could be called
-- * three different times, with different subsets of data, but the whole
-- * file must be written each time.
--=============================================================================

StoreFiledData: PUBLIC PROC [my: CvAscii.Common] = {
    dataFile: NSFile.Handle;
    sh: NSFileStream.Handle;
    parms: Courier.Parameters;
    tmpMy: CvAscii.CommonData;

    --/* fill out dummy */
    tmpMy ← my↑;
    IF my.owner # backstop THEN
        LoadFiledData[@tmpMy];

    --/* open data file */
    dataFile ← NSFile.OpenByReference[my.ref];

    --/* open stream on file */
    sh ← NSFileStream.Create[file: dataFile, closeOnDelete: TRUE];
    NSFileStream.SetLength[fileStream: sh, lengthInBytes: 0];

    --/* write data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND => Stream.Delete[sh];
        };

    --/* write key */
    BEGIN
    key: Key ← keyBits;

    parms ← [location: @key, description: DescribeKey];
    Courier.SerializeParameters[parms, sh];
    END;

    --/* write version */
    BEGIN
    ver: Version ← currentVersion;

    parms ← [location: @ver, description: DescribeVersion];
    Courier.SerializeParameters[parms, sh];
    END;

    --/* update portions of data record */
    SELECT my.owner FROM
        AtoVsrc =>
            {
            tmpMy.textRb[paraEndsWith] ← my.textRb[paraEndsWith];
            tmpMy.f.atovAsciiEncoding ← my.f.atovAsciiEncoding;
            };
        AtoVdst =>
            {
            tmpMy.f.font ← my.f.font;
            tmpMy.textRb[atovReplaceUnknown] ← my.textRb[atovReplaceUnknown];
```

```
            tmpMy.f.ignoreTrailing ← my.f.ignoreTrailing;
            };
        VtoAdst =>
            {
            tmpMy.f.vtoaAsciiEncoding ← my.f.vtoaAsciiEncoding;
            tmpMy.f.lineLen ← my.f.lineLen;
            tmpMy.f.charsSuffix ← my.f.charsSuffix;
            tmpMy.f.wordWrap ← my.f.wordWrap;
            tmpMy.textRb[endLine] ← my.textRb[endLine];
            tmpMy.textRb[endPara] ← my.textRb[endPara];
            tmpMy.textRb[vtoaReplaceUnknown] ← my.textRb[vtoaReplaceUnknown];
            tmpMy.textRb[replaceOffice] ← my.textRb[replaceOffice];
            tmpMy.f.convertTables ← my.f.convertTables;
            tmpMy.f.simulateFrames ← my.f.simulateFrames;
            };
        ENDCASE;

    --/* write filed data record */
    parms ← [location: @tmpMy.f, description: DescribeCommonObj];
    Courier.SerializeParameters[parms, sh];

    --/* write paraEndsWith string */
    parms ← [location: @tmpMy.textRb[paraEndsWith], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write atovReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[atovReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endLine string */
    parms ← [location: @tmpMy.textRb[endLine], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endPara string */
    parms ← [location: @tmpMy.textRb[endPara], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write vtoaReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[vtoaReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write replaceOffice string */
    parms ← [location: @tmpMy.textRb[replaceOffice], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write spare0 string */
    parms ← [location: @tmpMy.textRb[spare0], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write spare1 string */
    parms ← [location: @tmpMy.textRb[spare1], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write spare2 string */
    parms ← [location: @tmpMy.textRb[spare2], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    END;

    Stream.Delete[sh];
};

--====================
-- PROCEDURES
--====================

DescribeKey: Courier.Description = {
    p: LONG POINTER TO Key = notes.noteSize[SIZE[Key]];
    notes.noteLongCardinal[p];
};

DescribeVersion: Courier.Description = {
    p: LONG POINTER TO Version = notes.noteSize[SIZE[Version]];
};

DescribeCommonObj: Courier.Description = {
    p: LONG POINTER TO CvAscii.CommonObj = notes.noteSize[
      SIZE[CvAscii.CommonObj]];
};

END...

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:55:56 - Erickson - Changed default setting of paraEndsWith
                        to <CR> instead of <CR><LF>
```

```
-- File: RS232Chat.config - last edit:
-- Hodges.pa            18-Jul-85  9:39:36
-- RS232XChat.config
-- Yamamoto      21-May-84  9:41:43

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

RS232Chat: CONFIGURATION
  IMPORTS
    ByteBlt, DeviceCleanup,
    DLionInputOutput, FormSW, Heap, MStream,
    Process, Profile, Put, ResidentHeap, Runtime,
    SpecialHeap, SpecialRuntime, SpecialSpace, Stream,
    String, TIP, Time,
    Tool, TTY, TTYSW, UserTerminal, --Version,--
    WindowFont
  CONTROL RS232ChatImpl = {
    RS232CIO;
    RS232CIOHeadsDLion;
    RS232ChatImpl;
    }...
```

```
-- File: RS232ChatImpl.mesa - last edit:
-- Hodges.pa            12-Aug-85  0:47:35
-- RS232XChatImpl.mesa
-- Create by FormSWLayoutTool on 17-May-84 22:42
-- Yamamoto      21-May-84  9:56:28

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii,
  Environment,
  Format,
  FormSW,
  Heap,
  MStream,
  Process,
  Profile,
  Put,
  RS232C,
  RS232CCorrespondents,
  Runtime,
  Stream,
  String,
  Time,
  TIP,
  Tool,
  ToolWindow,
  TTY,
  TTYSW,
  UserInput,
  UserTerminal,
  Version,
  Window,
  WindowFont;

RS232ChatImpl: MONITOR
  IMPORTS
    FormSW, Heap, MStream, Process,
    Profile, Put, RS232C, Runtime, Stream, String, Time, Tool,
    TIP, TTY, TTYSW, UserTerminal, -- Version,-- WindowFont = {

  DataHandle: TYPE = LONG POINTER TO Data;
  Data: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    ttySW(4): Window.Handle ← NIL,
    lineSpeed(6): RS232C.LineSpeed ← bps9600,
    parity(7): RS232C.Parity ← none,
    stopBits(8): INTEGER ← 1,
    duplex(9): RS232C.Duplexity ← full,
    charLength(10): INTEGER ← 8,
    cH(11): RS232C.ChannelHandle ← TRASH,
    get(13): PROCESS ← NIL,
    put(14): PROCESS ← NIL,
    DisplayDataProcess(15): PROCESS ← NIL,
    connected(16): BOOLEAN ← FALSE,
    tty(17): TTY.Handle ← TTY.nullHandle,
    UserInputProcess(19): PROCESS ← NIL,
    flowControl(20): MyFlowControlStateIndicator ← on,
    mode(21): MyModeStateIndicator ← normal,
    xON(22): UNSPECIFIED ← DC1,
    xOFF(23): UNSPECIFIED ← DC3];

  MyFlowControlStateIndicator: TYPE = {on, off};
  MyModeStateIndicator: TYPE = {normal, raw};
  MyTransitionIndicator: TYPE = {goingInactive, goingActive};

  myTransitionIndicator: MyTransitionIndicator ← goingActive;

  DC1: UNSPECIFIED = 11H;
  DC3: UNSPECIFIED = 13H;

  QueueHandle: TYPE =  LONG POINTER TO QueueObject;
  QueueObject: TYPE = RECORD[
    next: QueueHandle,
    lostChars: BOOLEAN,
    cH: RS232C.CompletionHandle,
    string: LONG STRING];

  debug: Stream.Handle ← NIL;

  StuffOnQueue: CONDITION;
  aborting: BOOLEAN ← FALSE;

  FormItems: TYPE = {connect, disconnect, mode, baud, parity, stop, duplex, charWidth, flowControl};

  data: DataHandle ← NIL;
  wh: Window.Handle ← NIL;
  zone: UNCOUNTED ZONE ← Heap.Create[
    initial: 15, increment: 6, largeNodeThreshold: 512];

  Msg: Format.StringProc = {Put.Text[data.msgSW, s]};

  tip: PUBLIC TIP.Table ← NIL;

  herald: LONG STRING ← NIL;
  heap: PUBLIC UNCOUNTED ZONE ← Heap.systemZone;
```

```
  Connect:  FormSW.ProcType = {
    commParamObject: RS232C.CommParamObject ← [
      duplex: data.duplex, lineType: asynchronous,
      lineSpeed: data.lineSpeed, accessDetail: directConn[]];

    IF data.connected THEN
        Msg["Already connected.\n"L]
    ELSE
      {
        Msg["Connecting ... "L];
        aborting ← FALSE;
        data.cH ← RS232C.Create[0, @commParamObject, preemptAlways, preemptNever];
        data.connected ← TRUE;
        data.cH.SetParameter[[charLength[data.charLength]]];
        data.cH.SetParameter[[correspondent[RS232CCorrespondents.ttyHost]]];
        data.cH.SetParameter[[dataTerminalReady[TRUE]]];
        data.cH.SetParameter[[frameTimeout[5]]];
        data.cH.SetParameter[[lineSpeed[data.lineSpeed]]];
        data.cH.SetParameter[[parity[data.parity]]];
        data.cH.SetParameter[[requestToSend[TRUE]]];
        data.cH.SetParameter[[stopBits[data.stopBits]]];
        ChangeFlowControl[];
        data.get ← FORK GetData[];
        data.put ← FORK PutData[];
        data.DisplayDataProcess ← FORK TTYToDisplay[];
        data.UserInputProcess ← FORK KeyToTTY[];
        IF Profile.debugging THEN debug ← MStream.Log[
          "TTYDebug.log"L, [Release, NIL]];
        Msg["Connection open to RS232C port.\n"L];
      };
    };

  Release: MStream.PleaseReleaseProc = {
    MStream.SetLogReadLength[stream, stream.GetPosition[]];
    RETURN[no]};

  Disconnect:  FormSW.ProcType = {
    IF ~data.connected THEN
        Msg["Not connected.\n"L]
    ELSE
      {
        Msg["Disconnecting ... "L];
        aborting ← TRUE;
        data.cH.SetParameter[[dataTerminalReady[FALSE]]];
        data.cH.Suspend[all];
        Process.Abort[data.get];
        JOIN data.get;
        data.get ← NIL;
        Process.Abort[data.put];
        JOIN data.put;
        data.put ← NIL;
        Process.Abort[data.DisplayDataProcess];
        JOIN data.DisplayDataProcess;
        data.DisplayDataProcess ← NIL;
        Process.Abort[data.UserInputProcess];
        JOIN data.UserInputProcess;
        data.UserInputProcess ← NIL;
        data.cH.Delete[];
        IF debug # NIL THEN {debug.Delete[]; debug ← NIL};
        data.connected ← FALSE;
        IF myTransitionIndicator # goingInactive THEN
            Msg["Disconnected.\n"L];
      };
    };

  ChangeBaud:  FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      data.cH.SetParameter[[lineSpeed[data.lineSpeed]]];
    };

  ChangeParity:  FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      data.cH.SetParameter[[parity[data.parity]]];
    };

  ChangeStopBits:  FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      data.cH.SetParameter[[stopBits[data.stopBits]]];
    };

  ChangeDuplexity:  FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      Msg["Must disconnect to change duplexity.\n"L];
    };

  ChangeCharLength:  FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      data.cH.SetParameter[[charLength[data.charLength]]];
    };

  ChangeFlowControl: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      IF data.flowControl = on THEN
          data.cH.SetParameter[[flowControl[[type: xOnXOff, xOn: DC1, xOff: DC3]]]]
      ELSE
          data.cH.SetParameter[[flowControl[[type: none, xOn: 0, xOff: 0]]]];
```

```
    };

  ChangeMode: FormSW.EnumeratedNotifyProcType = {
    };

  ClientTransition: ToolWindow.TransitionProcType = {
    SELECT TRUE FROM
      old = inactive => {
        myTransitionIndicator ← goingActive;
        IF data = NIL THEN data ← zone.NEW[Data ← []];
        };
      new = inactive =>
        IF data # NIL THEN {
          myTransitionIndicator ← goingInactive;
          IF data.connected THEN  Disconnect[];
          zone.FREE[@data]};
      ENDCASE;
    };

<< Process name is "get" >>

  GetData: PROCEDURE  = {  -- forked as Process
    buffer: QueueHandle ← NIL;
    status: RS232C.TransferStatus;
    byteCount: CARDINAL;
    rec: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock, body: Environment.nullBlock,
      trailer: Environment.nullBlock];
    Process.SetPriority[5];
    BEGIN
      ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
      DO
        ENABLE ABORTED => EXIT;
        buffer ← GetFromQueue[@freelist];
        rec.body ← [LOOPHOLE[@buffer.string.text], 0, buffer.string.maxlength];
        buffer.cH ← data.cH.Get[@rec];
        PutToQueue[@outlist, buffer];
      ENDLOOP;
      EXITS quit => NULL;
    END;
    WHILE outlist # NIL DO
      buffer ← GetFromQueue[@outlist, FALSE];
      IF buffer = NIL THEN EXIT;
      [byteCount, status] ← data.cH.TransferWait[buffer.cH
                                                 ! ABORTED => NULL];
      InitBuffer[buffer];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    };

<< Process name is "put" >>

  PutData: PROCEDURE  = {  -- forked as Process
    status: RS232C.TransferStatus;
    byteCount: CARDINAL;
    deviceStatus: RS232C.DeviceStatus;
    buffer: QueueHandle ← NIL;
    Process.SetPriority[5];
    BEGIN  -- forked as Process
      ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
      DO
        ENABLE ABORTED => EXIT;
        buffer ← GetFromQueue[@outlist];
        [byteCount, status] ← data.cH.TransferWait[buffer.cH];
        buffer.string.length ← byteCount;
        IF status = aborted THEN EXIT;
        deviceStatus ← data.cH.GetStatus[];
        IF deviceStatus.statusAborted THEN EXIT;
        buffer.lostChars ← deviceStatus.dataLost OR deviceStatus.deviceError
          OR status # success;
        IF buffer.lostChars AND Profile.debugging THEN
          NoteError[deviceStatus, status];
        PutToQueue[@filledlist, buffer];
      ENDLOOP;
      EXITS quit => NULL;
    END; -- of enabled
    WHILE filledlist # NIL DO
      buffer ← GetFromQueue[@filledlist, FALSE];
      IF buffer = NIL THEN EXIT;
      InitBuffer[buffer];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    };

  NoteError: PROC[
    deviceStatus: RS232C.DeviceStatus, status: RS232C.TransferStatus] = {
    debug.PutString["Device status: ["L];
    PutBoolean[deviceStatus.statusAborted, "statusAborted"L, TRUE];
    PutBoolean[deviceStatus.dataLost, "dataLost"L, TRUE];
    PutBoolean[deviceStatus.breakDetected, "breakDetected"L, TRUE];
    PutBoolean[deviceStatus.clearToSend, "clearToSend"L, TRUE];
    PutBoolean[deviceStatus.dataSetReady, "dataSetReady"L, TRUE];
    PutBoolean[deviceStatus.carrierDetect, "carrierDetect"L, TRUE];
    PutBoolean[deviceStatus.ringHeard, "ringHeard"L, TRUE];
    PutBoolean[deviceStatus.ringIndicator, "ringIndicator"L, TRUE];
    PutBoolean[deviceStatus.deviceError, "deviceError"L, FALSE];
```

```
    debug.PutString["]\n"L];
    debug.PutString["Transfer status: "L];
    debug.PutString[SELECT status FROM
      success => "success"L,
      dataLost => "dataLost"L,
      deviceError => "deviceError"L,
      frameTimeout => "frameTimeout"L,
      checksumError => "checksumError"L,
      parityError => "parityError"L,
      asynchFramingError => "asynchFramingError"L,
      invalidChar => "invalidChar"L,
      invalidFrame => "invalidFrame"L,
      aborted => "aborted"L,
      ENDCASE => "disaster"L];
    debug.PutString["\n"L];
    };

  PutBoolean: PROC[b: BOOLEAN, s: LONG STRING, comma: BOOLEAN] = {
    debug.PutString[s];
    debug.PutString[": "L];
    debug.PutString[IF b THEN "TRUE"L ELSE "FALSE"L];
    IF comma THEN debug.PutString[", "L];
    };

  bufferMax: CARDINAL = 128;

  Init: PROCEDURE = {
    buffer: QueueHandle ← NIL;
    firstTime: BOOLEAN ← TRUE;
    fileName: STRING = "RS232Chat.TIP"L;
    contents: LONG STRING =
"-- RS232Chat.TIP; of 7-Jun-85 16:41:11
-- Created by System

SELECT TRIGGER FROM
  ENDCASE...
"L;
    tip ← TIP.CreateTable[file: fileName, contents: contents !
      TIP.InvalidTable =>
        IF type = badSyntax THEN {
          IF firstTime THEN {firstTime ← FALSE; Flash[]; RESUME}
          ELSE CONTINUE}];
    IF tip  # NIL THEN
      [] ← TIP.PushLocal[push: tip , onto: TIP.globalTable[ttySW]];

    FOR i: CARDINAL IN [0..10) DO
      buffer ← zone.NEW[
        QueueObject ← [
        next: NIL, lostChars: FALSE, cH: TRASH,
        string: zone.NEW[StringBody [bufferMax]]]];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    Process.SetTimeout[@StuffOnQueue, 1];
    Process.EnableAborts[@StuffOnQueue];
    herald ← heap.NEW[StringBody [40]];
    String.AppendString[herald, "RS232Chat v2.1 of "L];
    Time.Append[herald, Time.Unpack[Runtime.GetBcdTime[]]];
--    String.AppendString[herald, " running on Pilot "L];
--    Version.Append[herald];
    herald.length ← herald.length - 3; -- gun the seconds

    wh ← Tool.Create[
      makeSWsProc: MakeSWs, initialState: default, initialBox:[[0,30],[612,512]],
      clientTransition: ClientTransition, name: herald,
      cmSection: "RS232Chat"L];
    };

  << Process name is "UserInputProcess" >>

    KeyToTTY: PROC =
    {
      ch: CHARACTER;
      chs: PACKED ARRAY [0..2) OF Environment.Byte;
      ptr: LONG POINTER TO UNSPECIFIED ← @chs;
      physRecord: RS232C.PhysicalRecord ← [
        header: Environment.nullBlock,
        body: [
          startIndex: 0, stopIndexPlusOne: 1,
          blockPointer: ptr],
        trailer: Environment.nullBlock];

      Process.SetPriority[5];
      BEGIN
        ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
        DO
          ENABLE ABORTED => EXIT;
          ch ← data.tty.GetChar[];
          chs[0] ← LOOPHOLE[ch];
          [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
        ENDLOOP;
        EXITS quit => NULL;
      END;
    };

  << Process name is "DisplayDataProcess" >>

    TTYToDisplay: PROCEDURE = {
```

```
    debug.PutString["]\n"L];
    debug.PutString["Transfer status: "L];
    debug.PutString[SELECT status FROM
      success => "success"L,
      dataLost => "dataLost"L,
      deviceError => "deviceError"L,
      frameTimeout => "frameTimeout"L,
      checksumError => "checksumError"L,
      parityError => "parityError"L,
      asynchFramingError => "asynchFramingError"L,
      invalidChar => "invalidChar"L,
      invalidFrame => "invalidFrame"L,
      aborted => "aborted"L,
      ENDCASE => "disaster"L];
    debug.PutString["\n"L];
    };

  PutBoolean: PROC[b: BOOLEAN, s: LONG STRING, comma: BOOLEAN] = {
    debug.PutString[s];
    debug.PutString[": "L];
    debug.PutString[IF b THEN "TRUE"L ELSE "FALSE"L];
    IF comma THEN debug.PutString[", "L];
    };

  bufferMax: CARDINAL = 128;

  Init: PROCEDURE = {
    buffer: QueueHandle ← NIL;
    firstTime: BOOLEAN ← TRUE;
    fileName: STRING = "RS232Chat.TIP"L;
    contents: LONG STRING =
"-- RS232Chat.TIP; of 7-Jun-85 16:41:11
-- Created by System

SELECT TRIGGER FROM
  ENDCASE...
"L;
    tip ← TIP.CreateTable[file: fileName, contents: contents !
      TIP.InvalidTable =>
        IF type = badSyntax THEN {
          IF firstTime THEN {firstTime ← FALSE; Flash[]; RESUME}
          ELSE CONTINUE}];
    IF tip  # NIL THEN
      [] ← TIP.PushLocal[push: tip , onto: TIP.globalTable[ttySW]];

    FOR i: CARDINAL IN [0..10) DO
      buffer ← zone.NEW[
        QueueObject ← [
        next: NIL, lostChars: FALSE, cH: TRASH,
        string: zone.NEW[StringBody [bufferMax]]]];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    Process.SetTimeout[@StuffOnQueue, 1];
    Process.EnableAborts[@StuffOnQueue];
    herald ← heap.NEW[StringBody [40]];
    String.AppendString[herald, "RS232Chat v2.1 of "L];
    Time.Append[herald, Time.Unpack[Runtime.GetBcdTime[]]];
--    String.AppendString[herald, " running on Pilot "L];
--    Version.Append[herald];
    herald.length ← herald.length - 3; -- gun the seconds

    wh ← Tool.Create[
      makeSWsProc: MakeSWs, initialState: default, initialBox:[[0,30],[512,512]],
      clientTransition: ClientTransition, name: herald,
      cmSection: "RS232Chat"L];
    };

<< Process name is "UserInputProcess" >>

  KeyToTTY: PROC =
  {
    ch: CHARACTER;
    chs: PACKED ARRAY [0..2) OF Environment.Byte;
    ptr: LONG POINTER TO UNSPECIFIED ← @chs;
    physRecord: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock,
      body: [
        startIndex: 0, stopIndexPlusOne: 1,
        blockPointer: ptr],
      trailer: Environment.nullBlock];

    Process.SetPriority[6];
    BEGIN
      ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
      DO
        ENABLE ABORTED => EXIT;
        ch ← data.tty.GetChar[];
        chs[0] ← LOOPHOLE[ch];
        [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
      ENDLOOP;
      EXITS quit => NULL;
    END;
  };

<< Process name is "DisplayDataProcess" >>

  TTYToDisplay: PROCEDURE = {
```

```
    buffer: QueueHandle ← NIL;
    s: LONG STRING ← NIL;
    i, j: CARDINAL ← 0;

    Process.SetPriority[Process.priorityForeground];

    DO
      ENABLE ABORTED => EXIT;
      buffer ← GetBuffer[!
        ABORTED =>  EXIT; ANY => LOOP];
      IF buffer.lostChars THEN
         Msg["Characters lost.\n"L];
      j ← 0;
      s ← buffer.string;

      FOR i IN [0..s.length) DO
        IF data.mode = normal THEN       -- process certain cntl chars specially
          {
            SELECT  s[i] FROM
              Ascii.NUL,
              Ascii.LF,
              Ascii.DEL => NULL;

              Ascii.BS =>
                {
                  IF j > 0 THEN
                    {
                      s.length ← j;
                      data.tty.PutString[s];
                      s.length ← j ← 0;
                    };
                  data.tty.RemoveCharacter[];
                };
              ENDCASE => {s[j] ← s[i]; j ← j + 1};
          }
        ELSE  -- mode is raw, send all chars through regardless...
          {
            s[j] ← s[i];
            j ← j + 1;
          };
      ENDLOOP;

      IF j > 0 THEN
        {
          s.length ← j;
          data.tty.PutString[s];
        };

      ReturnBuffer[buffer];
      buffer ← NIL;
    ENDLOOP;
    IF buffer # NIL THEN ReturnBuffer[buffer];
    };  -- TTYToDisplay

  freelist, outlist, filledlist: QueueHandle ← NIL;

  GetBuffer: PROC RETURNS [buffer: QueueHandle] = {
    ENABLE UNWIND => NULL;
    buffer ← GetFromQueue[@filledlist];
    };  -- GetBuffer

  ReturnBuffer: PUBLIC PROC [buffer: QueueHandle] = {
    InitBuffer[buffer]; PutToQueue[@freelist, buffer]};

  InitBuffer: PROC [buffer: QueueHandle] = {
    buffer.lostChars ← FALSE; buffer.next ← NIL; };

  PutToQueue: ENTRY PROC [q: LONG POINTER TO QueueHandle, buf: QueueHandle] = {
    i: LONG POINTER TO QueueHandle ← NIL;
    FOR i ← q, @i.next UNTIL i↑ = NIL DO ENDLOOP;
    i↑ ← buf;
    buf.next ← NIL;
    BROADCAST StuffOnQueue;
    };

  GetFromQueue: ENTRY PROC [q: LONG POINTER TO QueueHandle, wait: BOOLEAN ← TRUE]
    RETURNS [buf: QueueHandle] = {
    ENABLE UNWIND => NULL;
    IF wait THEN WHILE q↑ = NIL DO
      WAIT StuffOnQueue;
      IF aborting THEN ERROR ABORTED;
      ENDLOOP;
    buf ← q↑;
    IF buf # NIL THEN q↑ ← buf.next};

  MakeSWs: Tool.MakeSWsProc = {
    logName: STRING ← [40];
    data.msgSW ← Tool.MakeMsgSW[window: window];
    data.formSW ← Tool.MakeFormSW[
      window: window, formProc: MakeForm];
    Tool.UnusedLogName[unused: logName, root: "RS232Chat.log"L];
    data.ttySW ← Tool.MakeTTYSW[window: window, name: logName];
    data.tty ← TTYSW.GetTTYHandle[data.ttySW];
    IF tip # NIL THEN [] ← TIP.SetTable[data.ttySW, tip];
   };

  MakeForm: FormSW.ClientItemsProcType = {
```

```
    OPEN FormSW;
    nItems: CARDINAL = FormItems.LAST.ORD + 1;
    baudArray: ARRAY[0..6) OF Enumerated ← [
      ["300"L, RS232C.LineSpeed.bps300.ORD], ["600"L, RS232C.LineSpeed.bps600.ORD],
      ["1200"L, RS232C.LineSpeed.bps1200.ORD], ["2400"L, RS232C.LineSpeed.bps2400.ORD],
      ["4800"L, RS232C.LineSpeed.bps4800.ORD], ["9600"L, RS232C.LineSpeed.bps9600.ORD]];

    parityArray: ARRAY[0..3) OF Enumerated ← [
      ["none"L, 0], ["odd"L, 1],
      ["even"L, 2]];

    stop: ARRAY[0..2) OF Enumerated ← [
      ["one"L, 1], ["two"L, 2]];

    duplexArray: ARRAY[0..2) OF Enumerated ← [
      ["full"L, 0], ["half"L, 1]];

    bits: ARRAY[0..4) OF Enumerated ← [
      ["5"L, 5], ["6"L, 6],
      ["7"L, 7], ["8"L, 8]];

    flowControlArray: ARRAY[0..2) OF Enumerated ← [
      ["FlowCntl ON"L, 0], ["FlowCntl OFF"L, 1]];

    modeArray: ARRAY[0..2) OF Enumerated ← [
      ["Normal"L, 0], ["Raw"L, 1]];

    items ← AllocateItemDescriptor[nItems];

<<--- Line 0 items --->>

    items[FormItems.connect.ORD] ← CommandItem[
      tag: "Connect"L, place: [CharPos[0], line0], proc: Connect];

    items[FormItems.disconnect.ORD] ← CommandItem[
      tag: "Disconnect"L, place: [CharPos[20], line0], proc: Disconnect];

    items[FormItems.mode.ORD] ← EnumeratedItem[
      tag: "Mode"L, place: [CharPos[40], line0], proc: ChangeMode, choices: DESCRIPTOR[modeArray],  value: @data.mode];

<<--- Line 1 items --->>

    items[FormItems.baud.ORD] ← EnumeratedItem[
      tag: "Baud"L, place: [CharPos[0], line1], proc: ChangeBaud, choices: DESCRIPTOR[baudArray], value: @data.lineSpeed];

    items[FormItems.parity.ORD] ← EnumeratedItem[
      tag: "Parity"L, place: [CharPos[20], line1], proc: ChangeParity, choices: DESCRIPTOR[parityArray], value: @data.parity];

    items[FormItems.stop.ORD] ← EnumeratedItem[
      tag: "Stop"L, place: [CharPos[40], line1], proc: ChangeStopBits, choices: DESCRIPTOR[stop], value: @data.stopBits];

<<--- Line 2 items --->>

    items[FormItems.duplex.ORD] ← EnumeratedItem[
      tag: "Duplex"L, place: [CharPos[0], line2], proc: ChangeDuplexity, choices: DESCRIPTOR[duplexArray], value: @data.duplex];

    items[FormItems.charWidth.ORD] ← EnumeratedItem[
      tag: "CharWidth"L, place: [CharPos[20], line2], proc: ChangeCharLength, choices: DESCRIPTOR[bits], value: @data.charLength];

    items[FormItems.flowControl.ORD] ← EnumeratedItem[
      tag: "FlowControl"L, place: [CharPos[40], line2], proc: ChangeFlowControl, choices: DESCRIPTOR[flowControlArray], value:
@data.flowControl];

<<--- Line 3 items --->>
    -- none --

    Msg["Disconnected.\n"L];

    RETURN[items: items, freeDesc: TRUE];
    };

  Flash: PUBLIC PROCEDURE =
    BEGIN
    ENABLE ABORTED => CONTINUE;
    UserTerminal.BlinkDisplay[];
    UserTerminal.Beep[262, 75];  -- c
    UserTerminal.Beep[392, 75];  -- g
    UserTerminal.Beep[659, 75];  -- e
    END;

  charWidth: CARDINAL ← WindowFont.CharWidth['0];
  CharPos: PROC[char: CARDINAL] RETURNS [x: INTEGER] = {
    x ← charWidth * char};


  -- Mainline code

  Init[];  -- this gets string out of global frame

  }...
```

```
-- Copyright (C) 1982  by Xerox Corporation. All rights reserved.
-- File: RS232C.Mesa
-- LastEdited: 21-Jan-85 11:12:35   By: SMA


DIRECTORY
  Environment USING [Byte],
  RS232CEnvironment USING [
    AutoRecognitionOutcome, CharLength, ClockSource, CommParamHandle,
    CommParamObject, CompletionHandle, Correspondent,
    DialMode, Duplexity, EncodeData, FlowControl, IdleState, LineSpeed, LineType,
    NetAccess, nullLineNumber, Parity, PhysicalRecord, PhysicalRecordHandle,
    ReserveType, StopBits, SyncChar, SyncCount];

RS232C: DEFINITIONS =
  BEGIN

  -- Interface Definitions
  Create: PROCEDURE [
    lineNumber: CARDINAL, commParams: CommParamHandle,
    preemptOthers, preemptMe: ReserveType] RETURNS [channel: ChannelHandle];
  Get: PROCEDURE [channel: ChannelHandle, rec: PhysicalRecordHandle]
    RETURNS [CompletionHandle];
  Put: PROCEDURE [channel: ChannelHandle, rec: PhysicalRecordHandle]
    RETURNS [CompletionHandle];
  TransferWait: PROCEDURE [channel: ChannelHandle, event: CompletionHandle]
    RETURNS [byteCount: CARDINAL, status: TransferStatus];
  Delete: PROCEDURE [channel: ChannelHandle];
  Suspend: PROCEDURE [channel: ChannelHandle, class: OperationClass];
  Restart: PROCEDURE [channel: ChannelHandle, class: OperationClass];
  GetStatus: PROCEDURE [channel: ChannelHandle] RETURNS [stat: DeviceStatus];
  StatusWait: PROCEDURE [channel: ChannelHandle, stat: DeviceStatus]
    RETURNS [newstat: DeviceStatus];

  --  RS232C-specific procedures  --
  AutoRecognitionWait: PROCEDURE [channel: ChannelHandle]
    RETURNS [outcome: AutoRecognitionOutcome];
  GetNextLine: PROCEDURE [lineNumber: CARDINAL] RETURNS [nextLineNumber: CARDINAL];
  TransmitNow: PROCEDURE [channel: ChannelHandle, event: CompletionHandle]
    RETURNS [byteCount: CARDINAL, status: TransferStatus];
  SendBreak: PROCEDURE [channel: ChannelHandle];
  SetParameter: PROCEDURE [channel: ChannelHandle, parameter: Parameter];
  SetLineType: PROCEDURE [channel: ChannelHandle, lineType: LineType];

  -- SIGNALs and ERRORs
  ChannelInUse, ChannelSuspended, InvalidLineNumber, InvalidParameter,
    NoRS232CHardware, SendBreakIllegal, UnimplementedFeature: ERROR;

  -- Interface type definitions
  LineType: TYPE = RS232CEnvironment.LineType;
  nullLineNumber: CARDINAL = RS232CEnvironment.nullLineNumber;
  ClockSource: TYPE = RS232CEnvironment.ClockSource;
  CommParamHandle: TYPE = RS232CEnvironment.CommParamHandle;
  CommParamObject: TYPE = RS232CEnvironment.CommParamObject;
  DialMode: TYPE = RS232CEnvironment.DialMode;
  Duplexity: TYPE = RS232CEnvironment.Duplexity;
  EncodeData: TYPE = RS232CEnvironment.EncodeData;
  IdleState: TYPE = RS232CEnvironment.IdleState;
  NetAccess: TYPE = RS232CEnvironment.NetAccess;
  ReserveType: TYPE = RS232CEnvironment.ReserveType;
  ChannelHandle: TYPE [2];
  AutoRecognitionOutcome: TYPE = RS232CEnvironment.AutoRecognitionOutcome;
  CompletionHandle: TYPE = RS232CEnvironment.CompletionHandle;
  OperationClass: TYPE = {input, output, other, all};
  PhysicalRecordHandle: TYPE = RS232CEnvironment.PhysicalRecordHandle;
  PhysicalRecord: TYPE = RS232CEnvironment.PhysicalRecord;
  TransferStatus: TYPE = {
    success, dataLost -- (caused by input buffer overrun -- , deviceError,
    frameTimeout, checksumError, parityError, asynchFramingError
    -- (i.e. stop bit(s) missing) -- , invalidChar, invalidFrame, aborted,
    disaster};
  DeviceStatus: TYPE = RECORD [
    statusAborted: BOOLEAN,
    dataLost: BOOLEAN,
    --latched: caused by arrival of data with no input buffer allocated
    breakDetected: BOOLEAN,  --latched
    clearToSend, dataSetReady, carrierDetect: BOOLEAN,
    -- correspond to signals in EIA RS-232-C Spec
    ringHeard: BOOLEAN,  --latched version of EIA RS-232-C Ring Indicator
    ringIndicator: BOOLEAN,
    deviceError: BOOLEAN];
  Parameter: TYPE = RECORD [
    SELECT type: ParameterType FROM
    charLength => [charLength: CharLength],
    clockSource => [clockSource: ClockSource],
    correspondent => [correspondent: Correspondent],
    dataTerminalReady => [dataTerminalReady: BOOLEAN],
    echoing => [echoing: BOOLEAN],
    encodeData => [encodeData: EncodeData],
    flowControl => [flowControl: FlowControl],
    frameTimeout => [frameTimeout: CARDINAL],
    idleState => [idleState: IdleState],
    latchBitClear => [latchBitClearMask: LatchBitClearMask],
    lineSpeed => [lineSpeed: LineSpeed],
    maxAsyncTimeout => [maxAsyncTimeout: CARDINAL],
    parity => [parity: Parity],
    requestToSend => [requestToSend: BOOLEAN],
```

```
    stopBits => [stopBits: StopBits],
    syncChar => [syncChar: SyncChar],
    syncCount => [syncCount: SyncCount],
    ENDCASE];
  ParameterType: TYPE = {
    charLength, clockSource, correspondent, dataTerminalReady, echoing,
    encodeData, flowControl, frameTimeout, idleState, latchBitClear,
    lineSpeed, maxAsyncTimeout, parity, requestToSend, stopBits, syncChar,
    syncCount};
  CharLength: TYPE = RS232CEnvironment.CharLength;
  Correspondent: TYPE = RS232CEnvironment.Correspondent;
  FlowControl: TYPE = RS232CEnvironment.FlowControl;
  LatchBitClearMask: TYPE = DeviceStatus;
  LineSpeed: TYPE = RS232CEnvironment.LineSpeed;
  Parity: TYPE = RS232CEnvironment.Parity;
  StopBits: TYPE = RS232CEnvironment.StopBits;
  SyncCount: TYPE = RS232CEnvironment.SyncCount;
  SyncChar: TYPE = RS232CEnvironment.SyncChar;

  END.  -- RS232C

LOG

Time: 1978  By: Victor Schwartz  Action: Created file
Time: October 1979  By: Victor Schwartz  Action: Pre-Teak LOG entries removed
Time: June 20, 1980  2:51 PM  By: Victor Schwartz  Action: Pre-Amargosa LOG entries removed.
Time: June 20, 1980  2:51 PM  By: Victor Schwartz  Action: Remove Abort and ChannelDoesNotExist.
  Change semantics of Suspend/Restart.
Time: January 20, 1981  12:43 PM  By: Danielson  Action: Add procedure GetLineCount
  to allow client to obtain count of total number of RS232C lines.
  Added deviceError to deviceStatus to allow reporting of terminal device errors.
Time: 16-Jul-81 13:29:25  By: Danielson  Action: Combined RS232C and RS232CManager
Time: 13-Aug-81 15:11:15  By: Danielson  Action: Added flow control, echoing and GetNextLine
Time: 28-Jun-82 13:56:34  By: Danielson  Action: Removed ois references
Time: 21-Jan-85 11:07:13  By: SMA  Action: added encodeData, idleState, maxAsyncTimeout, clockSource.
```

```
-- File RS232CDerived.df; edited by SMA  25-Jan-85 14:45:36

Exports [Huey:OSBU North:Xerox]<APilot>12.0>DF>     CameFrom [Idun]<Int>DF>

  RS232CDerived.df                          31-Oct-85 17:45:35 PST

Exports [Huey:OSBU North:Xerox]<APilot>12.0>RS232C>Public>     CameFrom [Idun]<Int>RS232C>Public>

  *+RS232CIO.bcd!1                          14-Feb-85 18:41:05 PST
  *RS232CIO.symbols!1                       14-Feb-85 18:40:26 PST

Directory [Huey:OSBU North:Xerox]<APilot>12.0>RS232C>Private>     CameFrom [Idun]<Int>RS232C>Private>

  RS232CDerived.cm!1                        25-Jan-85 14:46:05 PST
  RS232CIO.config!1                         25-Sep-84 10:50:54 PDT
  RS232CIO.pack!1                           28-Jun-82 14:21:16 PDT

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>RS232CPrivate.df  Of 31-Oct-85 17:45:39 PST
    Using [DialupImpl.bcd, RS232CDriverA.bcd, RS232CDriver8.bcd]
```

```
-- Copyright (C) 1983, 1984  by Xerox Corporation. All rights reserved.
-- RS232CDLion.df - edited by:
-- RJ     9-Feb-83 12:19:29
-- AF     3-Jun-85 17:46:29
-- DG    23-Sep-83 17:02:17
-- DG    11-Dec-83 11:34:50
-- DG     1-Mar-84 13:51:03; temporarily add DLionInputOutputExtras.* (remove after Klamath)
-- DG    28-Jul-84 14:43:13
-- DG    27-Sep-84 18:00:28
-- DG    26-Nov-84  2:57:47       removed DLionInputOutputExtras and moved DLionInputOutput.bcd to FacesFriends.df.
-- DG    18-Jan-85  0:51:15

Directory [Huey:OSBU North:Xerox]<APilot>12.0>DF>     CameFrom [Idun]<Int>DF>

  RS232CDLion.df                          31-Oct-85 17:45:36 PST

Directory [Huey:OSBU North:Xerox]<APilot>12.0>RS232CDLion>Public>     CameFrom [Idun]<Int>RS232CDLion>Public>

  *+RS232CIOHeadsDLion.bcd!3              29-Jul-85 12:01:42 PDT
  *RS232CIOHeadsDLion.symbols!3           29-Jul-85 12:01:16 PDT

Directory [Huey:OSBU North:Xerox]<APilot>12.0>RS232CDLion>Friends>     CameFrom [Idun]<Int>RS232CDLion>Friends>

  +RS232CDLion.stats!3                    29-Jul-85 12:03:03 PDT

Directory [Huey:OSBU North:Xerox]<APilot>12.0>RS232CDLion>Private>     CameFrom [Idun]<Int>RS232CDLion>Private>

  RS232CDLion.cm!1                        30-Jan-85  5:01:07 PST
  +RS232CDLion.includedBy!3               29-Jul-85 12:10:35 PDT
  +RS232CDLion.includes!3                 29-Jul-85 12:10:33 PDT

  +UnpackedRS232CIOHeadsDLion.bcd!3       29-Jul-85 12:01:14 PDT
  RS232CIOHeadsDLion.config!1             26-Sep-84 11:21:05 PDT
  RS232CIOHeadsDLion.pack!1               26-Nov-84  2:53:09 PST
  +RS232CIOHeadsDLion.map!3               29-Jul-85 12:01:42 PDT

  IOPInterfaceDefs.bcd!1                  14-Feb-85 16:43:05 PST
  IOPInterfaceDefs.mesa!1                 23-Jan-85  1:15:08 PST
  RS232CRS366.bcd!1                       14-Feb-85 16:43:19 PST
  RS232CRS366.mesa!1                      26-Nov-84  0:22:03 PST
  RS232CHeadDLion.bcd!3                   29-Jul-85 11:58:09 PDT
  RS232CHeadDLion.mesa!2                  26-Jul-85 16:10:21 PDT
  RS366HeadDLion.bcd!1                    14-Feb-85 16:44:05 PST
  RS366HeadDLion.mesa!1                   25-Nov-84 23:58:31 PST


Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>RS232CPublic.df  Of 31-Oct-85 17:45:40 PST
    Using [RS232CCorrespondents.bcd, RS232CEnvironment.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>PilotPublic.df  Of 31-Oct-85 17:45:29 PST
    Using [ByteBlt.bcd, Heap.bcd, Process.bcd, Zone.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>PilotFriends.df  Of 31-Oct-85 17:45:24 PST
    Using [ProcessPriorities.bcd, ResidentHeap.bcd, SpecialHeap.bcd,
       SpecialRuntime.bcd, SpecialSpace.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>MesaPublic.df  Of 31-Oct-85 17:45:04 PST
    Using [Environment.bcd, Inline.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>FacesCommon.df  Of 31-Oct-85 17:44:32 PST
    Using [DeviceCleanup.bcd, RS232CFace.bcd, RS366Face.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>FacesFriendsDLion.df  Of 31-Oct-85 17:44:33 PST
    Using [DLionInputOutput.bcd]
```

```
-- Copyright (C) 1982, 1983  by Xerox Corporation. All rights reserved.
-- File: RS232CEnvironment.mesa
-- LastEdited:   13-Feb-85 11:32:42  By: SMA
-- This DEFINITIONS module defines basic, (hopefully) unchanging TYPEs required both
-- by the RS232C channel (RS232C), the RS232C device face (RS232CFace),
-- the RS366 device face (RS366Face), and the dialing software (Dialup).


DIRECTORY
  Environment USING [Byte, Block];

RS232CEnvironment: DEFINITIONS =
  BEGIN
  AutoRecognitionOutcome: TYPE = RECORD [[0..15]];
  CharLength: TYPE = [5..8];
  ClockSource: TYPE = {internal, external};
  Duplexity: TYPE = {full, half};
  CompletionHandle: TYPE [2];
  Correspondent: TYPE = RECORD [[0..255]];
  DialMode: TYPE = {manual, auto};
  EncodeData: TYPE = {nrz, nrzi, fm0, fm1};
  FlowControl: TYPE = MACHINE DEPENDENT RECORD [
    type(0): {none, xOnXOff},
    xOn(1), xOff(2): UNSPECIFIED];
  IdleState: TYPE = {mark, flag};
  LineSpeed: TYPE = {
    bps50, bps75, bps110, bps134p5, bps150, bps300, bps600, bps1200, bps2400,
    bps3600, bps4800, bps7200, bps9600, bps19200, bps28800, bps38400, bps48000,
    bps56000, bps57600};
  LineType: TYPE = {
    bitSynchronous, byteSynchronous, asynchronous, autoRecognition};
  NetAccess: TYPE = {directConn, dialConn};
  nullLineNumber: CARDINAL = LAST[CARDINAL];
  Parity: TYPE = {none, odd, even, one, zero};
  PhysicalRecordHandle: TYPE = LONG POINTER TO PhysicalRecord;
  PhysicalRecord: TYPE = RECORD [header, body, trailer: Environment.Block];
  ReserveType: TYPE = {preemptNever, preemptAlways, preemptInactive};
  RetryCount: TYPE = [0..7];
  StopBits: TYPE = [1..2];
  SyncCount: TYPE = [0..7];
  SyncChar: TYPE = Environment.Byte;

  -- The following types help describe the communication equipment
  --(modems) being used.

  CommParamHandle: TYPE = LONG POINTER TO CommParamObject;
  CommParamObject: TYPE = MACHINE DEPENDENT RECORD [
    duplex(0): Duplexity,
    lineType(1): LineType,
    lineSpeed(2): LineSpeed,
    accessDetail(3): SELECT netAccess(3): NetAccess FROM
      directConn => NULL,
      dialConn => [
        dialMode(4): DialMode,
        dialerNumber(5): CARDINAL,
        retryCount(6): RetryCount],
      ENDCASE];

  END. -- RS232CEnvironment

LOG

Time: January 22, 1980  10:37 AM  By: Victor Schwartz  Action: Created file
Time: August 4, 1980  3:36 PM  By: Victor Schwartz  Action: Change CompletionHandle
  to an EXPORTed type, opaque to the client.
Time: 16-Jul-81 13:27:28  By: Bill Danielson  Action: Additions for combined RS232C
  and RS232CManager.
Time: 13-Aug-81 15:04:41  By: Bill Danielson  Action: Added flow control and echoing for
  872/873 box use
Time: 28-Jun-82 13:56:55  By: Bill Danielson  Action: Removed ois references
Time:  7-Apr-83 10:15:26  By: AOF  Action: Merged "Extras"
Time:  2-May-83  8:21:26  By: SMA  Action: Added line speeds above bps19200. (AR 12334)
Time:  2-Nov-84 10:47:48  By: SMA  Action: Made CommParamHandle LONG.
Time: 13-Feb-85 11:32:49  By: SMA  Action: Added ClockSource, EncodeData, IdleState.
```

```
-- File: RS232Chat.config - last edit:
-- Hodges.pa           18-Jul-85  9:39:36
-- RS232XChat.config
-- Yamamoto      21-May-84  9:41:43

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

RS232Chat: CONFIGURATION
  IMPORTS
    ByteBlt, DeviceCleanup,
    DLionInputOutput, FormSW, Heap, MStream,
    Process, Profile, Put, ResidentHeap, Runtime,
    SpecialHeap, SpecialRuntime, SpecialSpace, Stream,
    String, TIP, Time,
    Tool, TTY, TTYSW, UserTerminal, --Version,--
    WindowFont
  CONTROL RS232ChatImpl = {
    RS232CIO;
    RS232CIOHeadsDLion;
    RS232ChatImpl;
    }...
```

```
-- File: RS232Chat.df - last edit:
-- Hodges.pa           19-Jul-85 11:50:20
-- RS232CHacks.df
-- Yamamoto      21-May-84  9:54:05

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

Exports [Goofy:OSBU North:Xerox]<Hacks>12.0>DF>     ReleaseAs [Goofy:OSBU North:Xerox]<Hacks>

  RS232Chat.df                              16-Apr-86 14:51:33 PST

Exports [Goofy:OSBU North:Xerox]<Hacks>12.0>Tools>     ReleaseAs [Goofy:OSBU North:Xerox]<Hacks>

  +RS232Chat.bcd!4                          12-Aug-85  0:54:55 PDT
  +RS232ChatDove.bcd!1                      14-Apr-86  9:20:27 PST

Exports [Goofy:OSBU North:Xerox]<Hacks>12.0>Source>RS232Chat>     ReleaseAs [Goofy:OSBU North:Xerox]<Hacks>

  RS232ChatImpl.bcd!4                       12-Aug-85  0:50:47 PDT

Directory [Goofy:OSBU North:Xerox]<Hacks>12.0>Doc>     ReleaseAs [Goofy:OSBU North:Xerox]<Hacks>

  +RS232Chat.doc!4                          15-Apr-86 17:59:08 PST

Directory [Goofy:OSBU North:Xerox]<Hacks>12.0>Source>RS232Chat>     ReleaseAs [Goofy:OSBU North:Xerox]<Hacks>

  RS232Chat.config!2                        18-Jul-85  9:39:36 PDT
  RS232ChatDove.config!1                    14-Apr-86  9:16:11 PST
  RS232ChatImpl.mesa!4                      12-Aug-85  0:47:35 PDT

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>ComSoftPublic.df Of >
    Using [Ascii.bcd, Format.bcd, String.bcd, Time.bcd, TTY.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>FacesFriendsDove.df Of #
    Using [DoveInputOutput.bcd]

Imports [Ramrod:OSBU North:Xerox]<AMesa>12.0>DF>FileSystemPublic.df Of >
    Using [MFile.bcd, MStream.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>MesaPublic.df Of >
    Using [Environment.bcd]

Imports [RamRod:OSBU North:Xerox]<AMesa>12.0>DF>Pilot.df Of >
    Using [RS232CEnvironment.bcd, System.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>PilotFriends.df Of #
    Using [DebuggerSwap.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>PilotPublic.df Of >
    Using [ByteBlt.bcd, Heap.bcd, Process.bcd, Runtime.bcd, Stream.bcd,
       UserTerminal.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>RS232CDerived.df Of >
    Using [RS232CIO.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>RS232CDLion.df Of >
    Using [RS232CIOHeadsDLion.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>RS232CDove.df Of >
    Using [RS232CIOHeadsDove.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>RS232CPublic.df Of >
    Using [RS232C.bcd, RS232CCorrespondents.bcd]

Imports [Ramrod:OSBU North:Xerox]<AMesa>12.0>DF>TajoPublic.df Of >
    Using [FormSW.bcd, Profile.bcd, Put.bcd, TIP.bcd, Tool.bcd, ToolWindow.bcd,
       TTYSW.bcd, UserInput.bcd, Version.bcd, Window.bcd, WindowFont.bcd]
```

```
-- File: RS232ChatDove.config:
-- DSacks.es            14-Apr-86  9:03:43

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

RS232ChatDove: CONFIGURATION
  IMPORTS
    ByteBlt,DeviceCleanup, DebuggerSwap,
    DoveInputOutput, FormSW, Heap, MStream,
    Process, Profile, Put, ResidentHeap, Runtime,
    SpecialHeap, SpecialRuntime, Stream,
    String, TIP, Time,
    Tool, TTY, TTYSW, UserTerminal, --Version,--
    WindowFont
  CONTROL RS232ChatImpl = {
    RS232CIO;
    RS232CIOHeadsDove;
    RS232ChatImpl;
    }...
```

```
-- File: RS232ChatImpl.mesa - last edit:
-- Trow.pa        2-Feb-88 23:55:40
-- Hodges.pa     12-Aug-85  0:47:35
-- RS232XChatImpl.mesa
-- Create by FormSWLayoutTool on 17-May-84 22:42
-- Yamamoto      21-May-84  9:56:28

-- Copyright (c) 1984, 1985, 1988 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii,
  Environment,
  Format,
  FormSW,
  Heap,
  MStream,
  Process,
  Profile,
  Put,
  RS232C,
  RS232CCorrespondents,
  Runtime,
  Stream,
  String,
  Time,
  TIP,
  Tool,
  ToolWindow,
  TTY,
  TTYSW,
  UserInput,
  UserTerminal,
  Version,
  Window,
  WindowFont;

RS232ChatImpl: MONITOR
  IMPORTS
    FormSW, Heap, MStream, Process,
    Profile, Put, RS232C, Runtime, Stream, String, Time, Tool,
    TIP, TTY, TTYSW, UserTerminal, -- Version,-- WindowFont - {

  DataHandle: TYPE = LONG POINTER TO Data;
  Data: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    ttySW(4): Window.Handle ← NIL,
    lineSpeed(6): RS232C.LineSpeed ← bps9600,
    parity(7): RS232C.Parity ← none,
    stopBits(8): INTEGER ← 1,
    duplex(9): RS232C.Duplexity ← full,
    charLength(10): INTEGER ← 8,
    cH(11): RS232C.ChannelHandle ← TRASH,
    get(13): PROCESS ← NIL,
    put(14): PROCESS ← NIL,
    DisplayDataProcess(15): PROCESS ← NIL,
    connected(16): BOOLEAN ← FALSE,
    tty(17): TTY.Handle ← TTY.nullHandle,
    UserInputProcess(19): PROCESS ← NIL,
    flowControl(20): MyFlowControlStateIndicator ← off,
    mode(21): MyModeStateIndicator ← normal,
    xON(22): UNSPECIFIED ← DC1,
    xOFF(23): UNSPECIFIED ← DC3];

  MyFlowControlStateIndicator: TYPE = {on, off, echo};
  MyModeStateIndicator: TYPE = {normal, raw};
  MyTransitionIndicator: TYPE = {goingInactive, goingActive};

  myTransitionIndicator: MyTransitionIndicator ← goingActive;

  DC1: UNSPECIFIED = 11H;
  DC3: UNSPECIFIED = 13H;

  QueueHandle: TYPE =  LONG POINTER TO QueueObject;
  QueueObject: TYPE = RECORD[
    next: QueueHandle,
    lostChars: BOOLEAN,
    cH: RS232C.CompletionHandle,
    string: LONG STRING];

  debug: Stream.Handle ← NIL;

  StuffOnQueue: CONDITION;
  aborting: BOOLEAN ← FALSE;

  FormItems: TYPE = {connect, disconnect, mode, baud, parity, stop, duplex, charWidth, flowControl};

  data: DataHandle ← NIL;
  wh: Window.Handle ← NIL;
  zone: UNCOUNTED ZONE ← Heap.Create[
    initial: 15, increment: 6, largeNodeThreshold: 512];

  Msg: Format.StringProc = {Put.Text[data.msgSW, s]};

  tip: PUBLIC TIP.Table ← NIL;

  herald: LONG STRING ← NIL;
```

```
heap: PUBLIC UNCOUNTED ZONE ← Heap.systemZone;

Connect: FormSW.ProcType = {
  commParamObject: RS232C.CommParamObject ← [
    duplex: data.duplex, lineType: asynchronous,
    lineSpeed: data.lineSpeed, accessDetail: directConn[]];

  IF data.connected THEN
      Msg["Already connected.\n"L]
  ELSE
    {
      Msg["Connecting ... "L];
      aborting ← FALSE;
      data.cH ← RS232C.Create[0, @commParamObject, preemptAlways, preemptNever];
      data.connected ← TRUE;
      data.cH.SetParameter[[charLength[data.charLength]]];
      data.cH.SetParameter[[correspondent[RS232CCorrespondents.ttyHost]]];
      data.cH.SetParameter[[dataTerminalReady[TRUE]]];
      data.cH.SetParameter[[frameTimeout[5]]];
      data.cH.SetParameter[[lineSpeed[data.lineSpeed]]];
      data.cH.SetParameter[[parity[data.parity]]];
      data.cH.SetParameter[[requestToSend[TRUE]]];
      data.cH.SetParameter[[stopBits[data.stopBits]]];
      ChangeFlowControl[];
      data.get ← FORK GetData[];
      data.put ← FORK PutData[];
      data.DisplayDataProcess ← FORK TTYToDisplay[];
      data.UserInputProcess ← FORK KeyToTTY[];
      IF Profile.debugging THEN debug ← MStream.Log[
        "TTYDebug.log"L, [Release, NIL]];
      Msg["Connection open to RS232C port.\n"L];
    };
  };

Release: MStream.PleaseReleaseProc = {
  MStream.SetLogReadLength[stream, stream.GetPosition[]];
  RETURN[no]};

Disconnect: FormSW.ProcType = {
  IF↓data.connected THEN
      Msg["Not connected.\n"L]
  ELSE
    {
      Msg["Disconnecting ... "L];
      aborting ← TRUE;
      data.cH.SetParameter[[dataTerminalReady[FALSE]]];
      data.cH.Suspend[all];
      Process.Abort[data.get];
      JOIN data.get;
      data.get ← NIL;
      Process.Abort[data.put];
      JOIN data.put;
      data.put ← NIL;
      Process.Abort[data.DisplayDataProcess];
      JOIN data.DisplayDataProcess;
      data.DisplayDataProcess ← NIL;
      Process.Abort[data.UserInputProcess];
      JOIN data.UserInputProcess;
      data.UserInputProcess ← NIL;
      data.cH.Delete[];
      IF debug # NIL THEN {debug.Delete[]; debug ← NIL};
      data.connected ← FALSE;
      IF myTransitionIndicator # goingInactive THEN
          Msg["Disconnected.\n"L];
    };
  };

ChangeBaud: FormSW.EnumeratedNotifyProcType = {
  IF data.connected THEN
    data.cH.SetParameter[[lineSpeed[data.lineSpeed]]];
  };

ChangeParity: FormSW.EnumeratedNotifyProcType = {
  IF data.connected THEN
    data.cH.SetParameter[[parity[data.parity]]];
  };

ChangeStopBits: FormSW.EnumeratedNotifyProcType = {
  IF data.connected THEN
    data.cH.SetParameter[[stopBits[data.stopBits]]];
  };

ChangeDuplexity: FormSW.EnumeratedNotifyProcType = {
  IF data.connected THEN
    Msg["Must disconnect to change duplexity.\n"L];
  };

ChangeCharLength: FormSW.EnumeratedNotifyProcType = {
  IF data.connected THEN
    data.cH.SetParameter[[charLength[data.charLength]]];
  };

ChangeFlowControl: FormSW.EnumeratedNotifyProcType = {
  IF data.connected THEN
    SELECT data.flowControl FROM
      on => data.cH.SetParameter[[flowControl[[type: xOnXOff, xOn: DC1, xOff: DC3]]]]
      off => data.cH.SetParameter[[flowControl[[type: none, xOn: 0, xOff: 0]]]];
```

```
        wait => {
          data.cH.SetParameter[[flowControl[[type: none, xOn: 0, xOff: 0]]]];
          -- whatever else needs to be done for echo mode};
        ENDCASE;
    };

  ChangeMode: FormSW.EnumeratedNotifyProcType = {
    };


  ClientTransition: ToolWindow.TransitionProcType = {
    SELECT TRUE FROM
      old = inactive => {
        myTransitionIndicator ← goingActive;
        IF data = NIL THEN data ← zone.NEW[Data ← []];
        };
      new = inactive =>
        IF data # NIL THEN {
          myTransitionIndicator ← goingInactive;
          IF data.connected THEN  Disconnect[];
          zone.FREE[@data]};
      ENDCASE;
    };

<< Process name is "get" >>

  GetData: PROCEDURE  = {  -- forked as Process
    buffer: QueueHandle ← NIL;
    status: RS232C.TransferStatus;
    byteCount: CARDINAL;
    rec: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock, body: Environment.nullBlock,
      trailer: Environment.nullBlock];
    Process.SetPriority[5];
    BEGIN
      ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
      DO
        ENABLE ABORTED => EXIT;
        buffer ← GetFromQueue[@freelist];
        rec.body ← [LOOPHOLE[@buffer.string.text], 0, buffer.string.maxlength];
        buffer.cH ← data.cH.Get[@rec];
        PutToQueue[@outlist, buffer];
      ENDLOOP;
      EXITS quit => NULL;
    END;
    WHILE outlist # NIL DO
      buffer ← GetFromQueue[@outlist, FALSE];
      IF buffer = NIL THEN EXIT;
      [byteCount, status] ← data.cH.TransferWait[buffer.cH
                                                 ! ABORTED => NULL];
      InitBuffer[buffer];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    };

<< Process name is "put" >>

  PutData: PROCEDURE  = {  -- forked as Process
    status: RS232C.TransferStatus;
    byteCount: CARDINAL;
    deviceStatus: RS232C.DeviceStatus;
    buffer: QueueHandle ← NIL;
    Process.SetPriority[5];
    BEGIN  -- forked as Process
      ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
      DO
        ENABLE ABORTED => EXIT;
        buffer ← GetFromQueue[@outlist];
        [byteCount, status] ← data.cH.TransferWait[buffer.cH];
        buffer.string.length ← byteCount;
        IF status = aborted THEN EXIT;
        deviceStatus ← data.cH.GetStatus[];
        IF deviceStatus.statusAborted THEN EXIT;
        buffer.lostChars ← deviceStatus.dataLost OR deviceStatus.deviceError
          OR status # success;
        IF buffer.lostChars AND Profile.debugging THEN
          NoteError[deviceStatus, status];
        PutToQueue[@filledlist, buffer];
      ENDLOOP;
      EXITS quit => NULL;
    END; -- of enabled
    WHILE filledlist # NIL DO
      buffer ← GetFromQueue[@filledlist, FALSE];
      IF buffer = NIL THEN EXIT;
      InitBuffer[buffer];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    };

  NoteError: PROC[
    deviceStatus: RS232C.DeviceStatus, status: RS232C.TransferStatus] = {
    debug.PutString["Device status: ["L];
    PutBoolean[deviceStatus.statusAborted, "statusAborted"L, TRUE];
    PutBoolean[deviceStatus.dataLost, "dataLost"L, TRUE];
    PutBoolean[deviceStatus.breakDetected, "breakDetected"L, TRUE];
    PutBoolean[deviceStatus.clearToSend, "clearToSend"L, TRUE];
    PutBoolean[deviceStatus.dataSetReady, "dataSetReady"L, TRUE];
```

```
    PutBoolean[deviceStatus.carrierDetect, "carrierDetect"L, TRUE];
    PutBoolean[deviceStatus.ringHeard, "ringHeard"L, TRUE];
    PutBoolean[deviceStatus.ringIndicator, "ringIndicator"L, TRUE];
    PutBoolean[deviceStatus.deviceError, "deviceError"L, FALSE];
    debug.PutString["]\n"L];
    debug.PutString["Transfer status: "L];
    debug.PutString[SELECT status FROM
      success => "success"L,
      dataLost => "dataLost"L,
      deviceError => "deviceError"L,
      frameTimeout => "frameTimeout"L,
      checksumError => "checksumError"L,
      parityError => "parityError"L,
      asynchFramingError => "asynchFramingError"L,
      invalidChar => "invalidChar"L,
      invalidFrame => "invalidFrame"L,
      aborted => "aborted"L,
      ENDCASE => "disaster"L];
    debug.PutString["\n"L];
    };

  PutBoolean: PROC[b: BOOLEAN, s: LONG STRING, comma: BOOLEAN] = {
    debug.PutString[s];
    debug.PutString[": "L];
    debug.PutString[IF b THEN "TRUE"L ELSE "FALSE"L];
    IF comma THEN debug.PutString[", "L];
    };

  bufferMax: CARDINAL = 128;

  Init: PROCEDURE = {
    buffer: QueueHandle ← NIL;
    firstTime: BOOLEAN ← TRUE;
    fileName: STRING = "RS232Chat.TIP"L;
    contents: LONG STRING =
"-- RS232Chat.TIP; of 7-Jun-85 16:41:11
-- Created by System

SELECT TRIGGER FROM
  ENDCASE...
"L;
    tip ← TIP.CreateTable[file: fileName, contents: contents !
      TIP.InvalidTable =>
        IF type = badSyntax THEN {
          IF firstTime THEN {firstTime ← FALSE; Flash[]; RESUME}
          ELSE CONTINUE}];
    IF tip  # NIL THEN
      [] ← TIP.PushLocal[push: tip , onto: TIP.globalTable[ttySW]];

    FOR i: CARDINAL IN [0..10) DO
      buffer ← zone.NEW[
        QueueObject ← [
        next: NIL, lostChars: FALSE, cH: TRASH,
        string: zone.NEW[StringBody [bufferMax]]]];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    Process.SetTimeout[@StuffOnQueue, 1];
    Process.EnableAborts[@StuffOnQueue];
    herald ← heap.NEW[StringBody [40]];
    String.AppendString[herald, "RS232Chat v2.1 of "L];
    Time.Append[herald, Time.Unpack[Runtime.GetBcdTime[]]];
--    String.AppendString[herald, " running on Pilot "L];
--    Version.Append[herald];
    herald.length ← herald.length - 3; -- gun the seconds

    wh ← Tool.Create[
      makeSWsProc: MakeSWs, initialState: default, initialBox:[[0,30],[512,512]],
      clientTransition: ClientTransition, name: herald,
      cmSection: "RS232Chat"L];
    };

<< Process name is "UserInputProcess" >>

  KeyToTTY: PROC =
  {
     ch: CHARACTER;
     chs: PACKED ARRAY [0..2) OF Environment.Byte;
     ptr: LONG POINTER TO UNSPECIFIED ← @chs;
     physRecord: RS232C.PhysicalRecord ← [
        header: Environment.nullBlock,
        body: [
          startIndex: 0, stopIndexPlusOne: 1,
          blockPointer: ptr],
        trailer: Environment.nullBlock];

     Process.SetPriority[5];
     BEGIN
       ENABLE {UNWIND => NULL; ABORTED => GOTO quit};
       DO
         ENABLE ABORTED => EXIT;
         ch ← data.tty.GetChar[];
         chs[0] ← LOOPHOLE[ch];
         [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
       ENDLOOP;
       EXITS quit => NULL;
     END;
  };
```

```
<< Process name is "DisplayDataProcess" >>

  TTYToDisplay: PROCEDURE = {
    buffer: QueueHandle ← NIL;
    s: LONG STRING ← NIL;
    i, j: CARDINAL ← 0;

    Process.SetPriority[Process.priorityForeground];

    DO
      ENABLE ABORTED => EXIT;
      buffer ← GetBuffer[!
        ABORTED =>  EXIT; ANY => LOOP];
      IF buffer.lostChars THEN
         Msg["Characters lost.\n"L];
      j ← 0;
      s ← buffer.string;

      FOR i IN [0..s.length) DO
        IF data.mode = normal THEN       -- process certain cntl chars specially
          {
            SELECT  s[i] FROM
              Ascii.NUL,
              Ascii.LF,
              Ascii.DEL => NULL;

              Ascii.BS =>
                {
                  IF j > 0 THEN
                    {
                      s.length ← j;
                      data.tty.PutString[s];
                      s.length ← j ← 0;
                    };
                  data.tty.RemoveCharacter[];
                };
              ENDCASE => {s[j] ← s[i]; j ← j + 1};
          }
        ELSE  -- mode is raw, send all chars through regardless...
          {
            s[j] ← s[i];
            j ← j + 1;
          };
      ENDLOOP;

      IF j > 0 THEN
        {
          s.length ← j;
          data.tty.PutString[s];
        };

      ReturnBuffer[buffer];
      buffer ← NIL;
    ENDLOOP;
    IF buffer # NIL THEN ReturnBuffer[buffer];
    };  -- TTYToDisplay

  freelist, outlist, filledlist: QueueHandle ← NIL;

  GetBuffer: PROC RETURNS [buffer: QueueHandle] = {
    ENABLE UNWIND => NULL;
    buffer ← GetFromQueue[@filledlist];
    };  -- GetBuffer

  ReturnBuffer: PUBLIC PROC [buffer: QueueHandle] = {
    InitBuffer[buffer]; PutToQueue[@freelist, buffer]};

  InitBuffer: PROC [buffer: QueueHandle] = {
    buffer.lostChars ← FALSE; buffer.next ← NIL; };

  PutToQueue: ENTRY PROC [q: LONG POINTER TO QueueHandle, buf: QueueHandle] = {
    i: LONG POINTER TO QueueHandle ← NIL;
    FOR i ← q, @i.next UNTIL i↑ = NIL DO ENDLOOP;
    i↑ ← buf;
    buf.next ← NIL;
    BROADCAST StuffOnQueue;
    };

  GetFromQueue: ENTRY PROC [q: LONG POINTER TO QueueHandle, wait: BOOLEAN ← TRUE]
    RETURNS [buf: QueueHandle] = {
    ENABLE UNWIND => NULL;
    IF wait THEN WHILE q↑ = NIL DO
      WAIT StuffOnQueue;
      IF aborting THEN ERROR ABORTED;
      ENDLOOP;
    buf ← q↑;
    IF buf # NIL THEN q↑ ← buf.next};

  MakeSWs: Tool.MakeSWsProc = {
    logName: STRING ← [40];
    data.msgSW ← Tool.MakeMsgSW[window: window];
    data.formSW ← Tool.MakeFormSW[
      window: window, formProc: MakeForm];
    Tool.UnusedLogName[unused: logName, root: "RS232Chat.log"L];
    data.ttySW ← Tool.MakeTTYSW[window: window, name: logName];
    data.tty ← TTYSW.GetTTYHandle[data.ttySW];
```

```
    IF tip # NIL THEN [] ← TIP.SetTable[data.ttySW, tip];
    };

  MakeForm: FormSW.ClientItemsProcType = {
    OPEN FormSW;
    nItems: CARDINAL = FormItems.LAST.ORD + 1;
    baudArray: ARRAY[0..6) OF Enumerated ← [
      ["300"L, RS232C.LineSpeed.bps300.ORD], ["600"L, RS232C.LineSpeed.bps600.ORD],
      ["1200"L, RS232C.LineSpeed.bps1200.ORD], ["2400"L, RS232C.LineSpeed.bps2400.ORD],
      ["4800"L, RS232C.LineSpeed.bps4800.ORD], ["9600"L, RS232C.LineSpeed.bps9600.ORD]];

    parityArray: ARRAY[0..3) OF Enumerated ← [
      ["none"L, 0], ["odd"L, 1],
      ["even"L, 2]];

    stop: ARRAY[0..2) OF Enumerated ← [
      ["one"L, 1], ["two"L, 2]];

    duplexArray: ARRAY[0..2) OF Enumerated ← [
      ["full"L, 0], ["half"L, 1]];

    bits: ARRAY[0..4) OF Enumerated ← [
      ["5"L, 5], ["6"L, 6],
      ["7"L, 7], ["8"L, 8]];

    flowControlArray: ARRAY[0..3) OF Enumerated ← [
      ["FlowCntl ON"L, 0], ["FlowCntl OFF"L, 1], ["FlowCntl ECHO"L, 2]];

    modeArray: ARRAY[0..2) OF Enumerated ← [
      ["Normal"L, 0], ["Raw"L, 1]];

    items ← AllocateItemDescriptor[nItems];

<<--- Line 0 items --->>

    items[FormItems.connect.ORD] ← CommandItem[
      tag: "Connect"L, place: [CharPos[0], line0], proc: Connect];

    items[FormItems.disconnect.ORD] ← CommandItem[
      tag: "Disconnect"L, place: [CharPos[20], line0], proc: Disconnect];

    items[FormItems.mode.ORD] ← EnumeratedItem[
      tag: "Mode"L, place: [CharPos[40], line0], proc: ChangeMode, choices: DESCRIPTOR[modeArray],  value: @data.mode];

<<--- Line 1 items --->>

    items[FormItems.baud.ORD] ← EnumeratedItem[
      tag: "Baud"L, place: [CharPos[0], line1], proc: ChangeBaud, choices: DESCRIPTOR[baudArray], value: @data.lineSpeed];

    items[FormItems.parity.ORD] ← EnumeratedItem[
      tag: "Parity"L, place: [CharPos[20], line1], proc: ChangeParity, choices: DESCRIPTOR[parityArray], value: @data.parity];

    items[FormItems.stop.ORD] ← EnumeratedItem[
      tag: "Stop"L, place: [CharPos[40], line1], proc: ChangeStopBits, choices: DESCRIPTOR[stop], value: @data.stopBits];

<<--- Line 2 items --->>

    items[FormItems.duplex.ORD] ← EnumeratedItem[
      tag: "Duplex"L, place: [CharPos[0], line2], proc: ChangeDuplexity, choices: DESCRIPTOR[duplexArray], value: @data.duplex];

    items[FormItems.charWidth.ORD] ← EnumeratedItem[
      tag: "CharWidth"L, place: [CharPos[20], line2], proc: ChangeCharLength, choices: DESCRIPTOR[bits], value: @data.charLength];

    items[FormItems.flowControl.ORD] ← EnumeratedItem[
      tag: "FlowControl"L, place: [CharPos[40], line2], proc: ChangeFlowControl, choices: DESCRIPTOR[flowControlArray], value:
@data.flowControl];

<<--- Line 3 items --->>
    -- none --

    Msg["Disconnected.\n"L];

    RETURN[items: items, freeDesc: TRUE];
    };

  Flash: PUBLIC PROCEDURE =
    BEGIN
    ENABLE ABORTED => CONTINUE;
    UserTerminal.BlinkDisplay[];
    UserTerminal.Beep[262, 75];  -- c
    UserTerminal.Beep[392, 75];  -- g
    UserTerminal.Beep[659, 75];  -- e
    END;

  charWidth: CARDINAL ← WindowFont.CharWidth['0];
  CharPos: PROC[char: CARDINAL] RETURNS [x: INTEGER] = {
    x ← charWidth * char};


  -- Mainline code

  Init[];  -- this gets string out of global frame

  }...
```

```
-- File RS232CPublic.df; edited by SMA 13-Feb-85 11:27:39

Exports [Huey:OSBU North:Xerox]<APilot>12.0>DF>     CameFrom [Idun]<Int>DF>

  RS232CPublic.df                          31-Oct-85 17:45:40 PST

Exports [Huey:OSBU North:Xerox]<APilot>12.0>RS232C>Public>     CameFrom [Idun]<Int>RS232C>Public>

  *+Dialup.bcd!1                           13-Feb-85 11:34:00 PST
  *Dialup.mesa!1                           21-Jan-85 11:08:26 PST
  *+RS232C.bcd!1                           13-Feb-85 11:33:55 PST
  *RS232C.mesa!1                           21-Jan-85 11:12:36 PST
  *+RS232CControl.bcd!1                    21-Jan-85 11:12:53 PST
  *RS232CControl.mesa!1                     7-Dec-81 16:55:12 PST
  *+RS232CCorrespondents.bcd!1             13-Feb-85 11:33:52 PST
  *RS232CCorrespondents.mesa!1             14-Jul-83  9:50:09 PDT
  *+RS232CEnvironment.bcd!1                13-Feb-85 11:33:44 PST
  *RS232CEnvironment.mesa!1                13-Feb-85 11:32:58 PST

Directory [Huey:OSBU North:Xerox]<APilot>12.0>RS232C>Private>     CameFrom [Idun]<Int>RS232C>Private>

  RS232CPublic.cm!1                        30-Jun-83 16:30:30 PDT

Imports [Huey:OSBU North:Xerox]<APilot>12.0>DF>MesaPublic.df  Of 31-Oct-85 17:45:04 PST
    Using [Environment.bcd]
```

```
-- File: RS232XChat.config - last edit:
-- Hodges.pa            14-Nov-85 15:01:06
-- RS232XChat.config
-- Yamamoto       7-May-86 17:37:37

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

RS232XChat: CONFIGURATION
  IMPORTS
    Atom, ByteBlt, Caret, CmFile, Context, Cursor, DeviceCleanup,
    Display, DLionInputOutput, Exec, Format, FormSW,
    HeraldWindow, Heap, MFile, MStream,
    Process, Profile, Put, ResidentHeap, Runtime, Selection,
    SpecialHeap, SpecialRuntime, SpecialSpace, Stream, String, StringLookUp,
    TIP, Token,
    Tool, ToolFont, ToolWindow, UserInput, UserTerminal, Window, WindowFont
  CONTROL RS232XModem, Emu, RS232XChatImpl = {

    RS232XModem;
    Emu;
    RS232CIO;
    RS232CIOHeadsDLion;
    RS232XChatImpl;
    XModemConfig;
    }...
```

```
-- File: RS232XChat.df - last edit:
-- DSacks.es            29-Apr-86 10:18:41
-- BGY                  14-May-86 16:17:55

-- Copyright (C) 1984, 1985, 1986 by Xerox Corporation. All rights reserved.

Exports [Goofy:OSBU North:Xerox]<Hacks>12.0>DF>     ReleaseAs [A]<B>

  RS232XChat.df                              7-Jan-88 14:27:44 PST

Exports [Goofy:OSBU North:Xerox]<Hacks>12.0>Tools>     ReleaseAs [A]<B>

  +RS232XChat.bcd!8                          8-Sep-87 10:39:56 PDT
  +RS232XChatDove.bcd!6                      7-Jan-88 14:25:33 PST

Directory [Goofy:OSBU North:Xerox]<Hacks>12.0>Doc>     ReleaseAs [A]<B>

  +RS232XChat.doc!6                         15-May-86 12:36:45 PDT

Exports [Goofy:OSBU North:Xerox]<Hacks>12.0>Source>RS232XChat>     ReleaseAs [A]<B>

  RS232XChatImpl.bcd!6                      16-Jul-87 10:54:09 PDT
  +RSXChat.by!3                             14-May-86 17:13:48 PDT
  +RSXChatOptions.by!1                      14-May-86 16:57:33 PDT

Directory [Goofy:OSBU North:Xerox]<Hacks>12.0>Source>RS232XChat>     ReleaseAs [A]<B>

  RS232XChat.config!3                        7-May-86 17:43:14 PDT
  RS232XChatDove.config!3                    7-Jan-88 14:25:21 PST
  RS232XChatImpl.mesa!5                     16-Jul-87 10:54:01 PDT

  RS232XChatOps.mesa!2                      31-Mar-87 11:46:08 PST
  RS232XChatOps.bcd!2                       31-Mar-87 11:52:09 PST

  RS232XModem.mesa!1                        15-May-86 11:35:26 PDT
  RS232XModem.bcd!2                         31-Mar-87 11:56:05 PST

Imports [Goofy:OSBU North:Xerox]<Hacks>12.0>DF>XModem.df Of >
    Using [XModem.bcd, XModemConfig.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>ComSoftPublic.df Of >
    Using [Ascii.bcd, Format.bcd, String.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>MesaPublic.df Of >
    Using [Environment.bcd, Inline.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>PilotPublic.df Of >
    Using [Heap.bcd, Process.bcd, Stream.bcd, UserTerminal.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>RS232CDerived.df Of >
    Using [RS232CIO.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>RS232CDLion.df Of >
    Using [RS232CIOHeadsDLion.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>RS232CPublic.df Of >
    Using [RS232C.bcd, RS232CCorrespondents.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>FacesFriendsDove.df Of #
    Using [DoveInputOutput.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>PilotFriends.df Of #
    Using [DebuggerSwap.bcd]

Imports [Huey:OSBU North:Xerox]<APilot>12.2>DF>RS232CDove.df Of >
    Using [RS232CIOHeadsDove.bcd]

Imports [Ramrod:OSBU North:Xerox]<AMesa>12.0>DF>TajoPublic.df Of >
    Using [CmFile.bcd, Cursor.bcd, Exec.bcd, FormSW.bcd, HeraldWindow.bcd,
       Profile.bcd, Put.bcd, StringLookUp.bcd, TIP.bcd, Token.bcd,
       Tool.bcd, ToolWindow.bcd, UserInput.bcd, Window.bcd, WindowFont.bcd]

Imports [RamRod:OSBU North:Xerox]<AMesa>12.0>DF>Pilot.df Of >
    Using [RS232CEnvironment.bcd]

Imports [Ramrod:OSBU North:Xerox]<AMesa>12.0>DF>Emulator.df Of #
    Using [Emu.bcd, Emulator.bcd, EmulatorOps.bcd]

Imports [Ramrod:OSBU North:Xerox]<AMesa>12.0>DF>FileSystemPublic.df Of >
    Using [MFile.bcd, MStream.bcd]
```

```
-- File: RS232XChatDove.config - last edit:
-- JAV                    7-Jan-88 14:25:21
-- DSacks.es             26-Apr-86 15:12:05
-- BGY                    14-May-86 16:15:29
-- Copyright (C) 1984, 1985, 1988 by Xerox Corporation. All rights reserved.

RS232XChatDove: CONFIGURATION
  IMPORTS
    Atom, ByteBlt, Caret, CmFile, Context, Cursor,
    Exec, DeviceCleanup,
    Display, DoveInputOutput, Format, FormSW, Heap, HeraldWindow, MFile, MStream,
    Process, Profile, Put, ResidentHeap, Runtime, Selection,
    SpecialHeap, SpecialRuntime, SpecialSpace, Stream, String, StringLookUp,
    TIP, Token,
    Tool, ToolFont, ToolWindow, UserInput, UserTerminal, Window, WindowFont
  CONTROL RS232XModem, Emu, RS232XChatImpl = {
    RS232XModem;
    Emu;
    RS232CIO;
    RS232CIOHeadsDove;
    RS232XChatImpl;
    XModemConfig;
    }...
```

```
-- File: RS232XChatImpl.mesa - last edit:
-- JAV                 16-Jul-87 10:54:01
-- JDH          15-Nov-85 11:05:28
-- BGY          15-May-86 11:27:23

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii USING [DEL, NUL],
  CmFile USING [
    Close, Error, FindSection, Handle, NextValue, TableError, UserDotCmOpen],
  Dialup USING [
    Number, Outcome, pause],
  DialupExtras USING [DialExtra],
  Emulator USING [
    CR, Create, EmuSWType, PutChar, PutString, RefreshHint, SetEmulator,
    SetRefresh, SetRemote, StartLog, StopLog, table, TerminalType],
  Environment USING [nullBlock],
  Format USING [StringProc],
  FormSW USING [
    AllocateItemDescriptor, BooleanItem, ClientItemsProcType, CommandItem,
    Enumerated, EnumeratedItem, EnumeratedNotifyProcType, line0, line1, line2,
    line3, line4, NotifyProcType, Options, ProcType, StringItem],
  Heap USING [Create],
  Inline USING [BITAND],
  MStream USING [Handle, Log, PleaseReleaseProc, SetLogReadLength],
  Process USING [
    Abort, Detach, EnableAborts, GetCurrent, GetPriority, MsecToTicks, Priority,
    priorityForeground, priorityNormal, SetPriority, SetTimeout],
  Profile USING [debugging],
  Put USING [Text],
  RS232C USING [
    ChannelInUse, ChannelSuspended, CommParamObject, Create, Delete,
    DeviceStatus, Duplexity, FlowControl, Get,
    GetStatus, LineSpeed, Parity, PhysicalRecord, Put, SetParameter, Suspend,
    TransferStatus, TransferWait, TransmitNow],
  RS232CCorrespondents USING [ttyHost],
  RS232XChatOps USING [
    CreateOptionsWindow, CreateXModemWindow, Data, DataHandle, DC1, DC3,
    FlowControl, MyTransitionIndicator, QueueHandle, QueueObject],
  Stream USING [Delete, GetPosition, Handle, PutString],
  StringLookUp USING [emptyKey, InTable, noMatch],
  Token USING [Decimal, FreeTokenString, Handle, Item],
  Tool USING [
    Create, MakeClientSW, MakeFormSW, MakeMsgSW, MakeSWsProc, UnusedLogName],
  ToolWindow USING [TransitionProcType, WindowForSubwindow],
  UserInput USING [AttentionProcType, SetAttention, StringProcType],
  Window USING [Box, GetBox, Handle];

RS232XChatImpl: MONITOR
  IMPORTS
    CmFile, DialupExtras, Emulator, FormSW, Heap, Inline, MStream, Process, Profile, Put, RS232C, RS232XChatOps, Stream, StringLookUp,
Token, Tool, ToolWindow, UserInput, Window
  EXPORTS RS232XChatOps =
  {
  OPEN RS232XChatOps;

  debug: Stream.Handle ← NIL;

  StuffOnQueue: CONDITION;
  aborting, ttyon: BOOLEAN ← FALSE;

  FormItems: TYPE = {
    connect, disconnect, xModem, baud, parity, stop, duplex, charWidth,
    flowControl, terminal, writeToLog, options, phoneNumber, dialerType, lineType};

  data: PUBLIC DataHandle ← NIL;
  wh: PUBLIC Window.Handle ← NIL;
  zone: PUBLIC UNCOUNTED ZONE ← Heap.Create[
    initial: 15, increment: 6, largeNodeThreshold: 512];

  myTransitionIndicator: PUBLIC MyTransitionIndicator ← goingInactive;
  Msg: PUBLIC Format.StringProc = {Put.Text[data.msgSW, s]};

  UserAbort: PUBLIC UserInput.AttentionProcType = {
    IF data.dialProcess # NIL THEN Process.Abort[data.dialProcess];
    };

  Connect: FormSW.ProcType = {
    priority: Process.Priority ← Process.GetPriority[];
    IF data.connected THEN {Msg["Already connected.\n"L]; RETURN};
    Process.SetPriority[Process.priorityNormal];
    Process.Detach[FORK RealConnect[sw, item, index]];
    Process.SetPriority[priority];
    };

  RealConnect: FormSW.ProcType = {
    commParamObject: RS232C.CommParamObject ← [
      duplex: data.duplex, lineType: data.lineType, lineSpeed: data.lineSpeed,
      accessDetail: directConn[]];
    outCome: Dialup.Outcome;
    phoneNumber: LONG POINTER TO Dialup.Number ← NIL;

    IF data.connected THEN Msg["Already connected.\n"L]
    ELSE {
      Msg["Connecting ... "L];
      aborting ← FALSE;

RS232XChatImpl.mesa       16-Jul-87 10:54:01 PDT                                1
```

```
    data.cH ← RS232C.Create[0, @commParamObject, preemptAlways, preemptAlways !
      RS232C.ChannelInUse => {
        Msg["Channel In Use.\n"L];
        GOTO Exit}];
    data.cH.SetParameter[[charLength[data.charLength]]];
    data.cH.SetParameter[[lineSpeed[data.lineSpeed]]];
    data.cH.SetParameter[[parity[data.parity]]];
    data.cH.SetParameter[[stopBits[data.stopBits]]];
    data.cH.SetParameter[[correspondent[RS232CCorrespondents.ttyHost]]];
    data.cH.SetParameter[[frameTimeout[30]]];
    IF data.phoneNumber # NIL AND data.phoneNumber.length # 0 THEN {
      Msg["Dialing ... "L];
      phoneNumber ←
        zone.NEW[Dialup.Number[data.phoneNumber.length]];
        SELECT data.dialerType FROM
        smartmodem =>
          FOR i: CARDINAL IN [0..data.phoneNumber.length) DO
            SELECT data.phoneNumber[i] FROM
              IN ['0..'9] =>
                phoneNumber.number[i] ← LOOPHOLE[data.phoneNumber[i]];
              ', => phoneNumber.number[i] ← 44;        -- two second delay
              'T, 't => phoneNumber.number[i] ← 84;    -- touch tone dialling
              'P, 'p => phoneNumber.number[i] ← 80;    -- pulus dialling
              '; => phoneNumber.number[i] ← 59;        --
              '@ => phoneNumber.number[i] ← 64;        -- wait for quiet answer before dialling
              '! => phoneNumber.number[i] ← 33;        -- go on-hook for 1/2 second
            ENDCASE => {Msg["phone number syntax error"L];
                  zone.FREE[@phoneNumber];
                  data.cH.Delete[];
                  RETURN}
          ENDLOOP;
        Ventel =>
          FOR i: CARDINAL IN [0..data.phoneNumber.length) DO
            SELECT data.phoneNumber[i] FROM
              IN ['0..'9] =>
                phoneNumber.number[i] ← LOOPHOLE[data.phoneNumber[i]];
              '% => phoneNumber.number[i] ← 37;
              '& => phoneNumber.number[i] ← 38;
              '  => phoneNumber.number[i] ← 32;
            ENDCASE => {Msg["phone number syntax error"L];
                  zone.FREE[@phoneNumber];
                  data.cH.Delete[];
                  RETURN};
          ENDLOOP;
        RacalVadic => -- should be Racal Vadic
          FOR i: CARDINAL IN [0..data.phoneNumber.length) DO
            SELECT data.phoneNumber[i] FROM
              IN ['0..'9] =>
                phoneNumber.number[i] ← LOOPHOLE[data.phoneNumber[i]];
              '= => phoneNumber.number[i] ← 61;
            ENDCASE => {Msg["phone number syntax error"L];
                  zone.FREE[@phoneNumber];
                  data.cH.Delete[];
                  RETURN};
          ENDLOOP;
        RS366 =>
          FOR i: CARDINAL IN [0..data.phoneNumber.length) DO
            SELECT data.phoneNumber[i] FROM
              IN ['0..'9] =>
                phoneNumber.number[i] ← LOOPHOLE[data.phoneNumber[i] - '0];
              '* => phoneNumber.number[i] ← 10;
              '# => phoneNumber.number[i] ← 11;
              '= => phoneNumber.number[i] ← 12;
              '< => phoneNumber.number[i] ← 13;
              '> => phoneNumber.number[i] ← Dialup.pause;
              '+ => phoneNumber.number[i] ← 14;
              '/ => phoneNumber.number[i] ← 15;
              ENDCASE => {Msg["phone number syntax error"L];
                        zone.FREE[@phoneNumber];
                        data.cH.Delete[];
                        RETURN}
          ENDLOOP;
        V25bis =>
          FOR i: CARDINAL IN [0..data.phoneNumber.length) DO
            SELECT data.phoneNumber[i] FROM
              IN ['0..'9] =>
                phoneNumber.number[i] ← LOOPHOLE[data.phoneNumber[i]];
              ': => phoneNumber.number[i] ← 58;              -- wait separator
              '; => phoneNumber.number[i] ← 59;              -- parameter separator
              '< => phoneNumber.number[i] ← 60;
              '= => phoneNumber.number[i] ← 61;
              '> => phoneNumber.number[i] ← 62;
            ENDCASE => {Msg["phone number syntax error"L];
                        zone.FREE[@phoneNumber];
                        data.cH.Delete[];
                        RETURN}
          ENDLOOP;
        ENDCASE  => {Msg["unsupported dialer"L];
                        zone.FREE[@phoneNumber];
                        data.cH.Delete[];
                        RETURN};
        data.dialProcess ← Process.GetCurrent[];
        outCome ← DialupExtras.DialExtra[0, phoneNumber, 0, data.dialerType ! ABORTED, RS232C.ChannelSuspended => {
          data.dialProcess ← NIL;
          Msg["dialing aborted."L];
          GOTO Aborted}];
        data.dialProcess ← NIL;
```

```
        IF outCome # success THEN {
          SELECT outCome FROM
            failure => Msg["failed, dialing failure"L];
            aborted => Msg["failed, dialing aborted"L];
            formatError => Msg["failed, dialing formatError"L];
            transmissionError => Msg["failed, dialing transmissionError"L];
            dataLineOccupied => Msg["failed, dialing dataLineOccupied"L];
            dialerNotPresent => Msg["failed, dialing dialerNotPresent"L];
            dialingTimeout => Msg["failed, dialing dialingTimeout"L];
            transferTimeout => Msg["failed, dialing transferTimeout"L];
          ENDCASE => Msg["failed, dialing problem"L];
          zone.FREE[@phoneNumber];
          data.cH.Delete[];
          RETURN};
        zone.FREE[@phoneNumber]};
      data.connected ← TRUE;
      data.cH.SetParameter[[dataTerminalReady[TRUE]]];
      data.cH.SetParameter[[requestToSend[TRUE]]];
      ChangeFlowControl[];
      data.get ← FORK GetData[];
      data.put ← FORK PutData[];
      data.tty ← FORK TTYToDisplay[];
      IF Profile.debugging THEN
        debug ← MStream.Log["TTYDebug.log"L, [Release, NIL]];
      Msg["Connection open to RS232C port.\n"L];
      Emulator.SetRemote[data.fileSW, TRUE];
      };
    EXITS
      Exit => RETURN;
      Aborted => data.cH.Delete[];
    };

  Release: MStream.PleaseReleaseProc = {
    MStream.SetLogReadLength[stream, stream.GetPosition[]]; RETURN[no]};

  Disconnect: FormSW.ProcType = {
    IF data.connected THEN Msg["Not connected.\n"L]
    ELSE {
      Msg["Disconnecting ... "L];
      aborting ← TRUE;
      data.cH.SetParameter[[dataTerminalReady[FALSE]]];
      data.cH.Suspend[all];
      IF data.get # NIL THEN {
        Process.Abort[data.get];
        JOIN data.get;
        data.get ← NIL};
      IF data.put # NIL THEN {
        Process.Abort[data.put];
        JOIN data.put;
        data.put ← NIL};
      IF data.tty # NIL THEN {Process.Abort[data.tty]; JOIN data.tty; };
      data.tty ← NIL;
      data.cH.Delete[];
      IF debug # NIL THEN {debug.Delete[]; debug ← NIL};
      data.connected ← FALSE;
      Emulator.SetRemote[data.fileSW, FALSE];
      IF myTransitionIndicator # goingInactive THEN Msg["Disconnected.\n"L];
      };
    };

  ChangeBaud: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN data.cH.SetParameter[[lineSpeed[data.lineSpeed]]]; };

  ChangeParity: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN data.cH.SetParameter[[parity[data.parity]]]; };

  ChangeStopBits: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN data.cH.SetParameter[[stopBits[data.stopBits]]]; };

  ChangeDuplexity: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN Msg["Must disconnect to change duplexity.\n"L]; };

  ChangeCharLength: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN data.cH.SetParameter[[charLength[data.charLength]]]; };

  ChangeFlowControl: FormSW.EnumeratedNotifyProcType = {
    IF data.connected THEN
      IF data.flowControl = on THEN
        data.cH.SetParameter[[flowControl[[type: xOnXOff, xOn: DC1, xOff: DC3]]]]
      ELSE data.cH.SetParameter[[flowControl[[type: none, xOn: 0, xOff: 0]]]];
    };

  MyOptions: FormSW.ProcType = {
    myBox: Window.Box;
    IF data.options # NIL THEN {Msg["Options window already exists.\n"L]; RETURN};
    myBox ← [ToolWindow.WindowForSubwindow[sw].GetBox[].place, [500, 120]];
    myBox.place.x ← myBox.place.x + 50;
    myBox.place.y ← myBox.place.y + 50;
    CreateOptionsWindow[myBox];
    };

  ChangeTerminal: FormSW.EnumeratedNotifyProcType = {
    [] ← Emulator.SetEmulator[
      data.fileSW, Emulator.table[data.terminal],
      IF data.terminal = vt100 THEN Emulator.table[xvt52] ELSE NIL];
    };
```

```
  SetRefresh: PUBLIC FormSW.EnumeratedNotifyProcType = {
    Emulator.SetRefresh[data.fileSW, data.refresh]};

  MyXModem: FormSW.ProcType = {
    myBox: Window.Box;
    IF data.xmodem # NIL THEN {Msg["XModem window already exists.\n"L]; RETURN};
    IF data.connected THEN {Msg["No connection open.\n"L]; RETURN};
    myBox ← [ToolWindow.WindowForSubwindow[sw].GetBox[].place, [500, 120]];
    myBox.place.x ← myBox.place.x + 50;
    myBox.place.y ← myBox.place.y + 50;
    CreateXModemWindow[myBox];
    };


  ClientTransition: ToolWindow.TransitionProcType = {
    SELECT TRUE FROM
      old = inactive => {
        myTransitionIndicator ← goingActive;
        IF data = NIL THEN data ← zone.NEW[Data ← []];
        ProcessCm[];
        };
      new = inactive =>
        IF data # NIL THEN {
          myTransitionIndicator ← goingInactive;
          IF data.connected THEN Disconnect[];
          zone.FREE[@data]};
      ENDCASE;
    };

  GetData: PROCEDURE = {  -- forked as Process
    buffer: QueueHandle ← NIL;
    status: RS232C.TransferStatus;
    byteCount: CARDINAL;
    rec: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock, body: Environment.nullBlock,
      trailer: Environment.nullBlock];
    Process.SetPriority[Process.priorityNormal];
    BEGIN
    ENABLE { UNWIND => NULL; ABORTED => GOTO quit};
    DO
      ENABLE ABORTED => EXIT;
      buffer ← GetFromQueue[@freelist];
      rec.body ← [LOOPHOLE[@buffer.string.text], 0, buffer.string.maxlength];
      buffer.cH ← data.cH.Get[@rec];
      PutToQueue[@outlist, buffer];
      ENDLOOP;
    EXITS quit => NULL;
    END;
    WHILE outlist # NIL DO
      buffer ← GetFromQueue[@outlist, FALSE];
      IF buffer = NIL THEN EXIT;
      [byteCount, status] ← data.cH.TransferWait[buffer.cH ! ABORTED -> CONTINUE];
      InitBuffer[buffer];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    };

  PutData: PROCEDURE = {
    status: RS232C.TransferStatus;
    byteCount: CARDINAL;
    deviceStatus: RS232C.DeviceStatus;
    buffer: QueueHandle ← NIL;
    Process.SetPriority[Process.priorityNormal];
    BEGIN  -- forked as Process
    ENABLE { UNWIND => NULL; ABORTED => GOTO quit};
    DO
      ENABLE ABORTED => EXIT;
      buffer ← GetFromQueue[@outlist];
      [byteCount, status] ← data.cH.TransferWait[buffer.cH];
      buffer.string.length ← byteCount;
      IF status = aborted THEN EXIT;
      deviceStatus ← data.cH.GetStatus[ ! RS232C.ChannelSuspended => GOTO quit];
      IF deviceStatus.statusAborted THEN EXIT;
      buffer.lostChars ← deviceStatus.dataLost OR deviceStatus.deviceError
        OR status # success;
      IF buffer.lostChars THEN data.cH.SetParameter[[latchBitClear[[
        statusAborted:FALSE, dataLost: FALSE, breakDetected: FALSE,
        clearToSend: FALSE, dataSetReady: FALSE, carrierDetect: FALSE,
        ringHeard: FALSE, ringIndicator: FALSE, deviceError: FALSE]]]];
      IF buffer.lostChars AND Profile.debugging THEN
        NoteError[deviceStatus, status];
      PutToQueue[@filledlist, buffer];
      ENDLOOP;
    EXITS quit => NULL;
    END;  -- of enabled
    WHILE filledlist # NIL DO
      buffer ← GetFromQueue[@filledlist, FALSE];
      IF buffer = NIL THEN EXIT;
      InitBuffer[buffer];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    };

  NoteError: PROC [
    deviceStatus: RS232C.DeviceStatus, status: RS232C.TransferStatus] = {
    debug.PutString["Device status: ["L];
    PutBoolean[deviceStatus.statusAborted, "statusAborted"L, TRUE];
```

```
    PutBoolean[deviceStatus.dataLost, "dataLost"L, TRUE];
    PutBoolean[deviceStatus.breakDetected, "breakDetected"L, TRUE];
    PutBoolean[deviceStatus.clearToSend, "clearToSend"L, TRUE];
    PutBoolean[deviceStatus.dataSetReady, "dataSetReady"L, TRUE];
    PutBoolean[deviceStatus.carrierDetect, "carrierDetect"L, TRUE];
    PutBoolean[deviceStatus.ringHeard, "ringHeard"L, TRUE];
    PutBoolean[deviceStatus.ringIndicator, "ringIndicator"L, TRUE];
    PutBoolean[deviceStatus.deviceError, "deviceError"L, FALSE];
    debug.PutString["]\n"L];
    debug.PutString["Transfer status: "L];
    debug.PutString[
      SELECT status FROM
        success => "success"L,
        dataLost => "dataLost"L,
        deviceError => "deviceError"L,
        frameTimeout => "frameTimeout"L,
        checksumError => "checksumError"L,
        parityError => "parityError"L,
        asynchFramingError => "asynchFramingError"L,
        invalidChar => "invalidChar"L,
        invalidFrame => "invalidFrame"L,
        aborted => "aborted"L,
        ENDCASE => "disaster"L];
    debug.PutString["\n"L];
    };

  PutBoolean: PROC [b: BOOLEAN, s: LONG STRING, comma: BOOLEAN] = {
    debug.PutString[s];
    debug.PutString[": "L];
    debug.PutString[IF b THEN "TRUE"L ELSE "FALSE"L];
    IF comma THEN debug.PutString[", "L];
    };

  bufferMax: CARDINAL = 250;

  Init: PROCEDURE = {
    buffer: QueueHandle ← NIL;
    FOR i: CARDINAL IN [0..10) DO
      buffer ← zone.NEW[
        QueueObject ← [
        next: NIL, lostChars: FALSE, cH: TRASH,
        string: zone.NEW[StringBody [bufferMax]]]];
      PutToQueue[@freelist, buffer];
      ENDLOOP;
    Process.SetTimeout[@StuffOnQueue, Process.MsecToTicks[10]];
    Process.EnableAborts[@StuffOnQueue];
    wh ← Tool.Create[
      makeSWsProc: MakeSWs, initialState: default,
      initialBox: [[0, 30], [512, 700]], clientTransition: ClientTransition,
      name: "RS232XChat"L, cmSection: "RS232XChat"L];
    UserInput.SetAttention[wh, UserAbort];
    ChangeTerminal[];
    SetRefresh[];
    };


  << ProcessCm reads the User.cm file to get the initial settings for the baud rate, refresh, etc. >>

  ProcessCm: PROCEDURE = {
    h: CmFile.Handle = CmFile.UserDotCmOpen[ ! CmFile.Error => RESUME ];
    IF h = NIL THEN RETURN;
    BEGIN
    Options: TYPE = {
      baud, parity, stop, duplex, charWidth, flowControl, terminal, refresh};
    NILString: TYPE = LONG STRING ← NIL;

    a: ARRAY Options OF LONG STRING ← [
      baud: "Baud"L, parity: "Parity"L, stop: "StopBits"L, duplex: "Duplex"L,
      charWidth: "CharWidth"L, flowControl: "FlowControl"L, terminal: "Terminal"L,
      refresh: "Refresh"L];
    baudArray: ARRAY RS232C.LineSpeed OF NILString ← [
      bps300: "300"L, bps600: "600"L, bps1200: "1200"L, bps2400: "2400"L,
      bps4800: "4800"L, bps9600: "9600"L];

    parityArray: ARRAY RS232C.Parity OF NILString ← [
      none: "none"L, odd: "odd"L, even: "even"L];

    duplexArray: ARRAY RS232C.Duplexity OF LONG STRING ← [
      full: "full"L, half: "half"L];

    flowControlArray: ARRAY FlowControl OF LONG STRING ← [on: "on"L, off: "off"L];

    terminalArray: ARRAY Emulator.TerminalType OF NILString ← [
      addrinfo: "addrinfo"L, adm3: "adm3"L, adm3a: "adm3a"L, cdc456: "cdc456"L,
      dm1520: "dm1520"L, gt100: "gt100"L, h1000: "h1000"L, h1420: "h1420"L,
      h1500: "h1500"L, h1510: "h1510"L, h1520: "h1520"L, h2000: "h2000"L,
      isc8001: "isc8001"L, soroc: "soroc"L, teletec: "teletec"L, trs80: "trs80"L,
      vc303: "vc303"L, vt100: "vt100"L, vt50: "vt50"L, vt50h: "vt50h"L,
      vt52: "vt52"L, x820: "x820"L];

    refreshesArray: ARRAY Emulator.RefreshHint OF LONG STRING ← [
      always: "always"L, never: "never"L, half: "half"L, full: "full"L];

    value: LONG STRING;
    v: CARDINAL;

    IF CmFile.FindSection[h, "RS232XChat"L ! CmFile.Error => GOTO exit] THEN
```

```
      GOTO exit;
    DO
      SELECT CmFile.NextValue[
      h, LOOPHOLE[LONG[DESCRIPTOR[a]]] ! CmFile.TableError => RESUME ] FROM
        Options.baud.ORD => {
          value ← Token.Item[h];
          SELECT v ← StringLookUp.InTable[
          value, LOOPHOLE[LONG[DESCRIPTOR[baudArray]]]] FROM
            StringLookUp.emptyKey, StringLookUp.noMatch =>
              data.lineSpeed ← bps1200;
            ENDCASE => data.lineSpeed ← VAL[v];
          value ← Token.FreeTokenString[value];
          };
        Options.parity.ORD => {
          value ← Token.Item[h];
          SELECT v ← StringLookUp.InTable[
          value, LOOPHOLE[LONG[DESCRIPTOR[parityArray]]]] FROM
            StringLookUp.emptyKey, StringLookUp.noMatch => data.parity ← none;
            ENDCASE => data.parity ← VAL[v];
          value ← Token.FreeTokenString[value];
          };
        Options.stop.ORD => {
          data.stopBits ←
            SELECT v ← Token.Decimal[h, FALSE] FROM 0 => 1, 1, 2 => v ENDCASE => 1;
          };
        Options.duplex.ORD => {
          value ← Token.Item[h];
          SELECT v ← StringLookUp.InTable[
          value, LOOPHOLE[LONG[DESCRIPTOR[duplexArray]]]] FROM
            StringLookUp.emptyKey, StringLookUp.noMatch => data.duplex ← full;
            ENDCASE => data.duplex ← VAL[v];
          value ← Token.FreeTokenString[value];
          };
        Options.charWidth.ORD => {
          data.charLength ←
            SELECT v ← Token.Decimal[h, FALSE] FROM
              0 => 8,
              5, 6, 7, 8, 9 => v
              ENDCASE => 8;
          };
        Options.flowControl.ORD => {
          value ← Token.Item[h];
          SELECT v ← StringLookUp.InTable[
          value, LOOPHOLE[LONG[DESCRIPTOR[flowControlArray]]]] FROM
            StringLookUp.emptyKey, StringLookUp.noMatch => data.flowControl ← on;
            ENDCASE => data.flowControl ← VAL[v];
          value ← Token.FreeTokenString[value];
          };
        Options.terminal.ORD => {
          value ← Token.Item[h];
          SELECT v ← StringLookUp.InTable[
          value, LOOPHOLE[LONG[DESCRIPTOR[terminalArray]]]] FROM
            StringLookUp.emptyKey, StringLookUp.noMatch => data.terminal ← vt100;
            ENDCASE => data.terminal ← VAL[v];
          value ← Token.FreeTokenString[value];
          };
        Options.refresh.ORD => {
          value ← Token.Item[h];
          SELECT v ← StringLookUp.InTable[
          value, LOOPHOLE[LONG[DESCRIPTOR[refreshesArray]]]] FROM
            StringLookUp.emptyKey, StringLookUp.noMatch => data.refresh ← always;
            ENDCASE => data.refresh ← VAL[v];
          value ← Token.FreeTokenString[value];
          };
        ENDCASE => EXIT;
      ENDLOOP;
    [] ← CmFile.Close[h];

    EXITS exit => [] ← CmFile.Close[h];
    END;
    };


  KeyToTTY: UserInput.StringProcType = {
    physRecord: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock,
      body: [
      startIndex: 0, stopIndexPlusOne: string.length,
      blockPointer: LOOPHOLE[@string.text]], trailer: Environment.nullBlock];
    IF data.started THEN RETURN;
    IF data.connected THEN {
      ch: CHARACTER;
      FOR i: CARDINAL IN [0..string.length) DO
        IF (ch ← string[i]) = '\n THEN Emulator.CR[data.fileSW]
        ELSE Emulator.PutChar[data.fileSW, ch];
        ENDLOOP;
      RETURN;
      };
    [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
    };

  DisableIO: PUBLIC PROC = {
    priority: Process.Priority ← Process.GetPriority[];
    Process.SetPriority[Process.priorityForeground];
    IF data.tty # NIL THEN {
      Process.Abort[data.tty]; JOIN data.tty; data.tty ← NIL; };
    Process.SetPriority[priority];
```

```
    };

  EnableIO: PUBLIC PROC = {
    IF data.tty = NIL THEN data.tty ← FORK TTYToDisplay[]; };

  TTYToDisplay: PROCEDURE = {
    buffer: QueueHandle ← NIL;
    s: LONG STRING ← NIL;
    i, j: CARDINAL ← 0;
    Process.SetPriority[Process.priorityNormal];
    ttyon ← TRUE;
    DO
      ENABLE ABORTED => EXIT;
      buffer ← GetBuffer[ ! ABORTED => EXIT; ANY => LOOP];
      IF buffer.lostChars THEN Msg["Characters lost.\n"L];
      j ← 0;
      s ← buffer.string;
      FOR i IN [0..s.length) DO
        IF s[i] > '\177 THEN s[i] ← Inline.BITAND[s[i], 177B]; ENDLOOP;
      FOR i IN [0..s.length) DO
        IF (s[i] = Ascii.NUL) OR (s[i] = Ascii.DEL) THEN LOOP;
        s[j] ← s[i];
        j ← j + 1;
        ENDLOOP;
      s.length ← j;
      Emulator.PutString[data.fileSW, s];
      ReturnBuffer[buffer];
      buffer ← NIL;
      ENDLOOP;
    IF buffer # NIL THEN ReturnBuffer[buffer];
    ttyon ← FALSE;
    };  -- TTYToDisplay

  freelist, outlist, filledlist: QueueHandle ← NIL;

  GetBuffer: PUBLIC PROC RETURNS [buffer: QueueHandle] = {
    ENABLE UNWIND => NULL; buffer ← GetFromQueue[@filledlist]; };  -- GetBuffer

  ReturnBuffer: PUBLIC PROC [buffer: QueueHandle] = {
    InitBuffer[buffer]; PutToQueue[@freelist, buffer]};

  InitBuffer: PROC [buffer: QueueHandle] = {
    buffer.lostChars ← FALSE; buffer.next ← NIL; };

  PutToQueue: ENTRY PROC [q: LONG POINTER TO QueueHandle, buf: QueueHandle] = {
    ENABLE UNWIND => NULL;
    i: LONG POINTER TO QueueHandle ← NIL;
    FOR i ← q, @i.next UNTIL i↑ = NIL DO ENDLOOP;
    i↑ ← buf;
    buf.next ← NIL;
    BROADCAST StuffOnQueue;
    };

  GetFromQueue: ENTRY PROC [q: LONG POINTER TO QueueHandle, wait: BOOLEAN ← TRUE]
    RETURNS [buf: QueueHandle] = {
    ENABLE UNWIND => NULL;
    IF wait THEN
      WHILE q↑ = NIL DO WAIT StuffOnQueue; IF aborting THEN ERROR ABORTED; ENDLOOP;
    buf ← q↑;
    IF buf # NIL THEN q↑ ← buf.next};

  WriteToLog: FormSW.NotifyProcType = {
    IF data.writeToLog THEN Emulator.StartLog[data.fileSW]
    ELSE Emulator.StopLog[data.fileSW];
    };

  CreateSW: PROC [sw: Window.Handle, clientData: LONG POINTER] = {
    logName: STRING ← [40];
    Tool.UnusedLogName[unused: logName, root: "RS232XChat.log"L];
    Emulator.Create[
      sw: sw, data: Emulator.table[vt100], typeIn: KeyToTTY, logfile: logName,
      writeToLog: TRUE, refresh: half];

    };

  MakeSWs: Tool.MakeSWsProc = {
    data.msgSW ← Tool.MakeMsgSW[window: window];
    data.formSW ← Tool.MakeFormSW[window: window, formProc: MakeForm];
    data.fileSW ← Tool.MakeClientSW[window, CreateSW, NIL, Emulator.EmuSWType];
    UserInput.SetAttention[data.msgSW, UserAbort];
    UserInput.SetAttention[data.formSW, UserAbort];
    UserInput.SetAttention[data.fileSW, UserAbort];
    };

  MakeForm: FormSW.ClientItemsProcType = {
    OPEN FormSW;
    nItems: CARDINAL = FormItems.LAST.ORD + 1;
    baudArray: ARRAY [0..7) OF Enumerated ← [
      ["300"L, RS232C.LineSpeed.bps300.ORD], ["600"L, RS232C.LineSpeed.bps600.ORD],
      ["1200"L, RS232C.LineSpeed.bps1200.ORD], [
      "2400"L, RS232C.LineSpeed.bps2400.ORD], [
      "4800"L, RS232C.LineSpeed.bps4800.ORD], [
      "9600"L, RS232C.LineSpeed.bps9600.ORD], [
      "19200"L, RS232C.LineSpeed.bps19200.ORD]];
    parityArray: ARRAY [0..3) OF Enumerated ← [
      ["none"L, RS232C.Parity.none.ORD], ["odd"L, RS232C.Parity.odd.ORD], [
      "even"L, RS232C.Parity.even.ORD]];
```

```
    stop: ARRAY [0..2) OF Enumerated ← [["1"L, 1], ["2"L, 2]];
    duplexArray: ARRAY [0..2) OF Enumerated ← [
      ["full"L, RS232C.Duplexity.full.ORD], ["half"L, RS232C.Duplexity.half.ORD]];
    bits: ARRAY [0..5) OF Enumerated ← [
      ["5"L, 5], ["6"L, 6], ["7"L, 7], ["8"L, 8], ["9"L, 9]];
    flowControlArray: ARRAY [0..2) OF Enumerated ← [
      ["FlowCntl ON"L, FlowControl.on.ORD], [
      "FlowCntl OFF"L, FlowControl.off.ORD]];
    terminals: ARRAY [0..22) OF Enumerated ← [
      ["addrinfo"L, Emulator.TerminalType.addrinfo.ORD], [
      "adm3"L, Emulator.TerminalType.adm3.ORD], [
      "adm3a"L, Emulator.TerminalType.adm3a.ORD], [
      "cdc456"L, Emulator.TerminalType.cdc456.ORD], [
      "dm1520"L, Emulator.TerminalType.dm1520.ORD], [
      "gt100"L, Emulator.TerminalType.gt100.ORD], [
      "h1000"L, Emulator.TerminalType.h1000.ORD], [
      "h1420"L, Emulator.TerminalType.h1420.ORD], [
      "h1500"L, Emulator.TerminalType.h1500.ORD], [
      "h1510"L, Emulator.TerminalType.h1510.ORD], [
      "h1520"L, Emulator.TerminalType.h1520.ORD], [
      "h2000"L, Emulator.TerminalType.h2000.ORD], [
      "isc8001"L, Emulator.TerminalType.isc8001.ORD], [
      "soroc"L, Emulator.TerminalType.soroc.ORD], [
      "teletec"L, Emulator.TerminalType.teletec.ORD], [
      "trs80"L, Emulator.TerminalType.trs80.ORD], [
      "vc303"L, Emulator.TerminalType.vc303.ORD], [
      "vt100"L, Emulator.TerminalType.vt100.ORD], [
      "vt50"L, Emulator.TerminalType.vt50.ORD], [
      "vt50h"L, Emulator.TerminalType.vt50h.ORD], [
      "vt52"L, Emulator.TerminalType.vt52.ORD], [
      "x820"L, Emulator.TerminalType.x820.ORD]];
    refreshes: ARRAY [0..4) OF Enumerated ← [
      ["always"L, Emulator.RefreshHint.always.ORD], [
      "never"L, Emulator.RefreshHint.never.ORD], [
      "half"L, Emulator.RefreshHint.half.ORD], [
      "full"L, Emulator.RefreshHint.full.ORD]];
    lineTypeArray: ARRAY[0..3) OF Enumerated ← [
      ["bitSynchronous"L, 0], ["byteSynchronous"L, 1], ["asynchronous"L, 2]];
    dialerTypeArray: ARRAY[0..7) OF Enumerated ← [
      ["RS366"L, 0], ["Ventel"L, 1], ["smartmodem"L, 2], ["RacalVadic"L, 3], ["V25bis"L, 4], ["V25"L, 5], ["other"L, 6]];
    items ← FormSW.AllocateItemDescriptor[nItems];
    items[FormItems.connect.ORD] ← FormSW.CommandItem[
      tag: "Connect"L, place: [0, FormSW.line0], proc: Connect];
    items[FormItems.disconnect.ORD] ← FormSW.CommandItem[
      tag: "Disconnect"L, place: [180, FormSW.line0], proc: Disconnect];
    items[FormItems.xModem.ORD] ← FormSW.CommandItem[
      tag: "XModem"L, place: [312, FormSW.line0], proc: MyXModem];
    items[FormItems.baud.ORD] ← FormSW.EnumeratedItem[
      tag: "Baud"L, place: [0, FormSW.line1], proc: ChangeBaud,
      choices: DESCRIPTOR[baudArray], value: @data.lineSpeed];
    items[FormItems.parity.ORD] ← FormSW.EnumeratedItem[
      tag: "Parity"L, place: [180, FormSW.line1], proc: ChangeParity,
      choices: DESCRIPTOR[parityArray], value: @data.parity];
    items[FormItems.stop.ORD] ← FormSW.EnumeratedItem[
      tag: "Stop"L, place: [312, FormSW.line1], proc: ChangeStopBits,
      choices: DESCRIPTOR[stop], value: @data.stopBits];
    items[FormItems.duplex.ORD] ← FormSW.EnumeratedItem[
      tag: "Duplex"L, place: [0, FormSW.line2], proc: ChangeDuplexity,
      choices: DESCRIPTOR[duplexArray], value: @data.duplex];
    items[FormItems.charWidth.ORD] ← FormSW.EnumeratedItem[
      tag: "CharWidth"L, place: [180, FormSW.line2], proc: ChangeCharLength,
      choices: DESCRIPTOR[bits], value: @data.charLength];
    items[FormItems.flowControl.ORD] ← FormSW.EnumeratedItem[
      tag: "FlowControl"L, place: [312, FormSW.line2], proc: ChangeFlowControl,
      choices: DESCRIPTOR[flowControlArray], value: @data.flowControl];
    items[FormItems.terminal.ORD] ← FormSW.EnumeratedItem[
      tag: "Terminal"L, place: [0, FormSW.line3], proc: ChangeTerminal,
      choices: DESCRIPTOR[terminals], value: @data.terminal];
    items[FormItems.writeToLog.ORD] ← FormSW.BooleanItem[
      tag: "WriteToLog"L, place: [180, FormSW.line3], proc: WriteToLog,
      switch: @data.writeToLog];
    items[FormItems.options.ORD] ← FormSW.CommandItem[
      tag: "Options"L, place: [312, FormSW.line3], proc: MyOptions];
    items[FormItems.phoneNumber.ORD] ← StringItem[
      tag: "Ph. #"L, place: [0, line4], string: @data.phoneNumber,
      inHeap: TRUE];
    items[FormItems.dialerType.ORD] ← EnumeratedItem[
      tag: "Dialer type"L, place: [125, line4],  feedback: one, choices: DESCRIPTOR[dialerTypeArray], value: @data.dialerType];
    items[FormItems.lineType.ORD] ← EnumeratedItem[
      tag: "Line type"L, place: [312, line4],  feedback: one, choices: DESCRIPTOR[lineTypeArray], value: @data.lineType];

    Msg["Disconnected.\n"L];
    RETURN[items: items, freeDesc: TRUE];
    };


-- Mainline code

Init[];  -- this gets string out of global frame

}...
```

```
-- File: RS232XChatOps.mesa - last edit:
-- JAV                  31-Mar-87 11:46:08
-- BGY                   6-May-86 10:08:42

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  DialupExtras,
  Emulator,
  FormSW,
  Format,
  RS232C,
  RS232CEnvironment,
  Stream,
  Window;


RS232XChatOps: DEFINITIONS = {

  DataHandle: TYPE = LONG POINTER TO Data;
  Data: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    fileSW(4): Window.Handle ← NIL,
    lineSpeed(6): RS232C.LineSpeed ← bps1200,
    parity(7): RS232C.Parity ← none,
    stopBits(8): INTEGER ← 1,
    duplex(9): RS232C.Duplexity ← full,
    charLength(10): INTEGER ← 8,
    terminal(11): Emulator.TerminalType ← vt100,
    refresh(12): Emulator.RefreshHint ← always,
    cH(13): RS232C.ChannelHandle ← TRASH,
    get(15): PROCESS ← NIL,
    put(16): PROCESS ← NIL,
    tty(17): PROCESS ← NIL,
    connected(18): BOOLEAN ← FALSE,
    flowControl(19): FlowControl ← off,
    xmodem(20): Window.Handle ← NIL,
    xmdFormSW(22): Window.Handle ← NIL,
    xmdMsgSW(24): Window.Handle ← NIL,
    filename(26): LONG STRING ← NIL,
    filecheck(28): CARDINAL ← 0,
    dataDirection(29): CARDINAL ← 0,
    started(30): BOOLEAN ← FALSE,
    stream(31): Stream.Handle ← NIL,
    options(33): Window.Handle ← NIL,
    writeToLog(35): BOOLEAN ← TRUE,
    optionformSW(36): Window.Handle ← NIL,
    dialProcess(38): PROCESS ← NIL,
    lineType(39): RS232CEnvironment.LineType ← asynchronous,
    phoneNumber(40): LONG STRING ← NIL,
    dialerType(42): DialupExtras.DialerType ← smartmodem
    ];

  FlowControl: TYPE = {on, off};
  MyTransitionIndicator: TYPE = {goingInactive, goingActive};
  myTransitionIndicator: MyTransitionIndicator;

  DC1: UNSPECIFIED = 11H;
  DC3: UNSPECIFIED = 13H;

  QueueHandle: TYPE =  LONG POINTER TO QueueObject;
  QueueObject: TYPE = RECORD[
    next: QueueHandle,
    lostChars: BOOLEAN,
    cH: RS232C.CompletionHandle,
    string: LONG STRING];

  data: DataHandle;
  wh: Window.Handle;
  zone: UNCOUNTED ZONE;

  Msg: Format.StringProc;

  CreateXModemWindow, CreateOptionsWindow: PROC[box: Window.Box];

  DisableIO, EnableIO: PROC;

  GetBuffer: PROC RETURNS [buffer: QueueHandle];

  ReturnBuffer: PROC [buffer: QueueHandle];
  SetRefresh:  FormSW.EnumeratedNotifyProcType;

  }.
```

```
-- RS232XModem.mesa
-- BGY  15-May-86 11:30:02

-- Copyright (C) 1985, 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii USING [DEL, NUL],
  Cursor USING [Set],
  Emulator USING [CreateOptionsSheet, PutString],
  EmulatorOps USING [myTip],
  Environment USING [Block, Byte, bytesPerPage, nullBlock],
  Exec USING [AddCommand, ExecProc, Handle, Object, RemoveCommand],
  FormSW USING [
    AllocateItemDescriptor, ClientItemsProcType, CommandItem, Enumerated,
    EnumeratedItem, line0, line1, line2, line3, ProcType, StringItem],
  Heap USING [Delete],
  HeraldWindow USING [
    FreeCursorSlot, GetCursorSlot, SetCursor, SetCursorState, Slot],
  Inline USING [BITAND],
  MFile USING [Access, Acquire, Error, Handle, Object, Release],
  MStream USING [Error, Handle, ReadOnly, ReleaseData, WriteOnly],
  Process USING [Abort, Detach, GetCurrent, MsecToTicks, Pause],
  Put USING [Text],
  RS232C USING [PhysicalRecord, Put, TransmitNow],
  RS232XChatOps USING [
    data, DataHandle, DisableIO, EnableIO, GetBuffer, Msg, QueueHandle,
    ReturnBuffer, SetRefresh, wh, zone],
  Stream USING [
    Block, Byte, defaultInputOptions, defaultObject, Delete, DeleteProcedure,
    EndOfStream, GetByteProcedure, Handle, Object, PutByteProcedure, PutProcedure,
    SendNowProcedure],
  String USING [AppendString],
  TIP USING [CreateTable, globalTable, InvalidTable, PushLocal, SetTable, Table],
  Tool USING [Create, Destroy, MakeFormSW, MakeMsgSW, MakeSWsProc],
  ToolWindow USING [SetName, TransitionProcType],
  UserInput USING [WaitForConfirmation],
  UserTerminal USING [BlinkDisplay],
  Window USING [Box, Handle, Object],
  WindowFont USING [CharWidth],
  XModem USING [
    Abort, Create, OperatorAbort, PacketError, Receiving, Sending,
    SetOperatorAbort];

RS232XModem: MONITOR
  IMPORTS
    Cursor, Emulator, Exec, EmulatorOps, FormSW, Heap, HeraldWindow, Inline, MFile,
    MStream, RS232C, RS232XChatOps, Process, Put, Stream, String, TIP, Tool,
    ToolWindow, UserInput, UserTerminal, XModem, WindowFont
  EXPORTS RS232XChatOps =
  {
  OPEN RS232XChatOps;


  FormOptionItems: TYPE = {
    closeOptions, reCreateTipTable, terminalOptions, refresh};

  CreateXModemWindow: PUBLIC PROC [box: Window.Box] = {
    data.xmodem ← Tool.Create[
      makeSWsProc: MakeXMDSWs, initialState: default, initialBox: box,
      name: "XModem Transfers"L, cmSection: "XModem"L,
      clientTransition: Transition]};

  CreateOptionsWindow: PUBLIC PROC [box: Window.Box] = {
    data.options ← Tool.Create[
      makeSWsProc: MakeOptionsSWs, initialState: default, initialBox: box,
      clientTransition: Transition, name: "RS232XChat Options"L,
      cmSection: "RS232XChatOptions"L];
    };

  Transition: ToolWindow.TransitionProcType = {
    SELECT TRUE FROM
      old = inactive => IF data = NIL THEN {};
      new = inactive => NULL;
      ENDCASE;
    };

  MakeXMDSWs: Tool.MakeSWsProc = {
    data.xmdFormSW ← Tool.MakeFormSW[window: window, formProc: MakeXModem];
    data.xmdMsgSW ← Tool.MakeMsgSW[window: window];
    };

  MakeOptionsSWs: Tool.MakeSWsProc = {
    logName: LONG STRING ← [40];
    data.optionformSW ← Tool.MakeFormSW[window: window, formProc: MakeOptionForm];
    };

  XModemItems: TYPE = {start, abort, filename, dataDirection, filecheck};

  CloseOptions: FormSW.ProcType = {
    Tool.Destroy[data.options]; data.options ← NIL; };

  TerminalOptions: FormSW.ProcType = {Emulator.CreateOptionsSheet[data.fileSW]};

  ReCreateTipTable: FormSW.ProcType = {
    firstTime: BOOLEAN ← TRUE;
    tempTable: TIP.Table ← EmulatorOps.myTip;
```

```
    EmulatorOps.myTip ← TIP.CreateTable[
      file: "Emulator.TIP"L !
      TIP.InvalidTable =>
        IF type # badSyntax THEN CONTINUE
        ELSE {
          UserTerminal.BlinkDisplay[];
          IF firstTime THEN {firstTime ← FALSE; RESUME ; }}];
    IF EmulatorOps.myTip # NIL THEN {
      TIP.PushLocal[push: EmulatorOps.myTip, onto: TIP.globalTable[root]];
      [] ← TIP.SetTable[window: data.fileSW, table: EmulatorOps.myTip];
      Msg["TIP table recreated successfully.\n"L];
      }
    ELSE {
      EmulatorOps.myTip ← tempTable;
      Msg["Was not able to recreate the TIP table,  Still using old one.\n"L];
      };
    };

  MakeOptionForm: FormSW.ClientItemsProcType = {
    nItems: CARDINAL = FormOptionItems.LAST.ORD + 1;
    refreshes: ARRAY [0..4) OF FormSW.Enumerated ← [
      ["always"L, 0], ["never"L, 1], ["half"L, 2], ["full"L, 3]];
    items ← FormSW.AllocateItemDescriptor[nItems];
    items[FormOptionItems.closeOptions.ORD] ← FormSW.CommandItem[
      tag: "CloseOptions"L, place: [0, FormSW.line0], proc: CloseOptions];
    items[FormOptionItems.reCreateTipTable.ORD] ← FormSW.CommandItem[
      tag: "ReCreateTipTable"L, place: [180, FormSW.line0],
      proc: ReCreateTipTable];
    items[FormOptionItems.terminalOptions.ORD] ← FormSW.CommandItem[
      tag: "TerminalOptions"L, place: [312, FormSW.line0], proc: TerminalOptions];
    items[FormOptionItems.refresh.ORD] ← FormSW.EnumeratedItem[
      tag: "Refresh"L, place: [0, FormSW.line1], feedback: all, proc: SetRefresh,
      choices: DESCRIPTOR[refreshes], value: @data.refresh];
    RETURN[items: items, freeDesc: TRUE];
    };

  MakeXModem: FormSW.ClientItemsProcType = {
    OPEN FormSW;
    nItems: CARDINAL = XModemItems.LAST.ORD + 1;
    filecheck: ARRAY [0..2) OF Enumerated ← [["Checksum"L, 0], ["CRC"L, 1]];
    dataDirection: ARRAY [0..2) OF Enumerated ← [
      ["Transmit"L, 0], ["Receive"L, 1]];
    items ← AllocateItemDescriptor[nItems];
    items[XModemItems.start.ORD] ← CommandItem[
      tag: "Start"L, place: [CharPos[0], line0], proc: Start1];
    items[XModemItems.abort.ORD] ← CommandItem[
      tag: "Abort"L, place: [CharPos[15], line0], proc: XMDAbort];
    items[XModemItems.filename.ORD] ← StringItem[
      tag: "Filename"L, place: [CharPos[0], line1], inHeap: TRUE,
      string: @data.filename];
    items[XModemItems.dataDirection.ORD] ← EnumeratedItem[
      tag: "Data direction"L, place: [CharPos[0], line2], feedback: all,
      choices: DESCRIPTOR[dataDirection], value: @data.dataDirection];
    items[XModemItems.filecheck.ORD] ← EnumeratedItem[
      tag: "Filecheck"L, place: [CharPos[0], line3],
      choices: DESCRIPTOR[filecheck], value: @data.filecheck];
    RETURN[items: items, freeDesc: TRUE];
    };

  charWidth: CARDINAL ← WindowFont.CharWidth['0];
  CharPos: PROC [char: CARDINAL] RETURNS [x: INTEGER] = {x ← charWidth * char};

  MsgXMD: PROC [s: LONG STRING] = {
    IF data.xmodem = NIL OR PutIt[s] THEN Msg[s]; };

  PutIt: ENTRY PROC [s: LONG STRING] RETURNS [ok: BOOLEAN] = {
    ENABLE UNWIND => NULL;
    IF data.xmodem = NIL THEN RETURN[FALSE];
    Put.Text[data.xmdMsgSW, s];
    RETURN[TRUE];
    };


  Start1: FormSW.ProcType = {
    Msg["\n"L];
    IF data.started THEN {MsgXMD["Already started.\n"L]; RETURN};
    IF data.dataDirection # 0 AND FileExists[data.filename] THEN {
      yes: BOOLEAN;
      MsgXMD["File will be overwritten [Confirm].\n"L];
      Cursor.Set[mouseRed];
      yes ← UserInput.WaitForConfirmation[].okay;
      Cursor.Set[textPointer];
      IF yes THEN RETURN;
      };
    me ← FORK Start[sw, item, index];
    Process.Detach[me]};

  me: PROCESS ← NIL;
  Start: FormSW.ProcType = {
    slot: HeraldWindow.Slot ← NIL;
    InputStream, OutputStream: Stream.Handle ← NIL;
    buffer: LONG STRING ← NIL;
    Cleanup: PROC = {
      IF getbyte # NIL THEN Process.Abort[getbyte];
      WHILE getbyte # NIL DO Process.Pause[Process.MsecToTicks[1]]; ENDLOOP;
      IF OutputStream # NIL THEN Stream.Delete[OutputStream];
      IF InputStream # NIL THEN Stream.Delete[InputStream];
```

```
      EnableIO[];
      Process.Detach[FORK Destroyer[]];
      SetStopped[data];
      IF slot # NIL THEN slot ← HeraldWindow.FreeCursorSlot[slot];
      me ← NIL;
      zone.FREE[@buffer];
      };
    BEGIN
    ENABLE {
      UNWIND => me ← NIL;
      MStream.Error => {
        SELECT code FROM
          invalidOperation => {Msg["invalidOperation - MFile\n"L]; GOTO fileError};
          fileTooLong => {Msg["fileTooLong - MFile\n"L]; GOTO fileError};
          fileNotAvailable => {Msg["fileNotAvailable - MFile\n"L]; GOTO fileError};
          ENDCASE};
      XModem.PacketError, XModem.Abort, ABORTED => {
        Msg["XModem Error\n"L]; GOTO fileError};
      XModem.OperatorAbort => {Msg["OperatorAbort\n"L]; GOTO fileError}};

    EventNotify: PROC = {
      IF slot # NIL THEN HeraldWindow.SetCursorState[slot, invert]};

    IF data = NIL THEN RETURN;
    buffer ← zone.NEW[StringBody [150]];
    IF Started[data] THEN {MsgXMD["Already started.\n"L]; RETURN}
    ELSE SetStarted[data];
    DisableIO[];
    data.stream ← CreateRS232Stream[];
    XModem.SetOperatorAbort[FALSE];
    IF data.dataDirection = 0 THEN {
      InputStream ← GetReadStream[data.filename];
      OutputStream ← XModem.Create[data.stream, Stream.defaultInputOptions];
      slot ← HeraldWindow.GetCursorSlot[];
      IF slot # NIL THEN HeraldWindow.SetCursor[slot, ftpBoxes];
      String.AppendString[buffer, "XModem: Sending file "L];
      String.AppendString[buffer, data.filename];
      ToolWindow.SetName[data.xmodem, buffer];
      XModem.Sending[
        OutputStream, InputStream, IF data.filecheck = 0 THEN CheckSum ELSE CRC,
        EventNotify];
      }
    ELSE {
      OutputStream ← GetWriteStream[data.filename];
      InputStream ← XModem.Create[data.stream, Stream.defaultInputOptions];
      slot ← HeraldWindow.GetCursorSlot[];
      IF slot # NIL THEN HeraldWindow.SetCursor[slot, ftpBoxes];
      String.AppendString[buffer, "XModem: Retrieving file "L];
      String.AppendString[buffer, data.filename];
      ToolWindow.SetName[data.xmodem, buffer];
      XModem.Receiving[
        OutputStream, InputStream, IF data.filecheck = 0 THEN CheckSum ELSE CRC,
        EventNotify];
      };
    EXITS fileError => NULL;
    END;
    Cleanup[];
    };

  maxPages: CARDINAL = 2;
  maxBytes: CARDINAL = maxPages * Environment.bytesPerPage;

  StreamObject: TYPE = RECORD [
    stream: Stream.Object ← Stream.defaultObject,
    input: QueueHandle ← NIL,
    inputPos: CARDINAL ← 0,
    outputPos: CARDINAL ← 0,
    outputBuffer: PACKED ARRAY [0..maxBytes) OF Environment.Byte ← TRASH];

  StreamHandle: TYPE = LONG POINTER TO StreamObject;

  CreateRS232Stream: PROC RETURNS [stream: Stream.Handle] = {
    s: StreamHandle ← zone.NEW[StreamObject ← []];
    s.stream.put ← PutBlock;
    s.stream.putByte ← PutByte;
    s.stream.getByte ← GetByte;
    s.stream.sendNow ← SendNow;
    s.stream.delete ← Delete;
    stream ← LOOPHOLE[s];
    };

  getbyte: PROCESS ← NIL;

  GetByte: Stream.GetByteProcedure = {
    s: StreamHandle ← LOOPHOLE[sH];
    getbyte ← Process.GetCurrent[];
    IF s.input = NIL THEN s.input ← GetBuffer[ ! ABORTED => GOTO nope];
    byte ← LOOPHOLE[s.input.string[s.inputPos]];
    s.inputPos ← s.inputPos + 1;
    IF s.inputPos >= s.input.string.length THEN {
      ReturnBuffer[s.input]; s.inputPos ← 0; s.input ← NIL; };
    getbyte ← NIL;
    EXITS nope => {getbyte ← NIL; SIGNAL Stream.EndOfStream[0]};
    };

  MakeBlock: PROC [s: StreamHandle] RETURNS [block: Environment.Block] = INLINE {
    block ← [
```

```
      startIndex: 0, stopIndexPlusOne: s.outputPos,
      blockPointer: LOOPHOLE[@s.outputBuffer]];
    };

  PutByte: Stream.PutByteProcedure = {
    s: StreamHandle ← LOOPHOLE[sH];
    IF s.outputPos + 1 >= maxBytes THEN {
      physRecord: RS232C.PhysicalRecord ← [
        header: Environment.nullBlock, body: MakeBlock[s],
        trailer: Environment.nullBlock];
      [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
      s.outputPos ← 0;
      };
    s.outputBuffer[s.outputPos] ← byte;
    s.outputPos ← s.outputPos + 1;
    };

  SendNow: Stream.SendNowProcedure = {
    s: StreamHandle ← LOOPHOLE[sH];
    physRecord: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock, body: MakeBlock[s],
      trailer: Environment.nullBlock];
    [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
    s.outputPos ← 0;
    };

  PutBlock: Stream.PutProcedure = {
    s: StreamHandle ← LOOPHOLE[sH];
    charsLeft, spaceLeft: CARDINAL;
    physRecord: RS232C.PhysicalRecord ← [
      header: Environment.nullBlock, body: TRASH, trailer: Environment.nullBlock];
    charsLeft ← block.stopIndexPlusOne - block.startIndex;
    spaceLeft ← maxBytes - s.outputPos;
    IF charsLeft > maxBytes THEN {
      -- got a large buffer, so send our buffer then the large one then quit
      physRecord.body ← MakeBlock[s];
      [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
      s.outputPos ← 0;
      physRecord.body ← block;
      [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
      RETURN;
      };
    IF charsLeft > spaceLeft THEN {
      physRecord.body ← MakeBlock[s];
      [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
      s.outputPos ← 0;
      };
    FOR i: CARDINAL IN [block.startIndex..block.stopIndexPlusOne) DO
      s.outputBuffer[s.outputPos] ← block.blockPointer[i];
      s.outputPos ← s.outputPos + 1;
      ENDLOOP;
    IF endRecord THEN {
      physRecord.body ← MakeBlock[s];
      [] ← data.cH.TransmitNow[data.cH.Put[@physRecord]];
      s.outputPos ← 0;
      };
    };

  Delete: Stream.DeleteProcedure = {
    s: StreamHandle ← LOOPHOLE[sH];
    i, j: CARDINAL ← 0;
    string: LONG STRING ← NIL;
    IF s.input # NIL THEN {
      IF s.inputPos < s.input.string.length THEN {
        -- put rest of buffer to tty
        pos: CARDINAL ← 0;
        string ← s.input.string;
        FOR i IN [s.inputPos..string.length) DO
          string[pos] ← string[i]; pos ← pos + 1; ENDLOOP;
        string.length ← pos;
        FOR i IN [0..string.length) DO
          IF string[i] > '\177 THEN string[i] ← Inline.BITAND[string[i], 177B]
          ENDLOOP;
        FOR i IN [0..string.length) DO
          IF (string[i] = Ascii.NUL) OR (string[i] = Ascii.DEL) THEN LOOP;
          string[j] ← string[i];
          j ← j + 1;
          ENDLOOP;
        string.length ← j;
        Emulator.PutString[data.fileSW, string];
        };
      ReturnBuffer[s.input];
      s.input ← NIL;
      s.inputPos ← 0;
      };
    zone.FREE[@s];
    };

  Destroyer: ENTRY PROC [] = {
    ENABLE UNWIND => NULL;
    window: Window.Handle ← data.xmodem;
    IF data.xmodem = NIL THEN RETURN;
    data.xmodem ← NIL;
    Tool.Destroy[window]};

  Started: ENTRY PROCEDURE [data: DataHandle] RETURNS [started: BOOLEAN] = {
    RETURN[data.started]};
```

```
SetStarted: ENTRY PROCEDURE [data: DataHandle] = {data.started ← TRUE};

SetStopped: ENTRY PROCEDURE [data: DataHandle] = {data.started ← FALSE};

FileOverWrite: SIGNAL RETURNS [BOOLEAN] = CODE;

GetReadStream: PROCEDURE [filename: LONG STRING]
  RETURNS [streamHandle: MStream.Handle] =
  BEGIN
  streamReleaseData: MStream.ReleaseData ← [NIL, NIL];
  RETURN[MStream.ReadOnly[filename, streamReleaseData]];
  END;  --GetReadStream

GetWriteStream: PROCEDURE [filename: LONG STRING]
  RETURNS [streamHandle: MStream.Handle] =
  BEGIN
  DontOverWrite: BOOLEAN ← TRUE;
  access: MFile.Access = anchor;
  streamReleaseData: MStream.ReleaseData ← [NIL, NIL];
  RETURN[MStream.WriteOnly[filename, streamReleaseData, text]];
  END;  --GetWriteStream

FileExists: PROCEDURE [filename: LONG STRING] RETURNS [yes: BOOLEAN ← FALSE] =
  BEGIN
  tempHandle: MFile.Handle ← NIL;
  tempHandle ← MFile.Acquire[
    filename, anchor, [NIL, NIL] !
    MFile.Error => SELECT code FROM noSuchFile => CONTINUE; ENDCASE => NULL; ];
  IF tempHandle # NIL THEN tempHandle.Release[];
  yes ← tempHandle # NIL;
  END;  --GetWriteStream

XMDAbort: FormSW.ProcType = {
  IF iStarted[data] THEN {Process.Detach[FORK Destroyer[]]; RETURN};
  XModem.SetOperatorAbort[TRUE];
  IF getbyte # NIL THEN Process.Abort[getbyte];
  WHILE getbyte # NIL DO Process.Pause[Process.MsecToTicks[1]]; ENDLOOP;
  IF me # NIL THEN Process.Abort[me];
  WHILE me # NIL DO Process.Pause[Process.MsecToTicks[1]]; ENDLOOP;
  };

Init: PROC = {Exec.AddCommand["RS232XChat.", Main, NIL, Unload]; }.

Main: Exec.ExecProc = {};

Unload: Exec.ExecProc = {
  IF wh # NIL THEN Tool.Destroy[wh];
  wh ← NIL;
  h.RemoveCommand["RS232XChat."L];
  Heap.Delete[zone];
  };

Init[];


}.
```

```
-- File: SampleBWSApplicationImpl.mesa - last edit:
-- Breisacher.ES       10-May-85  8:50:40
-- guzik.ES            29-Mar-85 14:28:15

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

-- This is a sample Basic Workstation application.

-- The purpose of this application is to be an example client of most of the commonly used Basic Workstation interfaces. It demonstrates
use of Containee, XString, StarWindowShell, PropertySheet and FormWindow, Selection, XMessage, Context, Window, TIP. Several NSFile
operations are used also.

-- This application does some the rather uninteresting things, but it demonstrates the use of several Basic Workstation interfaces.
   -- When the user hits OPEN, it displays the contents of the file as text.
   -- When the user hits PROPS, it displays the file name, create date, etc. and lets the user change the file name.
   -- When the user moves or copies the current selection to this file, it tries to convert the current selection to a string and appends
the string to the contents of the file. If the current selection can be converted to a file, it uses the file's name as the string.

DIRECTORY
  ApplicationFolder USING [FindDescriptionFile, FromName],
  Atom USING [ATOM, MakeAtom, null],
  Attention USING [Post],
  Containee USING [ChangeProc, DataHandle, DefaultFileConvertProc, Error, GenericProc, GetCachedName, GetImplementation, Implementation,
ReturnTicket, SetImplementation, Signal, Ticket],
  Context USING [Create, Data, Find, Type, UniqueType],
  Courier USING [Error],
  Display USING [White],
  Event USING [AddDependency, AgentProcedure],
  Heap USING [Create],
  NSFile USING [Attribute, AttributesRecord, ClearAttributes, Close, Error, GetAttributes, Handle, nullHandle, nullReference,
OpenByReference, Reference, Type],
  NSFileName USING [Character, nameVersionPairSeparator],
  NSFileStream USING [Create, GetLength, Handle, SetLength],
  OptionFile USING [Error, GetBooleanValue, GetStringValue],
  ProductFactoring USING [DescribeOption, Enabled, ProductOption],
  ProductFactoringProducts USING [Star],
  Prototype USING [Create, Find],
  SampleBWSApplicationOps USING [GetMessageHandle, MakePropertySheet, kerrorMessage1, kNotEnabled, kprototypeFileName, kApplicationName,
kDisplayMessage, kMessageToDisplay, kWhereToDisplay, kwindow, kattention, kboth, WhereToDisplay],
  Selection USING [CanYouConvert, Convert, ConvertProc, Enumerate, EnumerationProc, Free],
  SimpleTextDisplay USING [StringIntoWindow, systemFontHeight],
  SimpleTextFont USING [AddClientDefinedCharacter],
  StarDesktop USING [GetCurrentDesktopFile],
  Stream USING [Delete, GetPosition, SetPosition],
  StarWindowShell USING [Create, CreateBody, GetBody, Handle, TransitionProc],
  TIP USING [CreateTable, Table],
  Window USING [Dims, Handle, Place],
  XChar USING [Character, null],
  XFormat USING [NSString, Object, Reader, StreamObject],
  XMessage USING [Get, Handle],
  XString USING [AppendChar, AppendReader, AppendSTRING, Character, CopyReader, Equivalent, FreeReaderBytes, FreeWriterBytes,
FromNSString, FromSTRING, NewWriterBody, nullReaderBody, Reader, ReaderBody, ReaderFromWriter, Writer, WriterBody],
  XToken USING [Filtered, FilterProcType, FreeTokenString, FreeStreamHandle, Handle, StreamToHandle];

SampleBWSApplicationImpl: PROGRAM
  IMPORTS ApplicationFolder, Atom, Attention, Containee, Context, Courier, Display, Event, Heap, NSFile, NSFileStream, OptionFile,
ProductFactoring, Prototype, Selection, SampleBWSApplicationOps, SimpleTextDisplay, SimpleTextFont, StarDesktop, StarWindowShell,
Stream, TIP, XFormat, XMessage, XString, XToken
  EXPORTS = BEGIN

  -- TYPEs

  MyData: TYPE = LONG POINTER TO MyDataObject;

  MyDataObject: TYPE = RECORD [
    string: XString.ReaderBody ← XString.nullReaderBody,
    fileStream: NSFileStream.Handle ← [NIL],
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL];

  -- Constants and data

  -- User options from OptionFile.
  displayMessage: BOOLEAN ← FALSE;
  messageToDisplay: XString.Reader ← NIL;
  whereToDisplay: SampleBWSApplicationOps.WhereToDisplay ← window;

  sampleIconFileType: NSFile.Type = 100100; -- arbitrary
    -- See [Igor]<BWSHacks>BWSHacksFileTypes.doc.

  sampleIconFileIndex: CARDINAL = 0;

  oldImpl: LONG POINTER TO Containee.Implementation ← NIL;

  sampleTIPTable: TIP.Table ← NIL;

  sampleApplicationPFOption: ProductFactoring.ProductOption = 0;
    -- 0 was chosen arbitrarily for this sample.
    -- A real application should obtain a real ProductOption!

  open,
  props,
  canYouTakeSelection,
  takeSelection,
  takeSelectionCopy,
  logon: Atom.ATOM ← Atom.null;
```

```
    true: BOOLEAN ← TRUE;
    false: BOOLEAN ← FALSE;

    zone: UNCOUNTED ZONE ← Heap.Create [initial: 1];

    context: Context.Type ← Context.UniqueType[];

    bodyWindowDims: Window.Dims = [600, 10000]; -- arbitrary

    -- Procedures

    CanITake: PROCEDURE RETURNS [yes: BOOLEAN] = {
      -- I can take anything that's a file (I'll get its file name and store it in the file) or a string.
      RETURN[
        Selection.CanYouConvert[target: file, enumeration: FALSE] OR
        Selection.CanYouConvert[target: file, enumeration: TRUE] OR
        Selection.CanYouConvert[target: string, enumeration: FALSE] OR
        Selection.CanYouConvert[target: string, enumeration: TRUE] ];
      };

    DestroyContext: PROC [mydata: MyData, window: Window.Handle] = {
      --XString.FreeReaderBytes [@mydata.messageToDisplay, zone];
      zone.FREE [@mydata];
      };

    FindOrCreateIconFile: PROCEDURE = {
      -- This procedure ensures that there is a file of the appropriate type in the prototype catalog.
      mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];
      name: XString.ReaderBody ← XMessage.Get [
        mh, SampleBWSApplicationOps.kprototypeFileName];
      version: CARDINAL = 1;
      IF (Prototype.Find [type: sampleIconFileType, version: version] = NSFile.nullReference) THEN
        NSFile.Close [Prototype.Create [name: @name, type: sampleIconFileType,
          version: version] ];
      };

    GenericProc: Containee.GenericProc = {
      IF ~ProductFactoring.Enabled [option: [
          product: ProductFactoringProducts.Star,
          productOption: sampleApplicationPFOption]] THEN {
        mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];
        rb: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kNotEnabled];
        ERROR Containee.Error [@rb];
        };
      SELECT atom FROM
        canYouTakeSelection => RETURN [
          IF CanITake[] THEN @true ELSE @false];
        takeSelection, -- we treat MOVE and COPY the same
        takeSelectionCopy => {RETURN [
          IF Take[data, changeProc, changeProcData] THEN @true ELSE @false]};
        open => RETURN [ MakeShell[data, changeProc, changeProcData] ];
        props => RETURN [ SampleBWSApplicationOps.MakePropertySheet[data, changeProc, changeProcData] ];
        ENDCASE => RETURN oldImpl.genericProc [atom, data, changeProc, changeProcData];
        -- Note that we call the old implementation for any atoms that we don't want to handle. In most cases, the old implementation will
be the default implementation supplied by ContaineeImpl, which will probably display an appropriate message to the user.
      };

    GetStream: PROCEDURE [data: Containee.DataHandle]
      RETURNS [fileStream: NSFileStream.Handle] = {
      BEGIN -- nested for catching NSFile errors
      file: NSFile.Handle ← NSFile.OpenByReference [data.reference !
        NSFile.Error, Courier.Error => GOTO ErrorExit];
      fileStream ← NSFileStream.Create [file !
        NSFile.Error, Courier.Error => {
          NSFile.Close [file ! NSFile.Error, Courier.Error => CONTINUE];
          GOTO ErrorExit} ];
      EXITS ErrorExit => RETURN [fileStream: [NIL]];
      END; -- nested
      };

    GetContext: PROCEDURE [body: Window.Handle] RETURNS [mydata: MyData] = {
      mydata ← Context.Find[context, body];
      IF mydata = NIL THEN ERROR; -- just in case.
      RETURN [mydata];
      };

    GetOptionsAtLogon: PROCEDURE = {
      desktopRef: NSFile.Reference;
      [] ← Event.AddDependency [agent: LogonEvent,
        myData: NIL, event: logon];
      IF (desktopRef ← StarDesktop.GetCurrentDesktopFile []) # NSFile.nullReference THEN {
        -- If the desktop is NOT null, then a user's already logged on,
        --  i.e. we got loaded after logon.
        -- So we go read the options immediately by calling our
        --  Event.AgentProcedure directly.
        desktop: NSFile.Handle ← NSFile.OpenByReference [desktopRef];
        [] ← LogonEvent [event: logon,
          eventData: LOOPHOLE [desktop], myData: NIL];
        NSFile.Close [desktop];
        };
      };

    Init: PROCEDURE = {
      InitAtoms[];
      InitProductFactoring[];
      FindOrCreateIconFile[];
      InitTIPTable[];
```

```
    SetImplementation[];
    GetOptionsAtLogon[];
    };

  InitAtoms: PROCEDURE = {
    -- This gets all the strings out of the global frame.
    open ← Atom.MakeAtom["Open"L];
    props ← Atom.MakeAtom["Props"L];
    canYouTakeSelection ← Atom.MakeAtom["CanYouTakeSelection"L];
    takeSelection ← Atom.MakeAtom["TakeSelection"L];
    takeSelectionCopy ← Atom.MakeAtom["TakeSelectionCopy"L];
    logon ← Atom.MakeAtom["Logon"L];
    };


  InitProductFactoring: PROCEDURE = {
    mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];
    rb: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kApplicationName];
    ProductFactoring.DescribeOption [
      option: [
        product: ProductFactoringProducts.Star,
        productOption: sampleApplicationPFOption],
      desc: @rb];
    };

  InitSmallPicture: PROCEDURE RETURNS [XString.Character] = {
    -- This demonstrates a feature of SimpleTextFont that allows any client-defined bitmap to be made into an ordinary character. The
height of the client's bitmap must be less than SimpleTextDisplay.systemFontHeight.
    bits: ARRAY [0..13) OF WORD ← [
      177770B, 120050B, 177770B, 120050B, 120050B, 120050B,
      120050B, 120050B, 120050B, 120050B, 177770B, 120050B, 177770B];
    RETURN [ SimpleTextFont.AddClientDefinedCharacter [
      width: 13, height: 13, bitsPerLine: 16, bits: @bits] ];
    };

  InitTIPTable: PROCEDURE = {
    separator: XChar.Character = LOOPHOLE [NSFileName.nameVersionPairSeparator];
    pathName: XString.WriterBody ← XString.NewWriterBody[40, zone];
    AppendTIPFileName [@pathName];
    sampleTIPTable ← TIP.CreateTable [file: XString.ReaderFromWriter [@pathName]];
    XString.FreeWriterBytes [@pathName];
    };

  AppendTIPFileName: PROCEDURE [writer: XString.Writer] = {
    separator: XChar.Character = LOOPHOLE [NSFileName.nameVersionPairSeparator];
    internalName: XString.ReaderBody ← XString.FromSTRING ["SampleBWSApplication"L];
    tipFile: XString.ReaderBody ← XString.FromSTRING ["TIPFile"L];
    folderHandle: NSFile.Handle;
    folderRef: NSFile.Reference ← ApplicationFolder.FromName [@internalName];

    AppendName: PROCEDURE [value: XString.Reader] = {
      XString.AppendReader [to: writer, from: value];
      };

    IF folderRef = NSFile.nullReference THEN {
      XString.AppendSTRING [writer, "SampleBWSApplication.TIP"L];
      RETURN};
    -- ELSE --
    folderHandle ← NSFile.OpenByReference [folderRef];
    AppendFolderName [folderHandle, writer];
    XString.AppendChar [to: writer, c: separator];
    OptionFile.GetStringValue [section: @internalName, entry: @tipFile,
      callBack: AppendName,
      file: ApplicationFolder.FindDescriptionFile [folderHandle]];
    NSFile.Close [folderHandle];
    };

  AppendFolderName: PROCEDURE [applFolder: NSFile.Handle, writer: XString.Writer] = {
    attrs: NSFile.AttributesRecord;
    rb: XString.ReaderBody;
    NSFile.GetAttributes[applFolder, [interpreted: [name : TRUE]], @attrs];
    rb ← XString.FromNSString [attrs.name];
    XString.AppendReader [writer, @rb];
    NSFile.ClearAttributes[@attrs];
    };

  LogonEvent: Event.AgentProcedure = {
    <<[event: Event.EventType, eventData: LONG POINTER,
    myData: LONG POINTER]
    RETURNS [remove: BOOLEAN ← FALSE, veto: BOOLEAN ← FALSE]>>
    mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];

    CopyMessageToDisplay: PROCEDURE [value: XString.Reader] = {
      messageToDisplay ← XString.CopyReader [value, zone]};

    GetWhereToDisplay: PROCEDURE [value: XString.Reader] = {
      window: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kwindow];
      attention: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kattention];
      both: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kboth];
      whereToDisplay ← SELECT TRUE FROM
        XString.Equivalent [value, @window] => window,
        XString.Equivalent [value, @attention] => attention,
        XString.Equivalent [value, @both] => both,
        ENDCASE => window;
      };

    section: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kApplicationName];
```

```
    entry: XString.ReaderBody;

    -- Reset to defaults
    displayMessage ← FALSE;
    XString.FreeReaderBytes [messageToDisplay, zone];
    messageToDisplay ← NIL;
    whereToDisplay ← window;

    entry ← XMessage.Get [mh, SampleBWSApplicationOps.kDisplayMessage];
    displayMessage ← OptionFile.GetBooleanValue [@section, @entry !
      OptionFile.Error => CONTINUE];

    entry ← XMessage.Get [mh, SampleBWSApplicationOps.kMessageToDisplay];
    OptionFile.GetStringValue [@section, @entry, CopyMessageToDisplay !
      OptionFile.Error => CONTINUE];

    entry ← XMessage.Get [mh, SampleBWSApplicationOps.kWhereToDisplay];
    OptionFile.GetStringValue [@section, @entry, GetWhereToDisplay !
      OptionFile.Error => CONTINUE];
    };

  MakeShell: PROCEDURE [
    data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL]
    RETURNS [shell: StarWindowShell.Handle] = {
    body: Window.Handle ← NIL;
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    mydata: MyData ← NIL;

    fileStream: NSFileStream.Handle ← GetStream [data];
    IF fileStream = NIL THEN RETURN [shell: [NIL]]; -- couldn't open file

    [name, ticket] ← Containee.GetCachedName [data]; -- This retrieves the file name from the file.

    shell ← StarWindowShell.Create [name: @name, transitionProc: TransitionProc,
      sleeps: TRUE];

    Containee.ReturnTicket [ticket];

    body ← StarWindowShell.CreateBody [sws: shell, repaintProc: RepaintBody,
      box: [ place: [0,0], dims: bodyWindowDims ] ];
      -- Note we don't demonstrate TIP.NotifyProc here.
      -- See SampleBWSTool for an example TIP.NotifyProc.

    -- Allocate context data and "hang it off the body window"
    --  by using Context.Create.
    Context.Create [
      type: context,
      data: (mydata ← zone.NEW[MyDataObject ← [
        fileStream: fileStream,
        string: ReadFile [fileStream],
        changeProc: changeProc,
        changeProcData: changeProcData ] ]),
      proc: DestroyContext,
      window: body];
    };


  ReadFile: PROCEDURE [fileStream: NSFileStream.Handle]
    RETURNS [string: XString.ReaderBody] = {
    xTokenH: XToken.Handle ← NIL;
    NullFilter: XToken.FilterProcType = {RETURN [inClass: c # XChar.null]}; -- We read everything up to a null character.
    xTokenH ← XToken.StreamToHandle [fileStream];
    string ← XToken.Filtered [h: xTokenH, data: NIL,
      filter: NullFilter, skip: none, temporary: FALSE];
    [] ← XToken.FreeStreamHandle [xTokenH];
    };

  RepaintBody: PROCEDURE [body: Window.Handle] = {
    -- This is a very "dumb" display proc.
    -- It repaints the whole window every time it's called.
    mydata: MyData = GetContext [body];
    lines: CARDINAL ← 0;
    y: INTEGER ← 10;
    Display.White [body, [ [0,0], bodyWindowDims ] ]; -- whole window
    IF displayMessage THEN {
      IF whereToDisplay = window
        OR whereToDisplay = both THEN {
          [lines: lines] ← SimpleTextDisplay.StringIntoWindow [
            string: messageToDisplay,
            window: body,
            place: [x: 10, y: y],
            lineWidth: bodyWindowDims.w,
            maxNumberOfLines: 1000 -- arbitrary --];
          y ← y + (lines * SimpleTextDisplay.systemFontHeight);
          };
      IF whereToDisplay = attention
        OR whereToDisplay = both THEN
          Attention.Post [messageToDisplay];
      };
    [] ← SimpleTextDisplay.StringIntoWindow [
      string: @mydata.string,
      window: body,
      place: [x: 10, y: y],
      lineWidth: bodyWindowDims.w,
```

```
      maxNumberOfLines: 1000 -- arbitrary --];
    };

  SetImplementation: PROCEDURE = {
    mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];
    newImpl: Containee.Implementation ← Containee.GetImplementation [sampleIconFileType];
    oldImpl ← zone.NEW[Containee.Implementation ← newImpl];
    newImpl.convertProc ← Containee.DefaultFileConvertProc;
    newImpl.genericProc ← GenericProc;
    newImpl.name ← XMessage.Get [
      mh, SampleBWSApplicationOps.kApplicationName];
    [] ← Containee.SetImplementation [sampleIconFileType, newImpl];
    };

  Take: PROC [data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL]
    RETURNS [didIt: BOOLEAN] = {

    TakeFile: Selection.EnumerationProc =
      <<[element: Selection.Value, data: Selection.RequestorData]
        RETURNS [stop: BOOLEAN ← FALSE]>>
      BEGIN
      mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];
      ref: LONG POINTER TO NSFile.Reference ← element.value;
      fh: NSFile.Handle ← NSFile.nullHandle;
      attributes: NSFile.AttributesRecord;
      fh ← NSFile.OpenByReference [ref↑ !
        NSFile.Error, Courier.Error => CONTINUE];
      IF fh = NSFile.nullHandle THEN {
        mh: XMessage.Handle = SampleBWSApplicationOps.GetMessageHandle[];
        rb: XString.ReaderBody ← XMessage.Get [mh, SampleBWSApplicationOps.kerrorMessage1];
        SIGNAL Containee.Signal [@rb];
        RETURN};
      NSFile.GetAttributes[fh, [interpreted: [name: TRUE] ], @attributes];
      XFormat.NSString [@xformatObject, attributes.name];
      NSFile.ClearAttributes[@attributes];
      NSFile.Close[fh];
      Selection.Free[@element];
      END;

    TakeString: Selection.EnumerationProc =
      <<[element: Selection.Value, data: Selection.RequestorData]
        RETURNS [stop: BOOLEAN ← FALSE]>>
      BEGIN
      XFormat.Reader [@xformatObject, LOOPHOLE [element.value]];
      Selection.Free [@element];
      END;

    xformatObject: XFormat.Object;
    fileStream: NSFileStream.Handle ← GetStream [data];
    IF fileStream = NIL THEN RETURN [FALSE]; -- couldn't open file
    Stream.SetPosition [fileStream, NSFileStream.GetLength [fileStream] ]; -- position the stream at the end of the file.
    xformatObject ← XFormat.StreamObject [fileStream];

    SELECT TRUE FROM
      Selection.CanYouConvert [target: file, enumeration: TRUE] =>
        [] ← Selection.Enumerate [TakeFile, file, NIL];
      Selection.CanYouConvert [target: file, enumeration: FALSE] =>
        [] ← TakeFile[Selection.Convert[file], NIL];
      Selection.CanYouConvert [target: string, enumeration: TRUE] =>
        [] ← Selection.Enumerate [TakeString, string, NIL];
      Selection.CanYouConvert [target: string, enumeration: FALSE] =>
        [] ← TakeString[Selection.Convert[string], NIL];
      ENDCASE;

    NSFileStream.SetLength [[fileStream], Stream.GetPosition[fileStream]];
    Stream.Delete [fileStream]; -- Closes the file.

    -- We call the changeProc so that if the shell was being saved,
    --  this will cause it to be destroyed so that next time the icon
    --  is opened, we will read the file again.
    IF changeProc # NIL THEN
      changeProc [changeProcData, data, [interpreted: [name: TRUE] ] ];

    RETURN [TRUE];
    };

  TransitionProc: StarWindowShell.TransitionProc = {
    <<[sws: StarWindowShell.Handle, state: StarWindowShell.State]>>
    IF state = dead THEN
      BEGIN
      mydata: MyData ← GetContext [ StarWindowShell.GetBody [sws] ];
      [] ← XToken.FreeTokenString [@mydata.string];
      Stream.Delete [mydata.fileStream]; -- closes the file
      -- We always call the changeProc, even if nothing changed.
      IF mydata.changeProc # NIL THEN
        mydata.changeProc [changeProcData: mydata.changeProcData, noChanges: TRUE];
      END;
    };

  -- Main line code

  Init[]; -- Note that the message impl must be started first!

  END.
```

```
-- File: SampleBWSApplicationMessagesImpl.mesa - last edit:
-- Breisacher.ES        1-Mar-85 10:42:52
-- Created by FormWindowLayoutTool on September 20, 84 12:50:30

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  XMessage USING [AllocateMessages, ClientData, Handle, Messages, MsgEntry, RegisterMessages],
  XString USING [FromSTRING],
  SampleBWSApplicationOps;

SampleBWSApplicationMessagesImpl: PROGRAM
  IMPORTS XMessage, XString
  EXPORTS SampleBWSApplicationOps = {OPEN XS: XString, Ops: SampleBWSApplicationOps;

  h: XMessage.Handle ← NIL;

  DeleteMessages: PROCEDURE [clientData: XMessage.ClientData] = {};

  GetMessageHandle: PUBLIC PROCEDURE RETURNS [XMessage.Handle] = {RETURN[h]};

  InitMessages: PROCEDURE = {
    me: LONG STRING = "SampleBWSApplication Property Sheet"L;
    msgArray: ARRAY Ops.MessageKey OF XMessage.MsgEntry ← [
      psTitle: [
        msgKey: Ops.kpsTitle,
        msg: XS.FromSTRING["SampleBWSApplication Properties"L],
        type: pSheetItem,
        id: 1],
      psTagname: [
        msgKey: Ops.kpsTagname,
        msg: XS.FromSTRING["Name"L],
        type: pSheetItem,
        id: 2],
      psTagfiletype: [
        msgKey: Ops.kpsTagfiletype,
        msg: XS.FromSTRING["File type: "L],
        type: pSheetItem,
        id: 3],
      psTagcreatedOn: [
        msgKey: Ops.kpsTagcreatedOn,
        msg: XS.FromSTRING["Created On: "L],
        type: pSheetItem,
        id: 4],
      psTagsizeInPages: [
        msgKey: Ops.kpsTagsizeInPages,
        msg: XS.FromSTRING["Size In Pages: "L],
        type: pSheetItem,
        id: 5],
      psTagsizeInBytes: [
        msgKey: Ops.kpsTagsizeInBytes,
        msg: XS.FromSTRING["Size In Bytes: "L],
        type: pSheetItem,
        id: 6],
      prototypeFileName: [
        msgKey: Ops.kprototypeFileName,
        msg: XS.FromSTRING["Sample Icon File"L],
        type: others,
        id: 7],
      errorMessage1: [
        msgKey: Ops.kerrorMessage1,
        msg: XS.FromSTRING["Unable to open file!"L],
        type: errorMsg,
        id: 8],
      applicationName: [
        msgKey: Ops.kApplicationName,
        msg: XS.FromSTRING["Sample Application"L],
        type: others,
        id: 9],
      displayMessage: [
        msgKey: Ops.kDisplayMessage,
        msg: XS.FromSTRING["Display Message"L],
        type: others,
        id: 10],
      messageToDisplay: [
        msgKey: Ops.kMessageToDisplay,
        msg: XS.FromSTRING["Message To Display"L],
        type: others,
        id: 11],
      whereToDisplay: [
        msgKey: Ops.kWhereToDisplay,
        msg: XS.FromSTRING["Where To Display Message"L],
        type: others,
        id: 12],
      window: [
        msgKey: Ops.kwindow,
        msg: XS.FromSTRING["In Open Window"L],
        type: others,
        id: 13],
      attention: [
        msgKey: Ops.kattention,
        msg: XS.FromSTRING["In Attention Window"L],
        type: others,
        id: 14],
      both: [
        msgKey: Ops.kboth,
        msg: XS.FromSTRING["Both"L],
```

```
        type: others,
        id: 15],
      notEnabled: [
        msgKey: Ops.kNotEnabled,
        msg: XS.FromSTRING["Sample Application is not Product Factored."L],
        type: others,
        id: 16]
      ];

    messages: XMessage.Messages ← DESCRIPTOR [
      LOOPHOLE [
        msgArray,
        ARRAY [0..Ops.MessageKey.LAST.ORD] OF XMessage.MsgEntry] ];
    h ← XMessage.AllocateMessages [
      applicationName: "SampleBWSApplication"L,
      maxMessages: Ops.MessageKey.LAST.ORD + 1,
      clientData: NIL,
      proc: DeleteMessages ];
    XMessage.RegisterMessages[
      h: h,
      messages: messages,
      stringBodiesAreReal: FALSE];
    };

-- Mainline code

InitMessages[];

}...
```

```
-- File: SampleBWSApplicationOps.mesa - last edit:
-- Breisacher.ES        1-Mar-85 10:39:32

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Containee USING [ChangeProc, DataHandle],
  StarWindowShell USING [Handle],
  XMessage USING [Handle, MsgKey];

SampleBWSApplicationOps: DEFINITIONS = {

  -- Procedures

  MakePropertySheet: PROC [
    data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL]
    RETURNS [pSheetShell: StarWindowShell.Handle];

  FreeData: PROC;

  -- UserProfile enumerated option choices

  WhereToDisplay: TYPE = {window, attention, both};

  -- Messages

  MessageKey: TYPE = {applicationName, displayMessage, messageToDisplay, whereToDisplay, window, attention, both, psTitle, psTagname,
psTagfiletype, psTagcreatedOn, psTagsizeInPages, psTagsizeInBytes, prototypeFileName, errorMessage1, notEnabled};

  kApplicationName: XMessage.MsgKey = MessageKey.applicationName.ORD;
  kDisplayMessage: XMessage.MsgKey = MessageKey.displayMessage.ORD;
  kMessageToDisplay: XMessage.MsgKey = MessageKey.messageToDisplay.ORD;
  kWhereToDisplay: XMessage.MsgKey = MessageKey.whereToDisplay.ORD;
  kwindow: XMessage.MsgKey = MessageKey.window.ORD;
  kattention: XMessage.MsgKey = MessageKey.attention.ORD;
  kboth: XMessage.MsgKey = MessageKey.both.ORD;
  kpsTitle: XMessage.MsgKey = MessageKey.psTitle.ORD;
  kpsTagname: XMessage.MsgKey = MessageKey.psTagname.ORD;
  kpsTagfiletype: XMessage.MsgKey = MessageKey.psTagfiletype.ORD;
  kpsTagcreatedOn: XMessage.MsgKey = MessageKey.psTagcreatedOn.ORD;
  kpsTagsizeInPages: XMessage.MsgKey = MessageKey.psTagsizeInPages.ORD;
  kpsTagsizeInBytes: XMessage.MsgKey = MessageKey.psTagsizeInBytes.ORD;
  kprototypeFileName: XMessage.MsgKey = MessageKey.prototypeFileName.ORD;
  kerrorMessage1: XMessage.MsgKey = MessageKey.errorMessage1.ORD;
  kNotEnabled: XMessage.MsgKey = MessageKey.notEnabled.ORD;

  -- Obtain Handle for System Required Messages:

  GetMessageHandle: PROCEDURE RETURNS[h: XMessage.Handle ];

  }...
```

```
-- File: SampleBWSApplicationPSheet.mesa - last edit:
-- Breisacher.ES       10-May-85  8:39:45
-- Created by FormWindowLayoutTool on September 20, 84 12:50:19

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Containee USING [ChangeProc, Data, DataHandle],
  FormWindow USING [AppendItem, AppendLine, DoneLookingAtTextItemValue, HasAnyBeenChanged, HasBeenChanged, ItemKey, LayoutProc, Line,
LookAtTextItemValue, MakeItemsProc, MakeTagOnlyItem, MakeTextItem, SetTabStops, TabStops],
  Heap USING [Create],
  NSFile USING [Attribute, AttributesRecord, ChangeAttributes, ClearAttributeList, ClearAttributes, Close, Error, GetAttributes,
GetReference, Handle, OpenByReference, Selections, String],
  NSString USING [String],
  PropertySheet USING [Create, MenuItemProc],
  SampleBWSApplicationOps USING [GetMessageHandle, kpsTagcreatedOn, kpsTagfiletype, kpsTagname, kpsTagsizeInBytes, kpsTagsizeInPages,
kpsTitle],
  StarWindowShell USING [Handle],
  Window USING [Dims, Handle, Object, Place],
  XFormat USING [Date, Decimal, Handle, Object, Reader, WriterObject],
  XMessage USING [Get, Handle],
  XString USING [ClearWriter, FreeWriterBytes, FromNSString, NewWriterBody, nullReaderBody, NSStringFromReader, Reader, ReaderBody,
ReaderFromWriter, WriterBody];

SampleBWSApplicationPSheet: PROGRAM
  IMPORTS  FormWindow, Heap, NSFile, PropertySheet, SampleBWSApplicationOps, XFormat, XMessage, XString
  EXPORTS SampleBWSApplicationOps = {OPEN Ops: SampleBWSApplicationOps;

  -- TYPEs

  Items: TYPE = {name, filetype, createdOn, sizeInPages, sizeInBytes};

  DataObject: TYPE = RECORD [
    fh: NSFile.Handle,
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL];

  Data: TYPE = LONG POINTER TO DataObject;

  -- Global data

  localZone: UNCOUNTED ZONE ← Heap.Create [initial: 1];

  tabStopInterval: CARDINAL = 50;
  spaceAboveLine: CARDINAL = 15;

  size: Window.Dims ← [400, 350];
  placeToDisplay: Window.Place ← [500, 400];

  -- Procedures

  MakePropertySheet: PUBLIC PROC [
    data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL]
    RETURNS [pSheetShell: StarWindowShell.Handle] = {
    BEGIN
    mh: XMessage.Handle = Ops.GetMessageHandle[];
    mydata: Data ← localZone.NEW [DataObject ← [
      fh: NSFile.OpenByReference[data.reference
        ! NSFile.Error => GOTO ErrorExit],
      changeProc:  changeProc,
      changeProcData: changeProcData]];
    title: XString.ReaderBody ← XMessage.Get[mh, Ops.kpsTitle];

    pSheetShell ← PropertySheet.Create [
      formWindowItems: MakeItems,
      menuItemProc: MenuItemProc,
      menuItems: [done: TRUE, cancel: TRUE],
      size: size,
      title: @title,
      placeToDisplay: placeToDisplay,
      formWindowItemsLayout: DoLayout,
      display: FALSE,
      clientData: mydata];
    EXITS ErrorExit => pSheetShell ← [NIL];
    END;
    };

  MakeItems: FormWindow.MakeItemsProc = {
    mh: XMessage.Handle = Ops.GetMessageHandle[];
    mydata: Data = clientData;
    maxLength: CARDINAL = 50; -- arbitrary
    wb: XString.WriterBody ← XString.NewWriterBody [maxLength, localZone];
    xfo: XFormat.Object ← XFormat.WriterObject [@wb];
    rb: XString.ReaderBody ← XString.nullReaderBody;

    attributes: NSFile.AttributesRecord;
    attributeSelections: NSFile.Selections ← [
      interpreted: [
        name: TRUE,
        type: TRUE,
        createdOn: TRUE,
        sizeInPages: TRUE,
        sizeInBytes: TRUE] ];
    NSFile.GetAttributes[mydata.fh, attributeSelections, @attributes];
```

```
    BEGIN
    initString: XString.ReaderBody ← XString.FromNSString[attributes.name];
    rb ← XMessage.Get[mh, Ops.kpsTagname];
    FormWindow.MakeTextItem [
      window: window,
      myKey: Items.name.ORD,
      tag: @rb,
      width: 200,
      initString: @initString ];
    END;

    rb ← XMessage.Get [mh, Ops.kpsTagfiletype];
    XFormat.Reader [ @xfo,  @rb];
    XFormat.Decimal [ @xfo, attributes.type ];
    FormWindow.MakeTagOnlyItem [
      window: window,
      myKey: Items.filetype.ORD,
      tag: XString.ReaderFromWriter[@wb] ];

    rb ← XMessage.Get [mh, Ops.kpsTagcreatedOn];
    XString.ClearWriter [@wb];
    XFormat.Reader [ @xfo, @rb ];
    XFormat.Date [ @xfo, attributes.createdOn ];
    FormWindow.MakeTagOnlyItem [
      window: window,
      myKey: Items.createdOn.ORD,
      tag: XString.ReaderFromWriter[@wb] ];

    rb ← XMessage.Get [mh, Ops.kpsTagsizeInPages];
    XString.ClearWriter [@wb];
    XFormat.Reader [ @xfo, @rb ];
    XFormat.Decimal [ @xfo, attributes.sizeInPages ];
    FormWindow.MakeTagOnlyItem [
      window: window,
      myKey: Items.sizeInPages.ORD,
      tag: XString.ReaderFromWriter[@wb] ];

    rb ← XMessage.Get [mh, Ops.kpsTagsizeInBytes];
    XString.ClearWriter [@wb];
    XFormat.Reader [ @xfo, @rb ];
    XFormat.Decimal [ @xfo, attributes.sizeInBytes ];
    FormWindow.MakeTagOnlyItem [
      window: window,
      myKey: Items.sizeInBytes.ORD,
      tag: XString.ReaderFromWriter[@wb] ];

    XString.FreeWriterBytes[@wb];
    NSFile.ClearAttributes[@attributes];
    };

  DoLayout: FormWindow.LayoutProc = {
    line: FormWindow.Line;
    FormWindow.SetTabStops[window, [fixed[tabStopInterval]]];
    -- Line 1
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [window, Items.name.ORD, line];
    -- Line 2
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [window, Items.filetype.ORD, line];
    -- Line 3
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [window, Items.createdOn.ORD, line];
    -- Line 4
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [window, Items.sizeInPages.ORD, line];
    -- Line 5
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [window, Items.sizeInBytes.ORD, line];
    };

  MenuItemProc: PropertySheet.MenuItemProc = {
    mydata: Data ← clientData;
    fh: NSFile.Handle = mydata.fh;
    SELECT menuItem FROM
      done => {
        ok ← ApplyAnyChanges[formWindow, mydata].ok;
        NSFile.Close[fh];
        localZone.FREE[@mydata];
        RETURN[ok]};
      cancel => {
        data: Containee.Data ← [ NSFile.GetReference [mydata.fh] ];
        IF mydata.changeProc # NIL THEN
          mydata.changeProc[changeProcData: mydata.changeProcData,
            data: @data, changedAttributes: [], noChanges: TRUE];
          NSFile.Close[fh];
          localZone.FREE[@mydata];
          RETURN[ok: TRUE];
          };
      ENDCASE;
    RETURN[ok: FALSE];
    };

  ApplyAnyChanges: PROC [fw: Window.Handle, mydata: Data] RETURNS [ok: BOOLEAN] = {
    attrList: ARRAY [0..1) OF NSFile.Attribute;
    changedAttributes: NSFile.Selections ← [];
    ctChangedAttrs: CARDINAL ← 0;
```

```
  IF ~FormWindow.HasAnyBeenChanged [fw] THEN {
    IF mydata.changeProc # NIL THEN
      mydata.changeProc[changeProcData: mydata.changeProcData, noChanges: TRUE];
    RETURN [ok: TRUE]};

  FOR myItem: Items IN Items DO
    itemKey: FormWindow.ItemKey = myItem.ORD;
    IF ~FormWindow.HasBeenChanged [fw, itemKey] THEN LOOP;
    SELECT myItem FROM
      name => {
        rb: XString.ReaderBody ← FormWindow.LookAtTextItemValue [fw, itemKey];
        ns: NSString.String ← XString.NSStringFromReader [@rb, localZone];
        FormWindow.DoneLookingAtTextItemValue [fw, itemKey];
        attrList[ctChangedAttrs] ← [name[ns]];
        changedAttributes.interpreted[name] ← TRUE; };
      ENDCASE;
    ctChangedAttrs ← ctChangedAttrs + 1;
    ENDLOOP;

  IF ctChangedAttrs > 0 THEN {
    data: Containee.Data ← [ NSFile.GetReference [mydata.fh] ];
    NSFile.ChangeAttributes [mydata.fh, DESCRIPTOR[@attrList, ctChangedAttrs]];
    NSFile.ClearAttributeList [DESCRIPTOR[@attrList, ctChangedAttrs]];
    IF mydata.changeProc # NIL THEN
      mydata.changeProc[mydata.changeProcData, @data, changedAttributes];
    }
  ELSE -- Note we call the changeProc even when there are no changes.
    IF mydata.changeProc # NIL THEN
      mydata.changeProc[changeProcData: mydata.changeProcData, noChanges: TRUE];

  RETURN [ok: TRUE];
  };

}...
```

```
-- File: SampleBWSTool.mesa - last edit:
-- Breisacher.ES       21-Jun-84 16:23:34

-- This is a sample "tool" that can be used as a template for building test programs for the Basic Workstation.  Note that no special
files or file types are needed to create such a "tool".
-- It simply adds a menu item to the attention window menu.  When this menu item is bugged, the MenuProc creates a StarWindowShell with a
single body window in it.  Multiple instances of the "tool" can be created by invoking the menu item more than once.
-- The body window has simple display and notify procs. The last mouse button pushed by the user (with the mouse in the body window) is
kept track of and displayed in the body window. This is done solely to demonstrate the use of the Context interface.
-- Several menu items are placed in the header of the StarWindowShell.

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Attention USING [AddMenuItem, Post],
  Context USING [Create, Data, Find, Type, UniqueType],
  Display USING [replaceFlags],
  Heap USING [systemZone],
  MenuData USING [CreateItem, CreateMenu, ItemHandle, MenuHandle, MenuProc],
  SimpleTextDisplay USING [StringIntoWindow],
  StarWindowShell USING [Create, CreateBody, GetBody, GetZone, Handle, Push, SetRegularCommands],
  TIP USING [NotifyProc, Results],
  Window USING [Dims, Handle, InvalidateBox, Object, Place, Validate],
  XFormat USING [Char, Decimal, Handle, Object, String, WriterObject],
  XString USING [FromSTRING, NewWriterBody, ReaderBody, ReaderFromWriter, WriterBody];

SampleBWSTool: PROGRAM
  IMPORTS Atom, Attention, Context, Heap, MenuData, SimpleTextDisplay, StarWindowShell, Window, XFormat, XString = BEGIN

  -- TYPEs

  Data: TYPE = LONG POINTER TO DataObject;

  DataObject: TYPE = RECORD [
    lastMouseButton: PointOrAdjust,
    place: Window.Place ← [0,0] ];

  PointOrAdjust: TYPE = {point, adjust, neither};

  -- Constants

  bodyWindowDims: Window.Dims = [1000, 1000];

  sysZ: UNCOUNTED ZONE = Heap.systemZone;

  -- Data

  context: Context.Type ← Context.UniqueType[];

  pointDown, adjustDown: Atom.ATOM ← Atom.null;

  -- Procedures

  DestroyContext: PROC [data: Data, window: Window.Handle] = {
    -- Note that since Data was allocated out of the
    -- systemZone, this procedure is unnecessary, but it is
    -- included here as an example of a Context.DestroyProcType.
    -- The default Context.DestroyProcType assumes the data was
    -- allocated out of the systemZone and frees it from there.
    sysZ.FREE [@data];
    };

  GetContext: PROC [body: Window.Handle] RETURNS [data: Data] = {
    data ← Context.Find[context, body];
    IF data = NIL THEN ERROR; -- just in case.
    RETURN [data];
    };

  Init: PROC = {
    sampleTool: XString.ReaderBody ← XString.FromSTRING["Sample Tool"L];
    Attention.AddMenuItem [
      MenuData.CreateItem [
        zone: sysZ,
        name: @sampleTool,
        proc: MenuProc] ];
    };

  InitAtoms: PROC = {
    pointDown ← Atom.MakeAtom["PointDown"L];
    adjustDown ← Atom.MakeAtom["AdjustDown"L];
    };

  MenuProc: MenuData.MenuProc = {
    another: XString.ReaderBody ← XString.FromSTRING["Another"L];
    repaint: XString.ReaderBody ← XString.FromSTRING["Repaint"L];
    post: XString.ReaderBody ← XString.FromSTRING["Post A Message"L];
    sampleTool: XString.ReaderBody ← XString.FromSTRING["Sample Tool"L];

    -- Create the StarWindowShell.
    shell: StarWindowShell.Handle = StarWindowShell.Create [name: @sampleTool];

    -- Create a body window inside the StarWindowShell.
    body: Window.Handle = StarWindowShell.CreateBody [
      sws: shell,
      box: [ [0,0], bodyWindowDims ],
      repaintProc: Redisplay,
      bodyNotifyProc: NotifyProc ];
```

```
    -- Create some menu items.
    z: UNCOUNTED ZONE ← StarWindowShell.GetZone [shell];
    items: ARRAY [0..3) OF MenuData.ItemHandle ← [
      MenuData.CreateItem [zone: z, name: @another, proc: MenuProc],
      MenuData.CreateItem [zone: z, name: @repaint, proc: RepaintMenuProc],
      MenuData.CreateItem [zone: z, name: @post, proc: Post]
      ];
    myMenu: MenuData.MenuHandle = MenuData.CreateMenu [
      zone: z,
      title: NIL,
      array: DESCRIPTOR [items]];
    StarWindowShell.SetRegularCommands [sws: shell, commands: myMenu];

    -- Allocate data and "hang it off the body window" by using Context.Create.
    Context.Create [
      type: context,
      data: sysZ.NEW[DataObject ← [neither] ],
      proc: DestroyContext, -- This is unnecessary, but see the comment in DestroyContext.
      window: body];

    -- Put the StarWindowShell on the screen.
    StarWindowShell.Push [shell];
    };

  NotifyProc: TIP.NotifyProc = {
    data: Data ← GetContext [window];
    FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
      WITH z: input SELECT FROM
        coords => data.place ← z.place;
        atom => SELECT z.a FROM
          pointDown => data.lastMouseButton ← point;
          adjustDown => data.lastMouseButton ← adjust;
          ENDCASE;
        ENDCASE; -- WITH z: input
      ENDLOOP;
    Redisplay [window];
    };

  Post: MenuData.MenuProc = {
    msg: XString.ReaderBody ← XString.FromSTRING ["This is a sample attention window message."L];
    Attention.Post [@msg];
    };


  Redisplay: PROC [window: Window.Handle] = {
    data: Data ← GetContext [window];

    writerBody: XString.WriterBody ← XString.NewWriterBody [
      maxLength: 250, z: sysZ];
    xfo: XFormat.Object ← XFormat.WriterObject [w: @writerBody];

    XFormat.String [h: @xfo, s: "This is a sample string displayed in a body window of a StarWindowShell using
SimpleTextDisplay.StringIntoWindow.\n\nThe last mouse button pushed was: "L];
    XFormat.String [h: @xfo, s: SELECT data.lastMouseButton FROM
      point => "Point"L,
      adjust => "Adjust"L,
      ENDCASE => "Neither"L];
    XFormat.String [h: @xfo, s: " and the mouse was at window relative location: [x: "L];
    XFormat.Decimal [h: @xfo, n: data.place.x];
    XFormat.String [h: @xfo, s: ", y: "L];
    XFormat.Decimal [h: @xfo, n: data.place.y];
    XFormat.Char [h: @xfo, char: '].ORD ]; -- Note easy way to construct an XString.Character

    [] ← SimpleTextDisplay.StringIntoWindow [
      string: XString.ReaderFromWriter [@writerBody],
      window: window,
      place: [10,10],
      lineWidth: 300, -- arbitrary
      maxNumberOfLines: 10, -- arbitrary
      flags: Display.replaceFlags ];
    };

  RepaintMenuProc: MenuData.MenuProc = {
    body: Window.Handle = StarWindowShell.GetBody[[window]];
    Window.InvalidateBox[body, [[0, 0], [30000, 30000] ]];
    Window.Validate[body]; };

  -- Main line code

  Init[];
  InitAtoms[];

  END.
```

```
-- File: SampleMsgFileImpl.mesa - last edit:
-- Breisacher.ES       19-Apr-85 16:26:02
-- Bradshaw.ES         28-Jan-85 10:13:09
-- Created by editing SampleBWSApplicationMessagesImpl

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  ApplicationFolder USING [FindDescriptionFile, FromName],
  BWSFileTypes USING [systemFileCatalog],
  Catalog USING [Open],
  Heap USING [systemZone],
  NSFile USING [Close, Error, Find, GetReference, Handle, nullHandle, nullReference, OpenByReference, Reference],
  NSString USING [FreeString, String],
  OptionFile USING [GetStringValue, GetWorkstationProfile],
  SampleBWSApplicationOps,
  XMessage USING [ClientData, FreeMsgDomainsStorage, Handle, MessagesFromReference, MsgDomains],
  XString USING [FromSTRING, NSStringFromReader, Reader, ReaderBody];

SampleMsgFileImpl: PROGRAM
  IMPORTS ApplicationFolder, Catalog, Heap, NSFile, NSString, OptionFile, XMessage, XString
  EXPORTS SampleBWSApplicationOps = {

  -- Data

  h: XMessage.Handle ← NIL;

  localZone: UNCOUNTED ZONE ← Heap.systemZone;

  -- Procedures

  DeleteMessages: PROCEDURE [clientData: XMessage.ClientData] = {};

  GetMessageHandle: PUBLIC PROCEDURE RETURNS [XMessage.Handle] = {RETURN[h]};

  InitMessages: PROCEDURE = {
    internalName: XString.ReaderBody ← XString.FromSTRING ["SampleBWSApplication"L];
    msgDomains: XMessage.MsgDomains ← NIL;
    msgDomains ← XMessage.MessagesFromReference [
      file: GetMessageFileRef [ApplicationFolder.FromName [@internalName]],
      clientData: NIL,
      proc: DeleteMessages ];
    h ← msgDomains[0].handle;
    XMessage.FreeMsgDomainsStorage [msgDomains];
    };

  GetMessageFileRef: PROCEDURE [folder: NSFile.Reference]
    RETURNS [msgFile: NSFile.Reference ← NSFile.nullReference] = {
    folderHandle: NSFile.Handle ← NSFile.nullHandle;
    adf: NSFile.Reference ← NSFile.nullReference;
    internalName: XString.ReaderBody ← XString.FromSTRING ["SampleBWSApplication"L];
    messageFile: XString.ReaderBody ← XString.FromSTRING ["MessageFile"L];

    FindMessageFileFromName: PROCEDURE [value: XString.Reader] = {
      nssName: NSString.String ← XString.NSStringFromReader [r: value, z: localZone];
      msgFileHandle: NSFile.Handle ← NSFile.nullHandle;
      -- We do NSFile.Find here in case the name has an asterisk in it.
      msgFileHandle ← NSFile.Find [directory: folderHandle,
        scope: [filter: [matches[attribute: [name[nssName]]]]] !
          NSFile.Error => {msgFileHandle ← NSFile.nullHandle; CONTINUE}];
      IF msgFileHandle = NSFile.nullHandle THEN ERROR; -- no message file!
      msgFile ← NSFile.GetReference [msgFileHandle];
      NSFile.Close [msgFileHandle];
      NSString.FreeString [z: localZone, s: nssName];
      };

    IF folder = NSFile.nullReference THEN {
      -- No application folder, so use the system catalog and the WorkstationProfile
      folderHandle ← Catalog.Open [BWSFileTypes.systemFileCatalog];
      adf ← OptionFile.GetWorkstationProfile []}
    ELSE {
      -- There was an application folder, so use the folder and the adf inside it.
      folderHandle ← NSFile.OpenByReference [folder];
      adf ← ApplicationFolder.FindDescriptionFile [folderHandle]};
    OptionFile.GetStringValue [section: @internalName,
      entry: @messageFile,
      callBack: FindMessageFileFromName,
      file: adf];
    NSFile.Close [folderHandle];
    };

  -- Mainline code

  InitMessages[];

  }...
```

```
-- File: Converter.mesa
-- Last Revised by: Caro                          4-Jan-87 11:51:11
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Event
      USING [EventType],
  NSFile
      USING [Handle, nullHandle, Reference, Session,
      nullReference, Type],
  PropertySheet
      USING [MenuItemType],
  Window
      USING [Handle],
  XString
      USING [Reader, ReaderBody];

<<
--=============================================================================
-- OVERVIEW:

This is the public interface for the converter icon.  A common user interface and attributes model is supplied by the converter common
software for use by clients.  Clients should read the WPM chapter "Converter" for details of usage.  A brief summary follows.

Old Features (for backward compatibility)
- Support for single file input conversions
- Log file for recording conversion messages
- Preservation of user defined defaults in icon attributes
New Features
- Optional background processing
- New, simpler, user interface
- User defined suffix for output files
- Support for multi-file input conversions
- Client defined parameters for conversions

For single file input conversions, use Register[].
For multi-file input conversions, use RegisterMultiple[].

Note that the strings (srcFormat & dstFormat) passed to the register procedures and all other procedures taking these arguments, are
tested for strict equality, therefore the converter is CASE SENSITIVE.  The standard string for "VP Document" can be acquired through
the public interface ConverterMsg.

The NSFile.Type provided to the register procedures is the type that converter icon will use as a hint towards determining the correct
conversion to apply for any given input file.  However, the user may override the selection made by the converter, so it is up to the
client conversion to know enough about the source format it claims to understand to recognize invalid input.

The register procedures return status, which should be checked by clients.

Clients must use the PostMessage procedure to display messages to the user.  PostMessage guarantees that messages are handled
appropriately with respect to foreground/background mode, and that messages are stored in the log.

Access to the current user's Document Pagination entry (in the User Profile) is available.  The format of the entry is:

        [Conversion]
        Document Pagination: value

where "value" is one of {compress, simple, none}. This is meant to be used by clients that will be creating VP documents, and thus using
the DocInterchangeDefs.FinishCreation procedure. This procedure provides control of pagination through a
DocInterchangeDefs.PaginateOption parameter. The PaginateOption type in this module is meant to mean the same thing as the
DocInterchange type, but a different type was used to avoid a dependency of the Converter on the Documents world. Clients should use the
following for the FinishCreation parameter:

        (SELECT Converter.GetPOption[] FROM
           compress => compress,
           simple => simple,
           none => none,
         ENDCASE => ERROR)

--=============================================================================
>>

Converter: DEFINITIONS =
BEGIN

--===================
-- CONSTANTS
--===================

anyType: NSFile.Type = 4415; --/* firstStarType + 63, root */
<<
If the client conversion does not care what the input icon type is, or is specifically designed to accept any type, then use this
constant for "srcType" passed to Register[], or as the type in the corresponding element of a TypeList.
>>

nullID: RegistrationID = 0;

--===================
-- TYPES
--===================

CvData: TYPE[2];

ConvertProc: TYPE = PROC[
    source: NSFile.Handle,
    cvData: CvData,
```

```
    session: NSFile.Session,
    srcInstance: LONG POINTER ← NIL,
    dstInstance: LONG POINTER ← NIL,
    background: BOOLEAN ← FALSE]
  RETURNS [dest: NSFile.Handle ← NSFile.nullHandle];
<<
ConvertProc is used for either of the register procedures. Argument "source" is the source or input file, opened (and closed) by the
converter icon implementation.  Argument "cvData" is needed by the client to pass to support or utility routines, such as PostMessage[].
 Argument "session" is the session that all client NSFile operations should be run in, including the creation of "dest".  Arguments
"srcInstance" and "dstInstance" are the client created instance data generated by either or both DependentOptionProcs (intended to
contain current parameters for use by the client).  By convention, a NIL value means that the client should load parameters from
permanent storage, if any (analogous to conversion with DISPLAY OPTIONS off).  Argument "background" lets the client know whether it is
in a forked process or not.  Note that the converter icon implementation automatically sets the forked process to clientBackground
priority.
>>

DependentOptionProc: TYPE = PROC [
    options: BOOLEAN ← TRUE,  -- FALSE if property sheet
    cvData: CvData,
    which: FormatToUse,
    srcFormat: XString.Reader,
    destFormat: XString.Reader,
    window: Window.Handle,
    oldInstance: LONG POINTER ← NIL]  -- turns window into formwindow
  RETURNS [
    menuItemProc: MenuItemProc,
    destroy: DestroyProc,
    instance: LONG POINTER  -- guaranteed to be passed to ConvertProc
    ];

FormatToUse: TYPE = {source, destination};

DestroyProc: TYPE = PROC [instance: LONG POINTER];

<<
DependentOptionProc is a call-back proc passed to the converter icon by the client conversion via the DestinationOptions[] or
SourceOptions[] procedures.  This procedure is called when the converter option or property sheet needs to display client defined
parameters in a formwindow.  This procedure should turn the "window" passed into a formwindow.  The client's conversion is uniquely
identified by the pair of strings "srcFormat" and "destFormat".  The particular parameter set (formwindow) is determined by "which".
Return argument "menuItemProc" will be called by the converter icon when the user clicks on a command button, and is analogous to the
menuItemProc of the PropertySheet interface.  Parameter (formwindow) data should be returned in "instance", but "instance" must not be
attached to the formwindow, since the formwindow may be created and destroyed several times before menuItemProc AND ConvertProc are
called.  The converter will call the client provided "destroy" proc when instance is no longer needed.

By convention, "oldInstance" is passed with value NIL if the converter wants the client to allocate a new "instance".  If "oldInstance"
is non-NIL, then this means that the converter has destroyed the formwindow, but the user wants to see the previous settings again.
When "oldInstance" is non-NIL, the client should re-create the formWindow, but use the values of "oldInstance" as the defaults for the
data displayed in the formwindow.  Client may simply return "oldInstance" as "instance" if appropriate.  Note that it is the CLIENT's
responsibility to update "instance" if the formwindow is destroyed before a call to "menuItemProc".

IMPORTANT USAGE NOTE: Use the ancestors of "window" with CAUTION!  The parent of "window" is a formwindow, and a window item of ITS
parent, which is also a formwindow.  Initially, the parent window will be adjusted to an appropriate height so that "window" is fully
displayed.  The width of "window"'s parent is fixed (and can be obtained with FormWindow.NeededDims[].)  Make sure that the dims of
"window" fit within its parent. You can change the height of the window with ResizeDetailWindow.
>>

IndependentOptionProc: TYPE = PROC [
    input: LONG UNSPECIFIED]
  RETURNS [
    output: LONG UNSPECIFIED];
<<
Place-holder for environment independent conversion parameters.
>>

MenuItemProc: TYPE = PROC [
    instance: LONG POINTER, -- returned from DependentOptionProc
    menuItem: PropertySheet.MenuItemType]
  RETURNS [
    ok: BOOLEAN ← TRUE];
<<
If ok is returned FALSE, menu button processing is suspended and the parent sheet is left open.
>>

PaginateOption: TYPE = {compress, simple, none};
<<
NOTE: This type is meant to parallel DocInterchangeDefs.PaginateOption,
  and should probably change if it changes. However, the implementing code
  and the associated documentation should also change.
>>

RegistrationID: TYPE = CARDINAL;

Status: TYPE = {registered, overridden, alreadyExisted, busy, error};

TypeList: TYPE = LONG POINTER TO TypeListObj;
TypeListObj: TYPE = RECORD [
    ordered: BOOLEAN ← FALSE,
    1: SEQUENCE n: CARDINAL OF TypeTuple];
TypeTuple: TYPE = RECORD [
    srcType: NSFile.Type ← anyType,
    minExpected: CARDINAL ← 1,
    maxExpected: CARDINAL ← CARDINAL.LAST
    ];
<<
TypeList is passed to RegisterMultiple as a complex hint as to what the converter icon should expect for "correct" input to a multi-file
conversion.  "minExpected" and "maxExpected" can be used for forming ranges.  The client can allocate this data out of temporary memory,
as this data is copied by the converter icon.  If "maxExpected" is less than "minExpected", then any number of files are to be expected
```

```
(essentially infinite), by convention.
>>
nullTypeTuple: TypeTuple = [anyType, 1, CARDINAL.LAST];


--====================
-- SIGNALS
--====================

Failed: SIGNAL [failure: FailureModes];
FailureModes: TYPE = {
    fileError,
    invalidID,
    busy,
    unknown
    };
<<
Inform the client that something has gone wrong.
>>

--====================
-- PROCEDURES
--====================

--/* Register procedures: for use by client conversion initialization code */

Register: PROC [
    srcType: NSFile.Type,
    srcFormat: XString.Reader,
    destFormat: XString.Reader,
    convertProc: ConvertProc,
    sizeChange: CARDINAL ← 100,
    override: BOOLEAN ← TRUE,
    forkable: BOOLEAN ← FALSE]
  RETURNS [
    old: ConvertProc,
    status: Status ← registered,
    id: RegistrationID ← nullID];

--/* srcFormat and destFormat are copied */
<<
BACKWARD COMPATIBILITY NOTICE: This is compatible with old style conversion since additional parameters are automatically defaulted, BUT
the strings are passed here as Readers rather than ReaderBody's.  String comparison is case sensitive.

Register single file input conversion.  Argument "srcType" is a hint about the icon type that would match this conversion.  Arguments
"srcFormat" and "destFormat" are displayed as well as used as identifiers, so the client should take care in defining them
(multinational issues, message files, etc.)  Argument "sizeChange" is a hint as to how much file space is needed to do the conversion.
The value 100 means that the output will be approximately the same size as the input (i.e. 100%).  Fifty would mean that the output is
half the size, 200 means twice the size, etc.  Argument "override" indicates whether an existing (already registered) conversion with
the same identification (srcType, srcFormat & destFormat) should be overwritten.  "status" will be returned with "overridden".  The
argument "forkable" lets the converter icon know if the client is forkable.  NON RE-ENTRANT CLIENTS SHOULD SET THIS VALUE TO FALSE!
Return argument "old" is the existing conversion of the same idenfication, if any.  "status" is returned with "registered" if everything
is ok, or "alreadyExisted" if one already exists of that idenfication and override was FALSE.  Return argument "id" may be saved by the
client for use in identifying the conversion (but in most cases is not needed), but should NOT be written to a permanent storage (file)
as these ids are generated dynamically relative to a boot session.

If converter status is busy, registration FAILED!  Client should AddDependency to the appropriate EventType to retry the registration.
See GetEventType[];
>>

RegisterMultiple: PROC [
    typeList: TypeList ← NIL,  -- NIL means any type acceptable
    srcFormat: XString.Reader,
    destFormat: XString.Reader,
    multiConvertProc: ConvertProc,
    sizeChange: CARDINAL ← 100,
    override: BOOLEAN ← TRUE,
    forkable: BOOLEAN ← FALSE]
  RETURNS [
    old: ConvertProc,
    status: Status ← registered,
    id: RegistrationID ← nullID];

--/* typeList, srcFormat, and destFormat are all copied */
<<
Arguments are the same as Register[], except: "typeList" is a more complicated hint as to what the converter should expect in the input
folder.  "multiConvertProc" is analogous to the "convertProc" of Register[].  String comparison is case sensitive.  If status is busy,
registration FAILED!  Client should AddDependency to the appropriate EventType to retry the registration.  See GetEventType[].
>>

--/* Conversion parameter (options) registration procedures */

DestinationOptions: PROC [
    srcFormat: XString.Reader,
    destFormat: XString.Reader, -- one option implementation per src/dest pair
    dependentOptions: DependentOptionProc ← NIL,
    independentOptions: IndependentOptionProc ← NIL,
    override: BOOLEAN ← FALSE]
  RETURNS [
    oldDep: DependentOptionProc ← NIL,
    oldInd: IndependentOptionProc ← NIL,
    status: Status ← registered];
<<
Client calls this procedure to register destination options.  Can only be called AFTER a call to one of the register procedures.
"status" is error if invalid or other problem.  If "status" is busy, client must AddDependency to the appropriate EventType and retry
the procedure call.  See GetEventType[].
>>
```

```
SourceOptions: PROC [
    srcFormat: XString.Reader,
    destFormat: XString.Reader, -- one option implementation per src/dest pair
    dependentOptions: DependentOptionProc ← NIL,
    independentOptions: IndependentOptionProc ← NIL,
    override: BOOLEAN ← FALSE]
  RETURNS [
    oldDep: DependentOptionProc ← NIL,
    oldInd: IndependentOptionProc ← NIL,
    status: Status ← registered];
<<
Client calls this procedure to register source options.  Can only be called AFTER a call to one of the register procedures.  "status" is
error if invalid or other problem.  If "status" is busy, client must AddDependency to the appropriate EventType and retry the procedure
call.  See GetEventType[];
>>

--/* Conversion parameter (options) access procedures */

CreateClientFile: PROC [cvData: CvData, srcFormat, destFormat: XString.Reader, prefix: XString.Reader] RETURNS [ref: NSFile.Reference]--!
NSFile.Error --;
<<
Client calls this procedure to create a private data file that will be associated with the converter icon.  This is useful for storing
large amounts of parameter data, like character translation tables.  Deleting the parent directory of the file returned is a fatal
error.  Client may delete created file at any time, however.

File names are guaranteed to be unique (the converter code concatenates identification information to the end of "prefix").  This implies
that "prefix" is not the entire name of the file.  "prefix" MUST NOT contain either of the wildcard characters (* or #).  The client
should not change the name of the file.  Note that file names that begin with the character "=" (XCharSet0.Codes0[equals]) or "$"
(XCharSet0.Codes[currency]) are reserved by the Foreign Conversion Team.  NSFile errors are allowed to propagate.  Files are created of
type tText (StarFileTypes.text).

It is up to the client to properly version stamp client files, and upgrade as needed.
>>


FindClientFile: PROC [cvData: CvData, srcFormat, destFormat: XString.Reader, prefix: XString.Reader] RETURNS [ref: NSFile.Reference ←
NSFile.nullReference]--! NSFile.Error--;
<<
Client calls this procedure to access any of a set of special private files, associated with the particular converter icon, that the
client may use to store filed data.  See CreateClientFile.  Deleting the parent directory of the file returned is a fatal error.  Client
may delete created file at any time, however.

If no such file is found, "ref" is returned with NSFile.nullReference.  File names are guaranteed to be unique (the converter code
concatenates identification information to the end of "prefix").  "prefix" MUST NOT contain either of the wildcard characters (* or #).
  Note that file names that begin with the character "=" (XCharSet0.Codes0[equals]) or "$" (XCharSet0.Codes[currency]) are reserved by
the Foreign Conversion Team.  NSFile errors are allowed to propagate.
>>


--/* Utility procedures */

GetSingle: PROC [
    id: RegistrationID]
  RETURNS [
    srcType: NSFile.Type,
    srcRb: XString.ReaderBody,
    dstRb: XString.ReaderBody,
    convertProc: ConvertProc,
    sizeChange: CARDINAL ← 100,
    forkable: BOOLEAN ← FALSE];  --! Failed[invalidID] --

GetMultiple: PROC [
    id: RegistrationID]
  RETURNS [
    typeList: TypeList ← NIL,
    srcRb: XString.ReaderBody,
    dstRb: XString.ReaderBody,
    multiConvertProc: ConvertProc,
    sizeChange: CARDINAL ← 100,
    forkable: BOOLEAN ← FALSE];  --! Failed[invalidID] --
<<
Client should not alter "typeList", since it is a pointer to the actual data kept by the converter icon.  Treat this data as read only.
Same goes for the XStrings.
>>

--/* Support procedures for client conversions */

GetEventType: PROC RETURNS [et: Event.EventType];
<<
If the register procedures return busy status, AddDependency to the EventType returned by this procedure.  If, for some reason, the
client wishes to preserve IMPORT independence from this interface, the string for the EventType can be found in the External
Implementation Specification (WPM Chapter "Converter").  The string for the EventType that is notified if the conversion common software
is not loaded is also defined in the External Implementation Specification (WPM Chapter "Converter"), but is NOT returned by this
procedure -- do not use the results of this procedure for the "not loaded" dependency!
>>


PostMessage: PROC [msg: XString.Reader, cvData: CvData, cr: BOOLEAN ← TRUE, clear: BOOLEAN ← TRUE];
<<
To properly handle posting of messages to the log file, or in the background, use this procedure.  Direct posts via Attention will not be
logged, or will conflict with foreground messages.  If "cr" is true, a carriage return (XFormat.CR) is posted to the log BEFORE "msg",
else no carriage return is posted.  If "clear" is true, then the previous attention message is cleared, else it is not.
>>


ResizeDetailWindow: PROC [cvData: CvData, window: Window.Handle, which: FormatToUse, newHeight: INTEGER ← 0] RETURNS [oldHeight:
```

```
INTEGER];
<<
"window" is the client's formwindow.  "which" indicates if the client's formwindow is for the source options or destination options.
"newHeight" is the new height for the formwindow.  "oldHeight" is the old height of the formwindow.  If "newHeight" is defaulted,
nothing is changed, but the old height is returned.
>>


GetZone: PROCEDURE RETURNS [z: UNCOUNTED ZONE];
<<
Access to converter icon's permanent zone.  Care should be taken when using this resource.
>>


GetPOption: PROCEDURE RETURNS [p: PaginateOption];
<<
Access to user defined pagination option.
>>

END....

 8-Nov-84 22:37:33 - MSchneider.pa - CREATE
10-Dec-84 17:15:18 - MSchneider - Change name, add Status, PostMessage. CvData
19-Dec-84 15:03:59 - MSchneider - make PostMessage take a ReaderBody, not a Reader, because XMessage.Get returns a ReaderBody now (BWS
4.0)
 4-Feb-85 12:37:42 - MSchneider - Make client specified source and destinations into ReaderBodies
12-Apr-85 12:16:36 - MSchneider - merged in ConversionZone, added lots of comments
 8-May-85  9:19:02 - MSchneider - Added PaginateOption type and pOption variable, plus associated comments
13-Aug-85 17:01:49 - MSchneider - Added sizeChange parameter to Register, plus associated comments.
16-Jul-86 12:06:42 - Caro - Major changes to support new features
15-Sep-86 13:17:50 - Caro - Took id out of DependentOptionProc
17-Dec-86 14:17:33 - Caro - Added session to ConvertProc
 4-Jan-87 11:54:43 - Caro - Added ok to MenuItemProc
```

```
-- File: CvAscii.mesa
-- Last Revised by: Erickson                    24-Nov-16:53:30
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation

DIRECTORY
  Converter
    USING [ConvertProc, CvData, DestroyProc, DependentOptionProc, MenuItemProc],
  NSFile
    USING [Reference],
  Window
    USING [Handle],
  XString
    USING [Reader, ReaderBody, Writer];

<<
-------------------------------------------------------------------------------
-- OVERVIEW:

Private definitions interface for the ascii conversion.
-------------------------------------------------------------------------------
>>

CvAscii: DEFINITIONS =
BEGIN

--------------------
-- CONSTANTS
--------------------

bit7or8: CARDINAL = 0;
pcAscii: CARDINAL = 1;

proportional: CARDINAL = 0;
fixed: CARDINAL = 1;

bit7: CARDINAL = 0;
bit8: CARDINAL = 1;
dstPCAscii: CARDINAL = 2;

unlimited: CARDINAL = 0;
limited: CARDINAL = 1;

dfltAsciiEncoding: CARDINAL = bit7or8;
dfltFont: CARDINAL = proportional;
dfltTrailing: BOOLEAN = FALSE;
dfltChars: CARDINAL = 80;
dfltWordWrap: BOOLEAN = TRUE;
dfltTables: BOOLEAN = FALSE;
dfltFrames: BOOLEAN = FALSE;
dfltEncoding: CARDINAL = bit7;
dfltLineLen: CARDINAL = unlimited;

leadingMargin: CARDINAL = 2;
pointsBetweenItems: CARDINAL = 10;

--------------------
-- TYPES
--------------------

Boolean: TYPE = MACHINE DEPENDENT RECORD[
  zeros(0:0..14): [0..7777B], value(0:15..15): BOOLEAN];

Common: TYPE = LONG POINTER TO CommonData;
CommonData: TYPE = RECORD [
  cvData: Converter.CvData,
  options: BOOLEAN,
  window: Window.Handle,
  owner: Owners,
  ref: NSFile.Reference,
  f: CommonObj,
  textRb: FiledXStrings,
  text: EncodedText,
  z: UNCOUNTED ZONE];
<<
The same data strcture is used by all the client formwindows/details sections.
>>

Filed: TYPE = LONG POINTER TO CommonObj;
CommonObj: TYPE = MACHINE DEPENDENT RECORD [
  atovAsciiEncoding: CARDINAL ← dfltAsciiEncoding,
  font: CARDINAL ← dfltFont,
  ignoreTrailing: Boolean ← [0, dfltTrailing],
  vtoaAsciiEncoding: CARDINAL ← dfltEncoding,
  lineLen: CARDINAL ← dfltLineLen,
  charsSuffix: CARDINAL ← dfltChars,
  wordWrap: Boolean ← [0, dfltWordWrap],
  convertTables: Boolean ← [0, dfltTables],
  simulateFrames: Boolean ← [0, dfltFrames],
  spare0: CARDINAL ← 0,
  spare1: CARDINAL ← 0,
  spare2: CARDINAL ← 0,
  spare3: CARDINAL ← 0];
<<
This data structure is the filed data object, along with the various strings/text items that come from the formwindows.
>>
```

```
EncodedText: TYPE = ARRAY TextIDs OF LONG STRING;
<<
Use long strings internally, since they are better suited to ASCII text.
>>

FiledXStrings: TYPE = ARRAY TextIDs OF XString.ReaderBody;
<<
Filed strings are kept here.
>>

Owners: TYPE = {AtoVsrc, AtoVdst, VtoAdst, backstop};

TextIDs: TYPE = {
  paraEndsWith,
  atovReplaceUnknown,
  endLine,
  endPara,
  vtoaReplaceUnknown,
  replaceOffice,
  spare0,
  spare1,
  spare2
  };


--====================
-- SIGNALS
--====================

Problem: SIGNAL [err: ProblemType];

ProblemType: TYPE = {obsoleteDataFile, fatalError, doDflts, other};

--====================
-- PROCEDURES
--====================

AsciiToVP: Converter.ConvertProc;
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];

Exported by CvAsciiToVPImpl.
>>


AsciiToVPSrcOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvAsciiToVPImpl.
>>


AsciiToVPDstOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvAsciiToVPImpl.
>>


CommonMenu: Converter.MenuItemProc;
<< = PROCEDURE [instance: LONG POINTER, menuItem: PropertySheet.MenuItemType] RETURNS [ok: BOOLEAN ← TRUE];

Exported by CvAsciiFWImpl.
>>


CreateCommon: PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: Owners] RETURNS [my: Common];  --!
NSFile.Error
<<
Exported by CvAsciiDataImpl.
>>


CreateFW: PROC [my: Common, window: Window.Handle, owner: Owners];
<<
Exported by CvAsciiFWImpl.
>>


DataFromWindow: PROC [w: Window.Handle] RETURNS [my: Common];
<<
Exported by CvAsciiMainImpl
>>


DataToWindow: PROC [my: Common, w: Window.Handle];
<<
Exported by CvAsciiMainImpl
>>
```

```
DestroyCommon: Converter.DestroyProc;
<< = PROCEDURE [instance: LONG POINTER];

Exported by CvAsciiDataImpl.
>>

GetPreMargin: PROC [item: MessageKey] RETURNS [leads: CARDINAL];
<<
Exported by CvAsciiMainImpl.
>>


GetMessage: PROC [msg: MessageKey] RETURNS [msgRb: XString.ReaderBody];
<<
Exported by CvAsciiMsgFileImpl.
>>


InitFiledData: PROC [my: Common];  -- ! NSFile.Error
<<
Create and initialize client file.  Exported by CvAsciiDataImpl.
>>


LoadFiledData: PROC [my: Common];  -- ! NSFile.Error, Problem
<<
Read filed data.  Exported by CvAsciiDataImpl.
>>


ParseItem: PROC [my: Common, r: XString.Reader, item: MessageKey, buf: XString.Writer ← NIL] RETURNS [ok: BOOLEAN, ls: LONG STRING];
<<
Exported by CvAsciiParseImpl.   If ok is FALSE, error during parse.  ls is NIL if item has null text.  buf is a temporary buffer that
will be created and destroyed each time the proc is called if defaulted, otherwise it will just be used.
>>


StoreFiledData: PROC [my: Common];   -- ! NSFile.Error
<<
Write filed data.  Exported by CvAsciiDataImpl.
>>


VPToAscii: Converter.ConvertProc;
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];

Exported by CvAsciiFromVPImpl.
>>


VPToAsciiDstOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvAsciiFromVPImpl.
>>

--=====================
-- MESSAGES
--=====================

MessageKey: TYPE = {
    asciiSrcDoc,
    asciiDstDoc,
    paraEndsWith,
    asciiEncoding,
    asciiEncodingChoices,
    font,
    fontSuffix,
    fontChoices,
    ignoreTrailing,
    ascii3wayChoices,
    lineLen,
    lineLenChoices,
    charsSuffix,
    wordWrap,
    endLine,
    endPara,
    replaceUnknown,
    replaceOffice,
    convertTables,
    simulateFrames,
    spare0,
    spare1,
    spare2,
    spare3,
    lastPsheetItem,
    left,
    right,
    cr,
    lf,
    nl,
    ff,
    tab,
    createError,
```

```
    notPF,
    paginating,
    skippedTableData,
    dfltMeta,
    dfltChar,
    prefix,
    doneFailed,
    backstop,
    metaError,
    charsOutOfBounds,
    fatalError,
    extraErr0,
    extraErr1,
    aToVDfltMeta};


END.

LOG
 5-Dec-84 15:01:26 - MSchneider.pa - CREATED
19-Dec-84 15:31:39 - MSchneider - update to BWS 4.0
16-Apr-85 10:40:52 - MSchneider - added some comments and owner statement
28-May-85  9:23:59 - MSchneider - took out messages now in common interface
26-Feb-87 16:17:12 - Caro - Added paginating and spares
18-Mar-87 14:02:39 - Caro - Completely rewritten for Enhancements I
24-Nov-87 16:51:13 - Erickson - added aToVDfltMeta to change A to V paraEndsWith default
                         from <CR><LF> to <CR>
```

```
-- File: CvAsciiDataImpl.mesa
-- Last Revised by: Erickson                    24-Nov-87 16:54:21
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Courier
    USING [Description, DeserializeParameters, Error, Free,
    Parameters, SerializeParameters],
  Converter
    USING [CreateClientFile, CvData, DestroyProc, FindClientFile],
  ConverterMsg
    USING [Get, kvpDocument],
  CvAscii
    USING [Common, CommonData, CommonObj,
    GetMessage, Owners, Problem, TextIDs],
  Environment
    USING [bytesPerPage],
  Heap
    USING [Create, Delete],
  NSFile
    USING [Delete, Error, Handle, nullReference, OpenByReference],
  NSFileStream
    USING [Create, GetLength, Handle, SetLength],
  Stream
    USING [Delete, InvalidOperation],
  Window
    USING [Handle],
  XString
    USING [CopyToNewReaderBody, DescribeReaderBody, nullReaderBody, ReaderBody];

<<
--------------------------------------------------------------------------------
-- OVERVIEW:

Data and filed data procedures
--------------------------------------------------------------------------------
>>

CvAsciiDataImpl: PROGRAM
IMPORTS
  Converter, ConverterMsg, Courier, CvAscii, Heap,
  NSFile, NSFileStream, Stream, XString
EXPORTS
  CvAscii =
BEGIN

--------------------
-- CONSTANTS
--------------------

keyBits: Key = 2707974433;   --/* never change this value! */

currentVersion: Version = 1;  --/* change this value if you alter the filed data format */
--------------------------------------------------------------------------------
-- History of Versions (update each time version number changes)
-- 18-Mar-87 11:48:29 -  1 - First version
--------------------------------------------------------------------------------

--------------------
-- TYPES
--------------------

Key: TYPE = LONG CARDINAL;

Version: TYPE = INTEGER;

--------------------
-- PUBLIC PROCEDURES
--------------------

CreateCommon: PUBLIC PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: CvAscii.Owners] RETURNS [my:
CvAscii.Common] = {
    z: UNCOUNTED ZONE ← Heap.Create[initial: 16, increment: 28];

    my ← z.NEW[CvAscii.CommonData ← [
      cvData: cvData,
      options: options,
      window: window,
      owner: owner,
      ref: NSFile.nullReference,
      f: [],
      textRb: ALL[XString.nullReaderBody],
      text: ALL[NIL],
      z: z]];

    --/* find client file */
    BEGIN
      ENABLE UNWIND => Heap.Delete[z];
    prefix: XString.ReaderBody ← CvAscii.GetMessage[prefix];
    src: XString.ReaderBody ← CvAscii.GetMessage[asciiSrcDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.FindClientFile[
      cvData: cvData,
      srcFormat: @src,
```

```
      destFormat: @dst,
      prefix: @prefix];

    IF my.ref = NSFile.nullReference THEN
        {
        --/* file never created, so initialize */
        InitFiledData[my];  --/* fills in my.ref */
        };

    --/* read data */
    BEGIN
      ENABLE CvAscii.Problem =>
        {
        file: NSFile.Handle ← NSFile.OpenByReference[my.ref];
        aToVMeta: XString.ReaderBody ← CvAscii.GetMessage[aToVDfltMeta];
        meta: XString.ReaderBody ← CvAscii.GetMessage[dfltMeta];
        char: XString.ReaderBody ← CvAscii.GetMessage[dfltChar];
        --/* get rid of old file, reinitialize */
        NSFile.Delete[file];
        InitFiledData[my];
        my.textRb ← [
          paraEndsWith: aToVMeta,
          atovReplaceUnknown: char,
          endLine: meta,
          endPara: meta,
          vtoaReplaceUnknown: char,
          replaceOffice: char,
          spare0: char,
          spare1: char,
          spare2: char];
        CONTINUE;
        };

    LoadFiledData[my];
    END;
    END;
};


DestroyCommon: PUBLIC Converter.DestroyProc = {
<< = PROCEDURE [instance: LONG POINTER];
>>
    my: CvAscii.Common ← instance;
    z: UNCOUNTED ZONE;

    IF my = NIL THEN RETURN;
    z ← my.z;
    Heap.Delete[z];
};


InitFiledData: PUBLIC PROC [my: CvAscii.Common] = {
    myObj: CvAscii.CommonData;
    aToVMeta: XString.ReaderBody ← CvAscii.GetMessage[aToVDfltMeta];
    meta: XString.ReaderBody ← CvAscii.GetMessage[dfltMeta];
    char: XString.ReaderBody ← CvAscii.GetMessage[dfltChar];

    --/* make dummy filed data */
    myObj.f ← [];
    myObj.textRb ← [
      paraEndsWith: aToVMeta,
      atovReplaceUnknown: char,
      endLine: meta,
      endPara: meta,
      vtoaReplaceUnknown: char,
      replaceOffice: char,
      spare0: char,
      spare1: char,
      spare2: char];

    --/* create client file */
    BEGIN
    prefix: XString.ReaderBody ← CvAscii.GetMessage[prefix];
    src: XString.ReaderBody ← CvAscii.GetMessage[asciiSrcDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.CreateClientFile[
      cvData: my.cvData,
      srcFormat: @src,
      destFormat: @dst,
      prefix: @prefix];
    END;

    myObj.ref ← my.ref;
    myObj.z ← my.z;
    myObj.owner ← backstop;  --/* let StoreFiledData know we are initializing */
    --/* store */
    StoreFiledData[@myObj];
};


LoadFiledData: PUBLIC PROC [my: CvAscii.Common] = {
    sh: NSFileStream.Handle ← [NIL];
    file: NSFile.Handle;
    parms: Courier.Parameters;
    tz: UNCOUNTED ZONE ← NIL;
```

```
    --/* read filed data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND =>
            {
            Stream.Delete[sh];
            IF tz # NIL THEN Heap.Delete[tz];
            };
        };

    --/* open data file */
    file ← NSFile.OpenByReference[my.ref];

    --/* open read stream on data file */
    sh ← NSFileStream.Create[file: file, closeOnDelete: TRUE];

    --/* create temporary zone for disjoint data */
    tz ← Heap.Create[(NSFileStream.GetLength[sh]/Environment.bytesPerPage) + 2];

    --/* read key */
    BEGIN
    key: Key;

    parms ← [location: @key, description: DescribeKey];
    Courier.DeserializeParameters[parms, sh, tz];
    IF key # keyBits THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        SIGNAL CvAscii.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read version */
    BEGIN
    ver: Version;

    parms ← [location: @ver, description: DescribeVersion];
    Courier.DeserializeParameters[parms, sh, tz];
    IF ver # currentVersion THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        SIGNAL CvAscii.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read commonObj */
    parms ← [location: @my.f, description: DescribeCommonObj];
    Courier.DeserializeParameters[parms, sh, tz];

    --/* read paraEndsWith */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[paraEndsWith] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read atovReplaceUnknown */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[atovReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endLine */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endLine] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endPara */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endPara] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read vtoaReplaceUnknown */
    BEGIN
```

```
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[vtoaReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read replaceOffice */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[replaceOffice] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* skip spares */
<<
    THROUGH [0..3) DO
        rb: XString.ReaderBody;
        parms ← [location: @rb, description: XString.DescribeReaderBody];
        Courier.DeserializeParameters[parms, sh, tz];
        Courier.Free[parms, tz];
    ENDLOOP;
>>

    END;

    --/* clean up */
    Stream.Delete[sh];
    Heap.Delete[tz];
};


-- ============================================================================
-- StoreFiledData
-- * This is tricky, since common data is used.  This routine could be called
-- * three different times, with different subsets of data, but the whole
-- * file must be written each time.
-- ============================================================================

StoreFiledData: PUBLIC PROC [my: CvAscii.Common] = {
    dataFile: NSFile.Handle;
    sh: NSFileStream.Handle;
    parms: Courier.Parameters;
    tmpMy: CvAscii.CommonData;

    --/* fill out dummy */
    tmpMy ← my↑;
    IF my.owner # backstop THEN
        LoadFiledData[@tmpMy];

    --/* open data file */
    dataFile ← NSFile.OpenByReference[my.ref];

    --/* open stream on file */
    sh ← NSFileStream.Create[file: dataFile, closeOnDelete: TRUE];
    NSFileStream.SetLength[fileStream: sh, lengthInBytes: 0];

    --/* write data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND => Stream.Delete[sh];
        };

    --/* write key */
    BEGIN
    key: Key ← keyBits;

    parms ← [location: @key, description: DescribeKey];
    Courier.SerializeParameters[parms, sh];
    END;

    --/* write version */
    BEGIN
    ver: Version ← currentVersion;

    parms ← [location: @ver, description: DescribeVersion];
    Courier.SerializeParameters[parms, sh];
    END;

    --/* update portions of data record */
    SELECT my.owner FROM
        AtoVsrc =>
            {
            tmpMy.textRb[paraEndsWith] ← my.textRb[paraEndsWith];
            tmpMy.f.atovAsciiEncoding ← my.f.atovAsciiEncoding;
            };
        AtoVdst =>
            {
            tmpMy.f.font ← my.f.font;
            tmpMy.textRb[atovReplaceUnknown] ← my.textRb[atovReplaceUnknown];
```

```
            tmpMy.f.ignoreTrailing ← my.f.ignoreTrailing;
            };
        VtoAdst =>
            {
            tmpMy.f.vtoaAsciiEncoding ← my.f.vtoaAsciiEncoding;
            tmpMy.f.lineLen ← my.f.lineLen;
            tmpMy.f.charsSuffix ← my.f.charsSuffix;
            tmpMy.f.wordWrap ← my.f.wordWrap;
            tmpMy.textRb[endLine] ← my.textRb[endLine];
            tmpMy.textRb[endPara] ← my.textRb[endPara];
            tmpMy.textRb[vtoaReplaceUnknown] ← my.textRb[vtoaReplaceUnknown];
            tmpMy.textRb[replaceOffice] ← my.textRb[replaceOffice];
            tmpMy.f.convertTables ← my.f.convertTables;
            tmpMy.f.simulateFrames ← my.f.simulateFrames;
            };
        ENDCASE;

    --/* write filed data record */
    parms ← [location: @tmpMy.f, description: DescribeCommonObj];
    Courier.SerializeParameters[parms, sh];

    --/* write paraEndsWith string */
    parms ← [location: @tmpMy.textRb[paraEndsWith], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write atovReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[atovReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endLine string */
    parms ← [location: @tmpMy.textRb[endLine], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endPara string */
    parms ← [location: @tmpMy.textRb[endPara], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write vtoaReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[vtoaReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write replaceOffice string */
    parms ← [location: @tmpMy.textRb[replaceOffice], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write spare0 string */
    parms ← [location: @tmpMy.textRb[spare0], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write spare1 string */
    parms ← [location: @tmpMy.textRb[spare1], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write spare2 string */
    parms ← [location: @tmpMy.textRb[spare2], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    END;

    Stream.Delete[sh];
};

--------------------
-- PROCEDURES
--------------------

DescribeKey: Courier.Description = {
    p: LONG POINTER TO Key = notes.noteSize[SIZE[Key]];
    notes.noteLongCardinal[p];
};

DescribeVersion: Courier.Description = {
    p: LONG POINTER TO Version = notes.noteSize[SIZE[Version]];
};

DescribeCommonObj: Courier.Description = {
    p: LONG POINTER TO CvAscii.CommonObj = notes.noteSize[
      SIZE[CvAscii.CommonObj]];
};

END...

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:55:56 - Erickson - Changed default setting of paraEndsWith
                        to <CR> instead of <CR><LF>
```

```
-- File: CvAsciiFromVPImpl.mesa
-- Last Revised by: Caro                          16-Sep-87 12:21:45
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, SP, TAB],
  BackgroundProcess
    USING [UserAbort],
  Converter
    USING [ConvertProc, CvData, DependentOptionProc, DestroyProc, MenuItemProc, PostMessage],
  ConverterMsg,
  CvAscii
    USING [bit7, bit8, Common, CommonMenu, CreateCommon, CreateFW,
    DestroyCommon, GetMessage, limited, MessageKey, Owners,
    ParseItem, Problem, unlimited],
  DocInterchangeDefs
    USING [Close, Doc, Enumerate, EnumProcsRecord, Error, NewParagraphProc, Open, OpenStatus, PFCProc, PageBreakProc, TextProc],
  NSFile
    USING [Attribute, Create, Delete, Error, GetReference, Handle, nullHandle, Session],
  NSFileStream
    USING [Create, Handle],
  StarFileTypes
    USING [text],
  Stream
    USING [PutChar, Delete],
  String
    USING [MakeString],
  TIP
    USING [UserAbort],
  XChar
    USING [Code, Set],
  XCharSet0
    USING [Codes0],
  XCharSet41
    USING [Codes41],
  XCharSet42
    USING [Codes42],
  XCharSet46
    USING [Codes46],
  XCharSet357
    USING [Codes357],
  XCharSets
    USING [Sets],
  XMessage
    USING [MsgKey],
  XString
    USING [InvalidEncoding, ReaderBody, Map, MapCharProc];

<<
--============================================================================
-- OVERVIEW:

VP to ASCII conversion.
--============================================================================
>>

CvAsciiFromVPImpl: PROGRAM
IMPORTS
  BackgroundProcess, Converter, ConverterMsg, CvAscii,
  DocInterchangeDefs,
  NSFile, NSFileStream, Stream, String, TIP, XChar, XString
EXPORTS
  CvAscii =
BEGIN

--====================
-- CONSTANTS
--====================

tabInterval: CARDINAL = 8;

lineHtInPoints: CARDINAL = 12;

--====================
-- TYPES
--====================

VAData: TYPE = LONG POINTER TO VADataObj;
VADataObj: TYPE = RECORD [
    source: NSFile.Handle,
    output: NSFileStream.Handle,  --/* created from dest */
    cvData: Converter.CvData,
    session: NSFile.Session,
    dst: CvAscii.Common,
    background: BOOLEAN,
    doc: DocInterchangeDefs.Doc,
    putc: PutCProc,
    firstPara: BOOLEAN,
    encoding: CARDINAL,          --/* value of user's char translation choice */
    line: LONG STRING,           --/* line buffer */
    n: CARDINAL,                 --/* index of next char in line buffer */
    pos: CARDINAL,               --/* current position on virtual line */
    max: CARDINAL,               --/* last column in line */
    lastWhite: CARDINAL,         --/* last white */
```

```
    wordWrap: BOOLEAN,
    after: CARDINAL,            --/* number of eop strings to output */
                                --/*   for previous paragraph */
    z: UNCOUNTED ZONE];

PutCProc: TYPE = PROC [va: VAData, c: CHARACTER];

--------------------
-- PUBLIC PROCEDURES
--------------------

VPToAscii: PUBLIC Converter.ConvertProc = {
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];
>>
  ENABLE CvAscii.Problem, NSFile.Error, XString.InvalidEncoding =>
      {
      msgRb: XString.ReaderBody ← CvAscii.GetMessage[fatalError];

      Post[msgRb, cvData];
      CONTINUE;
      };

    IF source = NSFile.nullHandle THEN RETURN;
    dest ← VtoA[source, cvData, session, srcInstance, dstInstance, background];
};


<<
-------------------------------------------------------------------------------
This DependentOptionProc creates instance data with CreateCommon.  The data is distinguished by the owner variable.  The CommonObj within
CvAscii.CommonData is the data structure written to the client file stored as the icon properties.  Only those fields pertaining to the
owner are used.
-------------------------------------------------------------------------------
>>


VPToAsciiDstOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvAscii.Owners ← VtoAdst;

    menuItemProc ← CvAscii.CommonMenu;
    destroy ← CvAscii.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvAscii.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvAscii.Problem => {owner ← backstop; instance ←
NIL; CONTINUE}]
    ELSE
        {
        my: CvAscii.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };


    --/* make formwindow */
    CvAscii.CreateFW[instance, window, owner];
};

--------------------
-- PROCEDURES
--------------------

VtoA: Converter.ConvertProc = {
    aborted: BOOLEAN ← FALSE;
    dataSkipped: BOOLEAN ← FALSE;
    attr: ARRAY [0..1) OF NSFile.Attribute ← [[type[StarFileTypes.text]]];
    enumProcs: DocInterchangeDefs.EnumProcsRecord ← [
      newParagraphProc: EndPrevAsciiPara,
      pageBreakProc: AddAsciiPage,
      textProc: AddAsciiText,
      pfcProc: AddAsciiPFC];
    openStatus: DocInterchangeDefs.OpenStatus;
    vaData: VADataObj;  --/* only works if Enumerate doesn't FORK */
    dst: CvAscii.Common ← NIL;

    --/* initialize instance data */
    IF dstInstance = NIL THEN
        {
        ENABLE NSFile.Error, CvAscii.Problem =>
            {
            msgRb: XString.ReaderBody ← CvAscii.GetMessage[extraErr0];  --" Unrecoverable ASCII conversion error: damaged converter icon.
"
            Converter.PostMessage[
              msg: @msgRb,
              cvData: cvData,
              cr: FALSE,
              clear: FALSE];
            GOTO terminate;
            };
        key: CvAscii.MessageKey ← CvAscii.MessageKey.FIRST;  --/* dummy */

        --/* we only care about dst */
        dst ← CvAscii.CreateCommon[cvData, FALSE, NIL, VtoAdst];
```

```
        dst.text[endLine] ← CvAscii.ParseItem[
          my: dst,
          r: @dst.textRb[endLine],
          item: key].ls;

        dst.text[endPara] ← CvAscii.ParseItem[
          my: dst,
          r: @dst.textRb[endPara],
          item: key].ls;

        dst.text[vtoaReplaceUnknown] ← CvAscii.ParseItem[
          my: dst,
          r: @dst.textRb[vtoaReplaceUnknown],
          item: key].ls;

        dst.text[replaceOffice] ← CvAscii.ParseItem[
          my: dst,
          r: @dst.textRb[replaceOffice],
          item: key].ls;
        EXITS terminate => RETURN;
        }
    ELSE
        {
        dst ← dstInstance;
        };

    vaData ← [
         source: source,
         output: [NIL],
         cvData: cvData,
         session: session,
         dst: dst,
         background: background,
         doc: TRASH,
         putc: IF dst.f.lineLen = CvAscii.unlimited THEN UnbufferedPutC ELSE BufferedPutC,
         firstPara: TRUE,
         encoding: dst.f.vtoaAsciiEncoding,
         line: NIL,
         n: 0,
         pos: 0,
         max: 0,
         lastWhite: CARDINAL.LAST,
         wordWrap: dst.f.wordWrap.value,
         after: 0,
         z: dst.z];

    IF dst.f.lineLen = CvAscii.limited THEN
        {
        --/* ASSERT: charsSuffix IN [10..256] */
        --/* create line buffer */
        vaData.line ← String.MakeString[z: vaData.z, maxlength: dst.f.charsSuffix];
        vaData.line.length ← vaData.line.maxlength;
        vaData.max ← dst.f.charsSuffix - 1;
        IF dst.text[endLine] # NIL AND dst.text[endLine].length < vaData.max THEN
            {
            --/* max column is limit less visible end-of-line characters */
            FOR i: CARDINAL IN [0..dst.text[endLine].length) DO
                SELECT dst.text[endLine][i] FROM
                    Ascii.CR, Ascii.LF, Ascii.FF => NULL;
                    ENDCASE => vaData.max ← vaData.max - 1;
            ENDLOOP;
            };
        };

    BEGIN
      ENABLE
        {
        NSFile.Error => GOTO nsErr;
        DocInterchangeDefs.Error => GOTO docErr;
        UNWIND => IF dstInstance = NIL THEN
            {
            CvAscii.DestroyCommon[dst];
            dst ← NIL;
            };
        };

    [vaData.doc, openStatus] ← DocInterchangeDefs.Open[
      docFileRef: NSFile.GetReference[source, vaData.session],
      session: vaData.session];
    IF openStatus # ok THEN GOTO docErr;

    dest ← NSFile.Create[
      directory: NSFile.nullHandle,
      attributes: DESCRIPTOR[attr],
      session: session ! NSFile.Error => {
          IF error = [space[mediumFull]] THEN
              Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], vaData.cvData]
          ELSE
              Post[CvAscii.GetMessage[createError], vaData.cvData];
          GOTO nsErr}];

    vaData.output ← NSFileStream.Create[
      file: dest,
      closeOnDelete: FALSE,
      session: vaData.session];

    dataSkipped ← DocInterchangeDefs.Enumerate[
```

```
      textContainer: [doc[vaData.doc]],
      procs: @enumProcs,
      clientData: @vaData ! ABORTED => {
        dataSkipped ← TRUE;
        aborted ← TRUE;
        Post[ConverterMsg.Get[ConverterMsg.kuserAbort], vaData.cvData];
        CONTINUE}];

    --/* AR 13705: flush any remaining text */
    --/* ASSERT: n = 0 IF dst.f.lineLen # CvAscii.limited */
    IF NOT aborted AND vaData.n > 0 THEN
        {
        RawPuts[@vaData, vaData.line, vaData.n];
        --/* AR 14393: terminate last paragraph */
        RawPuts[@vaData, vaData.dst.text[endPara]];
        };

    Stream.Delete[vaData.output ! NSFile.Error => {
      IF error = [space[mediumFull]] THEN
          Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], vaData.cvData]
      ELSE
          Post[ConverterMsg.Get[ConverterMsg.kunknownProblem], vaData.cvData];
      NSFile.Delete[dest, vaData.session];
      dest ← NSFile.nullHandle;
      GOTO nsErr}];
    IF dataSkipped THEN
          Post[ConverterMsg.Get[ConverterMsg.kdataSkipped], vaData.cvData];

    DocInterchangeDefs.Close[@vaData.doc];

    EXITS
      nsErr => {
        IF vaData.doc # NIL THEN
            DocInterchangeDefs.Close[@vaData.doc ! DocInterchangeDefs.Error => CONTINUE]};
      docErr => {
        key: XMessage.MsgKey ←
          SELECT openStatus FROM
            malFormed, incompatible => ConverterMsg.kincompatible,
            outOfDiskSpace, outOfVM => ConverterMsg.koutOfSpace,
            ENDCASE => ConverterMsg.kcantOpen;

        Post[ConverterMsg.Get[key], vaData.cvData];
        IF vaData.doc # NIL THEN
            DocInterchangeDefs.Close[@vaData.doc ! DocInterchangeDefs.Error => CONTINUE];
        dest ← NSFile.nullHandle};
    END;
    IF vaData.line # NIL THEN vaData.z.FREE[@vaData.line];
    --/* destroy instance data if created by this proc */
    IF dstInstance = NIL AND dst # NIL THEN CvAscii.DestroyCommon[dst];
};


Post: PROC [msgRb: XString.ReaderBody, cvData: Converter.CvData] = {
    Converter.PostMessage[
      msg: @msgRb,
      cvData: cvData,
      cr: TRUE,
      clear: FALSE];
};


CheckAbort: PROC [background: BOOLEAN] RETURNS [yes: BOOLEAN] = INLINE {
    yes ← (background AND BackgroundProcess.UserAbort[]) OR
      (NOT background AND TIP.UserAbort[NIL]);
};


--/* Enumeration Procs */

AddAsciiPage: DocInterchangeDefs.PageBreakProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData ← clientData;
    -- form feed appended for a new page
    va.putc[va, Ascii.FF];
};


EndPrevAsciiPara:  DocInterchangeDefs.NewParagraphProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps, paraProps:
DocInterchangePropsDefs.ReadonlyParaProps] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData ← clientData;

    --/* ASSERT: n = 0 IF dst.f.lineLen # CvAscii.limited */
    IF va.n > 0 THEN RawPuts[va, va.line, va.n];  --/* flush any pending text */

    --/* a new para char means we terminate the previous ASCII paragraph */
    IF va.firstPara AND paraProps.basicProps.preLeading < lineHtInPoints THEN
        va.firstPara ← FALSE
    ELSE
        {
        --/* ceiling to next highest line */
        newlines: CARDINAL ←
          (paraProps.basicProps.preLeading + lineHtInPoints - 1)  /
          lineHtInPoints;
```

```
        IF NOT va.firstPara THEN
            {
            --/* end previous paragraph */
            RawPuts[va, va.dst.text[endPara]];

            --/* append endLine strings for AFTER paragraph spacing */
            THROUGH [1..va.after] DO
                RawPuts[va, va.dst.text[endLine]];
            ENDLOOP;
            };

        --/* this newPara character contains properties for the FOLLOWING *
        --/*    paragraph, therefore output BEFORE line spacing first */
        THROUGH [1..newlines] DO
            RawPuts[va, va.dst.text[endLine]];
        ENDLOOP;
        va.firstPara ← FALSE;
        };
    va.n ← 0;  --/* reset line index */
    va.pos ← 0;  --/* reset line position */
    va.lastWhite ← CARDINAL.LAST;  --/* reset last white */
    --/* save AFTER line spacing */
    va.after ←
      (paraProps.basicProps.postLeading + lineHtInPoints - 1) / lineHtInPoints;
};


AddAsciiPFC: DocInterchangeDefs.PFCProc = {};

<<
-------------------------------------------------------------------------------
AddAsciiText
This procedure does the bulk of the text handling.  Its main purpose is to translate VP characters into ASCII characters, according to
the user's encoding selection.
-------------------------------------------------------------------------------
>>

AddAsciiText: DocInterchangeDefs.TextProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps, text: XString.Reader, textEndContext:
XString.Context] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData = clientData;
    --/* local procs */
    ISO7: XString.MapCharProc = {
        escape: CHARACTER ← 377C;
        chset: XCharSets.Sets ← LOOPHOLE[XChar.Set[c]];
        putc: PutCProc = va.putc;

        SELECT chset FROM
            latin =>
                {
                code0: XCharSet0.Codes0 ← LOOPHOLE[XChar.Code[c]];

                SELECT code0 FROM
                    IN [space..tilde] =>
                        putc[va, LOOPHOLE[code0, CHAR]];
                    tab, LOOPHOLE[211B] =>
                        putc[va, Ascii.TAB];
                    lineFeed, newLine =>
                        {
                        Puts[va, va.dst.text[endLine]];
                        IF va.n > 0 THEN FlushLine[va];
                        };
                    dollar =>
                        putc[va, '$];
                    leftDoubleQuote,
                    rightDoubleQuote,
                    neutralDoubleQuote =>
                        putc[va, '"];
                    leftSingleQuote,
                    rightSingleQuote =>
                        putc[va, ''];
                    IN [upperAEdigraph..lowerEng] =>
                        SELECT code0 FROM
                            lowerAEdigraph =>
                                Puts[va, "ae"L];
                            upperAEdigraph =>
                                Puts[va, "AE"L];
                            lowerOEdiagraph =>
                                Puts[va, "oe"L];
                            upperOEdiagraph =>
                                Puts[va, "OE"L];
                            lowerIJdiagraph =>
                                Puts[va, "ij"L];
                            upperIJdiagraph =>
                                Puts[va, "IJ"L];
                            lowerOslash =>
                                putc[va, 'o];
                            upperOslash =>
                                putc[va, 'O];
                            lowerKgreenlandic =>
                                putc[va, 'K];
                            lowerIdotless =>
                                putc[va, 'i];
                            ENDCASE =>
                                Puts[va, va.dst.text[vtoaReplaceUnknown]];
                    ENDCASE =>
```

```
                    Puts[va, va.dst.text[vtoaReplaceUnknown]];
                };
            jisSymbol1 =>
                {
                code41: XCharSet41.Codes41 ← LOOPHOLE[XChar.Code[c]];
                SELECT code41 FROM
                    hyphen => va.putc[va, '-];
                    ENDCASE => Puts[va, va.dst.text[replaceOffice]];
                };
            generalSymbols1 =>
                {
                code357: XCharSet357.Codes357 ← LOOPHOLE[XChar.Code[c]];
                SELECT code357 FROM
                    nonBreakingHyphen => va.putc[va, '-];
                    nonBreakingSpace => va.putc[va, Ascii.SP];
                    ENDCASE => Puts[va, va.dst.text[replaceOffice]];
                };
             ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
          stop ← FALSE;
    };


    ISO8: XString.MapCharProc = {
        chset: XCharSets.Sets ← LOOPHOLE[XChar.Set[c]];

        SELECT chset FROM
            latin =>
                {
                code0: XCharSet0.Codes0 ← LOOPHOLE[XChar.Code[c]];

                SELECT code0 FROM
                    IN [space..tilde] =>
                        va.putc[va, LOOPHOLE[code0, CHAR]];
                    tab, LOOPHOLE[211B] =>
                        va.putc[va, Ascii.TAB];
                    lineFeed, newLine =>
                        {
                        Puts[va, va.dst.text[endLine]];
                        IF va.n > 0 THEN FlushLine[va];
                        };
                    dollar =>
                        va.putc[va, '$];
                    ENDCASE =>
                        va.putc[va, LOOPHOLE[code0, CHAR]];
                };
            jisSymbol1 =>
                {
                code41: XCharSet41.Codes41 ← LOOPHOLE[XChar.Code[c]];
                SELECT code41 FROM
                    hyphen => va.putc[va, '-];
                    ENDCASE => Puts[va, va.dst.text[replaceOffice]];
                };
            generalSymbols1 =>
                {
                code357: XCharSet357.Codes357 ← LOOPHOLE[XChar.Code[c]];
                SELECT code357 FROM
                    nonBreakingHyphen => va.putc[va, '-];
                    nonBreakingSpace => va.putc[va, Ascii.SP];
                    ENDCASE => Puts[va, va.dst.text[replaceOffice]];
                };
            ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
        stop ← FALSE;
    };


    PCASCII: XString.MapCharProc = {
        chset: XCharSets.Sets = LOOPHOLE[XChar.Set[c]];
        putc: PutCProc = va.putc;

        SELECT chset FROM
            latin =>
                {
                code0: XCharSet0.Codes0 ← LOOPHOLE[XChar.Code[c]];

                SELECT code0 FROM
                    IN [space..tilde] =>
                        putc[va, LOOPHOLE[code0, CHAR]];
                    tab, LOOPHOLE[211B] =>
                        putc[va, Ascii.TAB];
                    lineFeed, newLine =>
                        {
                        Puts[va, va.dst.text[endLine]];
                        IF va.n > 0 THEN FlushLine[va];
                        };
                    dollar =>
                        putc[va, '$];
                    leftDoubleQuote,
                    rightDoubleQuote,
                    neutralDoubleQuote =>
                        putc[va, '"];
                    leftSingleQuote,
                    rightSingleQuote =>
                        putc[va, ''];
                    invertedExclamation => putc[va, 255C];
                    cent => putc[va, 233C];
                    poundSterling => putc[va, 234C];
                    yen => putc[va, 235C];
```

```
            section => putc[va, 25C];
            leftDoubleGuillemet => putc[va, 256C];
            rightDoubleGuillemet => putc[va, 257C];
            leftArrow => putc[va, 33C];
            upArrow => putc[va, 30C];
            rightArrow => putc[va, 32C];
            downArrow => putc[va, 31C];
            degree => putc[va, 370C];
            paragraph => putc[va, 24C];
            upperAEdigraph => putc[va, 222C];
            lowerAEdigraph => putc[va, 221C];
            feminineSpanishOrdinal => putc[va, 246C];
            masculineSpanishOrdinal => putc[va, 247C];
            invertedQuestionMark => putc[va, 250C];
            oneHalf => putc[va, 253C];
            oneQuarter => putc[va, 254C];
            lowerSzed => putc[va, 341C];
            plusOrMinus => putc[va, 361C];
            ohmSign => putc[va, 352C];
            divide => putc[va, 366C];
            centeredDot => putc[va, 371C];
            overDotAccent => putc[va, 372C];
            superscript2 => putc[va, 375C];
            trademark, registered, copyright =>
                Puts[va, va.dst.text[replaceOffice]];
            ENDCASE =>
                Puts[va, va.dst.text[vtoaReplaceUnknown]];
        };
    jisSymbol1 =>
        {
        code41: XCharSet41.Codes41 ← LOOPHOLE[XChar.Code[c]];
        SELECT code41 FROM
            hyphen => putc[va, '-];
            lessThanOrEqualTo => putc[va, 363C];
            greaterThanOrEqualTo => putc[va, 362C];
            infinity => putc[va, 354C];
            male => putc[va, 13C];
            female => putc[va, 14C];
            ENDCASE => Puts[va, va.dst.text[replaceOffice]];
        };
    jisSymbol2 =>
        {
        code42: XCharSet42.Codes42 ← LOOPHOLE[XChar.Code[c]];
        SELECT code42 FROM
            blackUpTriangle => putc[va, 36C];
            blackDownTriangle => putc[va, 37C];
            VAL[176B] => putc[va, 11C];  -- large circle
            whiteSquare => Puts[va, va.dst.text[replaceOffice]];
            ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
        };
    greek =>
        {
        code46: XCharSet46.Codes46 ← LOOPHOLE[XChar.Code[c]];
        SELECT code46 FROM
            upperGamma => putc[va, 342C];
            upperTheta => putc[va, 351C];
            upperSigma => putc[va, 344C];
            upperPhi => putc[va, 350C];
            upperOmega => putc[va, 352C];
            lowerAlpha => putc[va, 340C];
            lowerDelta => putc[va, 353C];
            lowerEpsilon => putc[va, 356C];
            lowerMu => putc[va, 346C];
            lowerPi => putc[va, 343C];
            lowerSigma => putc[va, 345C];
            lowerTau => putc[va, 347C];
            lowerPhi => putc[va, 355C];
            ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
        };
    VAL[50B] =>
        {
        code50: CARDINAL ← LOOPHOLE[XChar.Code[c]];

        SELECT code50 FROM
            43B => putc[va, 332C];
            44B => putc[va, 277C];
            45B => putc[va, 331C];
            46B => putc[va, 300C];
            47B => putc[va, 303C];
            50B => putc[va, 302C];
            51B => putc[va, 264C];
            52B => putc[va, 301C];
            IN [120B..131B] => putc[va, LOOPHOLE[code50+145B, CHAR]];
            IN [132B..154B] => putc[va, LOOPHOLE[code50+164B, CHAR]];
            ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
        };
    generalSymbols2 =>
        {
        code356: CARDINAL ← LOOPHOLE[XChar.Code[c]];
        SELECT code356 FROM
            52B => putc[va, 376C];
            72B => putc[va, 177C];
            101B => putc[va, 10C];
            125B => putc[va, 12C];
            140B => putc[va, 260C];
            141B => putc[va, 262C];
            152B => putc[va, 251C];
```

```
                    172B => putc[va, 23C];
                    265B => putc[va, 22C];
                    266B => putc[va, 27C];
                    IN [271B..275B] => putc[va, LOOPHOLE[code356+42B, CHAR]];
                    276B => putc[va, 21C];
                    277B => putc[va, 20C];
                    314B => putc[va, 3C];
                    315B => putc[va, 4C];
                    317B => putc[va, 17C];
                    325B => putc[va, 16C];
                    335B => putc[va, 34C];
                    336B => putc[va, 26C];
                    337B => putc[va, 2C];
                    355B => putc[va, 364C];
                    356B => putc[va, 365C];
                    ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
                };
            generalSymbols1 =>
                {
                code357: XCharSet357.Codes357 ← LOOPHOLE[XChar.Code[c]];
                SELECT code357 FROM
                    doubleArrow => putc[va, 35C];
                    intersection => putc[va, 357C];
                    centeredBullet => putc[va, 7C];
                    not => putc[va, 252C];
                    equivalent => putc[va, 360C];
                    equalByDefinition => putc[va, 360C];
                    approximatelyEqual2 => putc[va, 367C];
                    root => putc[va, 373C];
                    shade => putc[va, 261C];
                    florin => putc[va, 237C];
                    pesetas => putc[va, 236C];
                    spades => putc[va, 6C];
                    clubs => putc[va, 5C];
                    smileFace => putc[va, 1C];
                    thinVerticalLine => putc[va, 263C];
                    thinHorizontalLine => putc[va, 304C];
                    thinIntersectingLines => putc[va, 305C];
                    nonBreakingHyphen => putc[va, '-];
                    nonBreakingSpace => putc[va, Ascii.SP];
                    ENDCASE => Puts[va, va.dst.text[replaceOffice]];
                };
            VAL[361B] =>  --/* accented characters */
                {
                code361: CARDINAL ← LOOPHOLE[XChar.Code[c]];
                SELECT code361 FROM
                    47B => putc[va, 216C];
                    50B => putc[va, 217C];
                    55B => putc[va, 200C];
                    61B => putc[va, 220C];
                    114B => putc[va, 245C];
                    124B => putc[va, 231C];
                    145B => putc[va, 232C];
                    241B => putc[va, 205C];
                    242B => putc[va, 240C];
                    243B => putc[va, 203C];
                    247B => putc[va, 204C];
                    250B => putc[va, 206C];
                    255B => putc[va, 207C];
                    260B => putc[va, 212C];
                    261B => putc[va, 202C];
                    262B => putc[va, 210C];
                    265B => putc[va, 211C];
                    276B => putc[va, 215C];
                    277B => putc[va, 241C];
                    300B => putc[va, 214C];
                    304B => putc[va, 213C];
                    314B => putc[va, 244C];
                    317B => putc[va, 225C];
                    320B => putc[va, 242C];
                    321B => putc[va, 223C];
                    324B => putc[va, 224C];
                    337B => putc[va, 227C];
                    340B => putc[va, 243C];
                    341B => putc[va, 226C];
                    345B => putc[va, 201C];
                    355B => putc[va, 230C];
                    ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
                };
            VAL[376B] =>
                IF LOOPHOLE[XChar.Code[c], CARDINAL] = 250B THEN
                    putc[va, 374C]
                ELSE
                    Puts[va, va.dst.text[vtoaReplaceUnknown]];
            ENDCASE => Puts[va, va.dst.text[vtoaReplaceUnknown]];
        stop ← FALSE;
    };

    --/* begin code */
    IF CheckAbort[va.background] THEN ERROR ABORTED;
    [] ← XString.Map[
      r: text,
      proc: SELECT va.encoding FROM
        CvAscii.bit7 => ISO7, CvAscii.bit8 => ISO8, ENDCASE => PCASCII];
};
```

```
--/* put procs */

UnbufferedPutC: PutCProc = {
    Stream.PutChar[va.output, c];
};


BufferedPutC: PutCProc = {
    line: LONG STRING ← va.line;
    output: NSFileStream.Handle ← va.output;

    IF va.pos > va.max THEN
        {
        IF va.wordWrap THEN
            {
            offset: CARDINAL;

            --/* determine offset to new text */
            IF va.lastWhite = CARDINAL.LAST THEN
                {
                IF va.n > 0 THEN
                    {
                    va.lastWhite ← va.n - 1;
                    offset ← va.n;
                    }
                ELSE
                    offset ← va.lastWhite ← 0;
                }
            ELSE
                offset ← va.lastWhite + 1;

            --/* flush to mark */
            FOR i: CARDINAL IN [0..va.lastWhite] DO
                Stream.PutChar[output, line[i]];
            ENDLOOP;

            --/* end line */
            RawPuts[va, va.dst.text[endLine]];

            --/* restore line */
            FOR i: CARDINAL IN [offset..va.n) DO
                line[i-offset] ← line[i];
            ENDLOOP;
            va.n ← va.n - offset;
            va.lastWhite ← CARDINAL.LAST;

            --/* reset pos */
            va.pos ← 0;
            FOR i: CARDINAL IN [0..va.n) DO
                va.pos ← IF line[i] = Ascii.TAB THEN
                    ((va.pos / tabInterval) + 1) * tabInterval
                ELSE IF (c = Ascii.CR OR c = Ascii.LF) THEN
                    va.pos
                ELSE
                    va.pos + 1;
            ENDLOOP;
            }
        ELSE
            {
            RawPuts[va, line, va.n];
            --/* end line */
            RawPuts[va, va.dst.text[endLine]];
            va.n ← 0;
            va.pos ← 0;
            };
        };

    IF va.n >= line.length THEN
        {
        RawPuts[va, line];
        va.n ← va.pos ← 0;
        va.lastWhite ← CARDINAL.LAST;
        };

    --/* append character */
    line[va.n] ← c;

    IF c = Ascii.SP THEN va.lastWhite ← va.n;
    va.pos ← IF c = Ascii.TAB THEN
        ((va.pos / tabInterval) + 1) * tabInterval
    ELSE IF (c = Ascii.CR OR c = Ascii.LF) THEN
        va.pos
    ELSE
        va.pos + 1;

    va.n ← va.n + 1;
};


--/* put a string */
Puts: PROC [va: VAData, s: LONG STRING] = {
    IF s = NIL THEN RETURN;
    IF s.length = 0 THEN RETURN;
```

```
    FOR i: CARDINAL IN [0..s.length) DO
        va.putc[va, s[i]];
    ENDLOOP;
};


--/* raw put string */
RawPuts: PROC [va: VAData, s: LONG STRING, limit: CARDINAL ← CARDINAL.LAST] = {
    IF s = NIL THEN RETURN;
    IF s.length = 0 THEN RETURN;

    IF limit = CARDINAL.LAST THEN limit ← s.length;
    FOR i: CARDINAL IN [0..limit) DO
        Stream.PutChar[va.output, s[i]];
    ENDLOOP;
};


FlushLine: PROC [va: VAData] = {
    RawPuts[va, va.line, va.n];
    va.n ← va.pos ← 0;
    va.lastWhite ← CARDINAL.LAST;
};


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:30:20 - Caro - Added error catcher in ConvertProc over CreateCommon,
                            ISO8 now has correct ENDCASE
10-Jul-87 11:31:10 - Caro - Added before/after line spacing
19-Aug-87 11:03:02 - Caro - Fixed AR 13535 by updating oldInstance window
                            Fixed AR 13705 by flushing remaining text
16-Sep-87 12:21:09 - Caro - Fixed AR 14393 by terminating with endPara
```

```
-- File: CvAsciiFWImpl.mesa
-- Last Revised by: Erickson                    17-Dec-87 16:03:15
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Attention
    USING [Post],
  Converter
    USING [MenuItemProc, ResizeDetailWindow],
  CvAscii,
  FormWindow
    USING [AppendItem, AppendLine, ChoiceChangeProc, ChoiceItems, Create,
    GetBooleanItemValue, GetChoiceItemValue,
    GetIntegerItemValue, GetTextItemValue,
    HasBeenChanged, HasAnyBeenChanged, LayoutProc, Line, MakeItemsProc,
    MakeBooleanItem, MakeChoiceItem, MakeIntegerItem, MakeTextItem,
    MinDimsChangeProc,
    SetBooleanItemValue, SetChoiceItemValue, SetIntegerItemValue,
    SetTabStops, SetTextItemValue,
    SetVisibility, TabStops, SetSelection, SetInputFocus],
  FormWindowMessageParse
    USING [FreeChoiceItems, ParseChoiceItemMessage],
  NSFile
    USING [Error],
  Window
    USING [Handle],
  XString
    USING [FreeReaderBytes, FreeWriterBytes, NewWriterBody, nullReaderBody,
    ReaderBody, WriterBody, InvalidNumber, Overflow];

<<
--=============================================================================
-- OVERVIEW:

Formwindow procedures
--=============================================================================
>>

CvAsciiFWImpl: PROGRAM
IMPORTS
  Attention, Converter, CvAscii,
  FormWindow, FormWindowMessageParse, NSFile, XString
EXPORTS
  CvAscii =
BEGIN

--====================
-- CONSTANTS
--====================

textWidth: CARDINAL = 320;
tabStopInterval: CARDINAL = CvAscii.pointsBetweenItems/2;

--====================
-- TYPES
--====================

--====================
-- PUBLIC PROCEDURES
--====================


CommonMenu: PUBLIC Converter.MenuItemProc = {
<< = PROCEDURE [instance: LONG POINTER, menuItem: PropertySheet.MenuItemType] RETURNS [ok: BOOLEAN ← TRUE];>>
    my: CvAscii.Common = instance;
    aToVMeta : XString.ReaderBody ← CvAscii.GetMessage[aToVDfltMeta];
    meta: XString.ReaderBody ← CvAscii.GetMessage[dfltMeta];
    char: XString.ReaderBody ← CvAscii.GetMessage[dfltChar];

    IF my = NIL THEN RETURN[ok: TRUE];

    SELECT menuItem FROM
        defaults =>
            {
            SELECT my.owner FROM
                AtoVsrc =>
                    {
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.paraEndsWith.ORD,
                      newValue: @aToVMeta,
                      repaint: FALSE];
                    FormWindow.SetChoiceItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.asciiEncoding.ORD,
                      newValue: CvAscii.dfltAsciiEncoding,
                      repaint: FALSE];
                    };
                AtoVdst =>
                    {
                    FormWindow.SetChoiceItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.font.ORD,
                      newValue: CvAscii.dfltFont,
                      repaint: FALSE];
```

```
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.replaceUnknown.ORD,
                      newValue: @char,
                      repaint: FALSE];
                    FormWindow.SetBooleanItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.ignoreTrailing.ORD,
                      newValue: CvAscii.dfltTrailing,
                      repaint: TRUE];
                    };
                VtoAdst =>
                    {
                    FormWindow.SetChoiceItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.ascii3wayChoices.ORD,
                      newValue: CvAscii.dfltEncoding,
                      repaint: FALSE];
                    FormWindow.SetChoiceItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.lineLen.ORD,
                      newValue: CvAscii.dfltLineLen,
                      repaint: FALSE];
                    FormWindow.SetIntegerItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.charsSuffix.ORD,
                      newValue: CvAscii.dfltChars,
                      repaint: FALSE];
                    FormWindow.SetBooleanItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.wordWrap.ORD,
                      newValue: CvAscii.dfltWordWrap,
                      repaint: FALSE];
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.endLine.ORD,
                      newValue: @meta,
                      repaint: FALSE];
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.endPara.ORD,
                      newValue: @meta,
                      repaint: FALSE];
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.replaceUnknown.ORD,
                      newValue: @char,
                      repaint: FALSE];
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.replaceOffice.ORD,
                      newValue: @char,
                      repaint: FALSE];
                    FormWindow.SetBooleanItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.convertTables.ORD,
                      newValue: CvAscii.dfltTables,
                      repaint: FALSE];
                    FormWindow.SetBooleanItemValue[
                      window: my.window,
                      item: CvAscii.MessageKey.simulateFrames.ORD,
                      newValue: CvAscii.dfltFrames,
                      repaint: TRUE];
                    };
                ENDCASE;
            };
            done =>
                {
                ENABLE NSFile.Error, CvAscii.Problem =>
                    {
                    msgRb: XString.ReaderBody ← CvAscii.GetMessage[doneFailed];
                    Attention.Post[@msgRb];
                    GOTO notOK;
                    };
                IF FormWindow.HasAnyBeenChanged[my.window] THEN
                    {
                    ok ← ApplyChanges[my];
                    IF NOT ok THEN GOTO notOK;
                    CvAscii.StoreFiledData[my];
                    };
                EXITS notOK => RETURN[ok: FALSE];
                };
            start =>
                {
                ok ← ApplyChanges[my];
                };
        ENDCASE;
};

CreateFW: PUBLIC PROC [my: CvAscii.Common, window: Window.Handle, owner: CvAscii.Owners] = {
    SELECT owner FROM
        AtoVsrc =>
            {
            FormWindow.Create[
              window: window,
              makeItemsProc: MakeAtoVSrc,
```

```
            layoutProc: LayoutAtoVSrc,
            minDimsChangeProc: GrowParent,
            clientData: my];
          CvAscii.DataToWindow[my, window];
          };
        AtoVdst =>
          {
          FormWindow.Create[
            window: window,
            makeItemsProc: MakeAtoVDst,
            layoutProc: LayoutAtoVDst,
            clientData: my];
          };
        VtoAdst =>
          {
          FormWindow.Create[
            window: window,
            makeItemsProc: MakeVtoADst,
            layoutProc: LayoutVtoADst,
            minDimsChangeProc: GrowParent,
            clientData: my];
          CvAscii.DataToWindow[my, window];
          };
        backstop =>
          {
          FormWindow.Create[
            window: window,
            makeItemsProc: MakeBackstop];
          };
        ENDCASE;
};

--====================
-- PROCEDURES
--====================

ApplyChanges: PROC [my: CvAscii.Common] RETURNS [ok: BOOLEAN ← TRUE] = {
    bufWb: XString.WriterBody ← XString.NewWriterBody[maxLength: 30, z: my.z];
    SELECT my.owner FROM
        AtoVsrc =>
          {
          IF FormWindow.HasBeenChanged[window: my.window, item: CvAscii.MessageKey.paraEndsWith.ORD] THEN
              {
              IF my.textRb[paraEndsWith] # XString.nullReaderBody THEN
                  XString.FreeReaderBytes[@my.textRb[paraEndsWith], my.z];
              my.textRb[paraEndsWith] ← FormWindow.GetTextItemValue[
                window: my.window,
                item: CvAscii.MessageKey.paraEndsWith.ORD,
                zone: my.z];
              };
          [ok: ok, ls: my.text[paraEndsWith]] ← CvAscii.ParseItem[
            my: my,
            r: @my.textRb[paraEndsWith],
            item: CvAscii.MessageKey.paraEndsWith,
            buf: @bufWb];
          IF NOT ok THEN RETURN;

          IF FormWindow.HasBeenChanged[window: my.window, item: CvAscii.MessageKey.asciiEncoding.ORD] THEN
              {
              my.f.atovAsciiEncoding ← FormWindow.GetChoiceItemValue[
                window: my.window,
                item: CvAscii.MessageKey.asciiEncoding.ORD];
              };
          };
       AtoVdst =>
          {
          IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.font.ORD] THEN
              {
              my.f.font ← FormWindow.GetChoiceItemValue[
                window: my.window,
                item: CvAscii.MessageKey.font.ORD];
              };
          IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.replaceUnknown.ORD] THEN
              {
              IF my.textRb[atovReplaceUnknown] # XString.nullReaderBody THEN
                  XString.FreeReaderBytes[@my.textRb[atovReplaceUnknown], my.z];
              my.textRb[atovReplaceUnknown] ← FormWindow.GetTextItemValue[
                window: my.window,
                item: CvAscii.MessageKey.replaceUnknown.ORD,
                zone: my.z];
              };
          [ok: ok, ls: my.text[atovReplaceUnknown]] ← CvAscii.ParseItem[
            my: my,
            r: @my.textRb[atovReplaceUnknown],
            item: CvAscii.MessageKey.replaceUnknown,
            buf: @bufWb];
          IF NOT ok THEN RETURN;

          IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.ignoreTrailing.ORD] THEN
              {
              my.f.ignoreTrailing.value ← FormWindow.GetBooleanItemValue[
                window: my.window,
                item: CvAscii.MessageKey.ignoreTrailing.ORD];
              };
          };
       VtoAdst =>
          {
```

```
    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.ascii3wayChoices.ORD] THEN
        {
        my.f.vtoaAsciiEncoding ← FormWindow.GetChoiceItemValue[
          window: my.window,
          item: CvAscii.MessageKey.ascii3wayChoices.ORD];
        };
    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.lineLen.ORD] THEN
        {
        my.f.lineLen ← FormWindow.GetChoiceItemValue[
          window: my.window,
          item: CvAscii.MessageKey.lineLen.ORD];
        };
    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.charsSuffix.ORD] THEN
        {
        my.f.charsSuffix ← CARDINAL[FormWindow.GetIntegerItemValue[window: my.window,
          item: CvAscii.MessageKey.charsSuffix.ORD !
            XString.InvalidNumber => {
              msgRb: XString.ReaderBody ← CvAscii.GetMessage[extraErr1];
              Attention.Post[@msgRb];
              GOTO Badnum;
              };
            XString.Overflow => {
              my.f.charsSuffix ← 0;
              CONTINUE;
              }]];
        IF my.f.charsSuffix NOT IN [10..256] THEN
            {
            msgRb: XString.ReaderBody ← CvAscii.GetMessage[charsOutOfBounds];
            Attention.Post[@msgRb];
            GOTO Badnum;
            };
        EXITS
          Badnum => {
            FormWindow.SetSelection[window: my.window,
                                    item: CvAscii.MessageKey.charsSuffix.ORD,
                                    firstChar: 0, lastChar: CARDINAL.LAST];
            FormWindow.SetInputFocus[window: my.window,
                                     item: CvAscii.MessageKey.charsSuffix.ORD,
                                     beforeChar: CARDINAL.LAST];
            RETURN[ok: FALSE];
          };
        };
    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.wordWrap.ORD] THEN
        {
        my.f.wordWrap.value ← FormWindow.GetBooleanItemValue[
          window: my.window,
          item: CvAscii.MessageKey.wordWrap.ORD];
        };
    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.endLine.ORD] THEN
        {
        IF my.textRb[endLine] # XString.nullReaderBody THEN
            XString.FreeReaderBytes[@my.textRb[endLine], my.z];
        my.textRb[endLine] ← FormWindow.GetTextItemValue[
          window: my.window,
          item: CvAscii.MessageKey.endLine.ORD,
          zone: my.z];
        };
    [ok: ok, ls: my.text[endLine]] ← CvAscii.ParseItem[
      my: my,
      r: @my.textRb[endLine],
      item: CvAscii.MessageKey.endLine,
      buf: @bufWb];
    IF NOT ok THEN RETURN;

    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.endPara.ORD] THEN
        {
        IF my.textRb[endPara] # XString.nullReaderBody THEN
            XString.FreeReaderBytes[@my.textRb[endPara], my.z];
        my.textRb[endPara] ← FormWindow.GetTextItemValue[
          window: my.window,
          item: CvAscii.MessageKey.endPara.ORD,
          zone: my.z];
        };
    [ok: ok, ls: my.text[endPara]] ← CvAscii.ParseItem[
      my: my,
      r: @my.textRb[endPara],
      item: CvAscii.MessageKey.endPara,
      buf: @bufWb];
    IF NOT ok THEN RETURN;

    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.replaceUnknown.ORD] THEN
        {
        IF my.textRb[vtoaReplaceUnknown] # XString.nullReaderBody THEN
            XString.FreeReaderBytes[@my.textRb[vtoaReplaceUnknown], my.z];
        my.textRb[vtoaReplaceUnknown] ← FormWindow.GetTextItemValue[
          window: my.window,
          item: CvAscii.MessageKey.replaceUnknown.ORD,
          zone: my.z];
        };
    [ok: ok, ls: my.text[vtoaReplaceUnknown]] ← CvAscii.ParseItem[
      my: my,
      r: @my.textRb[vtoaReplaceUnknown],
      item: CvAscii.MessageKey.replaceUnknown,
      buf: @bufWb];
    IF NOT ok THEN RETURN;

    IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.replaceOffice.ORD] THEN
```

```
                {
                IF my.textRb[replaceOffice] # XString.nullReaderBody THEN
                    XString.FreeReaderBytes[@my.textRb[replaceOffice], my.z];
                my.textRb[replaceOffice] ← FormWindow.GetTextItemValue[
                  window: my.window,
                  item: CvAscii.MessageKey.replaceOffice.ORD,
                  zone: my.z];
                };
            [ok: ok, ls: my.text[replaceOffice]] ← CvAscii.ParseItem[
              my: my,
              r: @my.textRb[replaceOffice],
              item: CvAscii.MessageKey.replaceOffice,
              buf: @bufWb];
            IF NOT ok THEN RETURN;

            IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.convertTables.ORD] THEN
                {
                my.f.convertTables.value ← FormWindow.GetBooleanItemValue[
                  window: my.window,
                  item: CvAscii.MessageKey.convertTables.ORD];
                };
            IF FormWindow.HasBeenChanged[my.window, CvAscii.MessageKey.simulateFrames.ORD] THEN
                {
                my.f.simulateFrames.value ← FormWindow.GetBooleanItemValue[
                  window: my.window,
                  item: CvAscii.MessageKey.simulateFrames.ORD];
                };
            };
        ENDCASE;
    XString.FreeWriterBytes[@bufWb];
};


GrowParent: FormWindow.MinDimsChangeProc = {
<< = PROCEDURE [window: Window.Handle, old: Window.Dims, new: Window.Dims];
>>
    my: CvAscii.Common = CvAscii.DataFromWindow[window];
    oldHeight: INTEGER;

    --/* don't adjust the first time window is viewed */
    IF my = NIL THEN RETURN;
    IF old = new THEN RETURN;

    --/* defaulting newHeight returns oldHeight without resizing */
    oldHeight ← Converter.ResizeDetailWindow[
      cvData: my.cvData,
      window: window,
      which: IF my.owner = AtoVsrc THEN source ELSE destination];

    --/* now resize window */
    [] ← Converter.ResizeDetailWindow[
      cvData: my.cvData,
      window: window,
      which: IF my.owner = AtoVsrc THEN source ELSE destination,
      newHeight: oldHeight + (new.h - old.h)];
};


MakeBackstop: FormWindow.MakeItemsProc = {
    tag: XString.ReaderBody ← CvAscii.GetMessage[backstop];

    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.backstop.ORD,
      boxed: FALSE,
      readOnly: TRUE,
      width: 400,
      initString: @tag];
};


MakeAtoVDst: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    my: CvAscii.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvAscii.GetMessage[font];
    tmp ← CvAscii.GetMessage[fontChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    sfx: XString.ReaderBody ← CvAscii.GetMessage[fontSuffix];

    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvAscii.MessageKey.font.ORD,
      tag: @tag,
      suffix: @sfx,
      values: values,
      initChoice: my.f.font,
      fullyDisplayed: TRUE];

    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;
```

```
    tag ← CvAscii.GetMessage[replaceUnknown];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.replaceUnknown.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[atovReplaceUnknown]];

    tag ← CvAscii.GetMessage[ignoreTrailing];
    FormWindow.MakeBooleanItem[
      window: window,
      myKey: CvAscii.MessageKey.ignoreTrailing.ORD,
      label: [string[tag]],
      initBoolean: my.f.ignoreTrailing.value];
};


MakeAtoVSrc: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    my: CvAscii.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvAscii.GetMessage[paraEndsWith];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.paraEndsWith.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[paraEndsWith]];

    tmp ← CvAscii.GetMessage[asciiEncodingChoices];
    tag ← CvAscii.GetMessage[asciiEncoding];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];

    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvAscii.MessageKey.asciiEncoding.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.atovAsciiEncoding,
      fullyDisplayed: TRUE];

    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;
};


MakeVtoADst: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];>>
    my: CvAscii.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvAscii.GetMessage[asciiEncoding];
    tmp ← CvAscii.GetMessage[ascii3wayChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];

    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvAscii.MessageKey.ascii3wayChoices.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.vtoaAsciiEncoding,
      fullyDisplayed: TRUE];

    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvAscii.GetMessage[lineLen];
    tmp ← CvAscii.GetMessage[lineLenChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];

    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvAscii.MessageKey.lineLen.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.lineLen,
      changeProc: LineLenXProc,
      fullyDisplayed: TRUE];

    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvAscii.GetMessage[charsSuffix];
    FormWindow.MakeIntegerItem[
      window: window,
      myKey: CvAscii.MessageKey.charsSuffix.ORD,
      suffix: @tag,
      visibility: IF my.f.lineLen = CvAscii.limited THEN visible ELSE invisible,
      signed: FALSE,
      width: 30,
```

```
      initInteger: INTEGER[my.f.charsSuffix]];

    tag ← CvAscii.GetMessage[wordWrap];
    FormWindow.MakeBooleanItem[
      window: window,
      myKey: CvAscii.MessageKey.wordWrap.ORD,
      visibility: IF my.f.lineLen = CvAscii.limited THEN visible ELSE invisible,
      label: [string[tag]],
      initBoolean: my.f.wordWrap.value];

    tag ← CvAscii.GetMessage[endLine];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.endLine.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[endLine]];

    tag ← CvAscii.GetMessage[endPara];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.endPara.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[endPara]];

    tag ← CvAscii.GetMessage[replaceUnknown];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.replaceUnknown.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[vtoaReplaceUnknown]];

    tag ← CvAscii.GetMessage[replaceOffice];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvAscii.MessageKey.replaceOffice.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[replaceOffice]];

    --+!+/* for future development */
    tag ← CvAscii.GetMessage[convertTables];
    FormWindow.MakeBooleanItem[
      window: window,
      myKey: CvAscii.MessageKey.convertTables.ORD,
      visibility: invisible,  --+!+/* disabled for VP 2.0 */
      label: [string[tag]],
      initBoolean: my.f.convertTables.value];

    tag ← CvAscii.GetMessage[simulateFrames];
    FormWindow.MakeBooleanItem[
      window: window,
      myKey: CvAscii.MessageKey.simulateFrames.ORD,
      visibility: invisible, --+!+/* disabled for VP 2.0 */
      label: [string[tag]],
      initBoolean: my.f.simulateFrames.value];
};


LayoutAtoVDst: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    leadingMargin: CARDINAL = CvAscii.leadingMargin;
    spaceAboveLine: CARDINAL = 5;
    line: FormWindow.Line;
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

    FormWindow.SetTabStops[window, tabChoice];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.font.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[font] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[font] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.replaceUnknown.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[replaceUnknown] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[replaceUnknown] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.ignoreTrailing.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[ignoreTrailing] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[ignoreTrailing] / tabStopInterval,
      repaint: FALSE];

};
```

```
LayoutAtoVSrc: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    leadingMargin: CARDINAL = CvAscii.leadingMargin;
    spaceAboveLine: CARDINAL = 5;
    line: FormWindow.Line;
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

    FormWindow.SetTabStops[window, tabChoice];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.paraEndsWith.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[paraEndsWith] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[paraEndsWith] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.asciiEncoding.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[asciiEncoding] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[asciiEncoding] / tabStopInterval,
      repaint: FALSE];

};


LayoutVtoADst: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];>>
    leadingMargin: CARDINAL = CvAscii.leadingMargin;
    spaceAboveLine: CARDINAL = 5;
    line: FormWindow.Line;
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

    FormWindow.SetTabStops[window, tabChoice];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.ascii3wayChoices.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[ascii3wayChoices] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[ascii3wayChoices] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.lineLen.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[lineLen] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[lineLen] / tabStopInterval,
      repaint: FALSE];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.charsSuffix.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[charsSuffix],
      tabStop: ,
      repaint: FALSE];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.wordWrap.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[wordWrap],
      tabStop: ,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.endLine.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[endLine] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[endLine] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.endPara.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[endPara] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[endPara] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.replaceUnknown.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[replaceUnknown] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[replaceUnknown] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
```

```
      window: window,
      item: CvAscii.MessageKey.replaceOffice.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[replaceOffice] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[replaceOffice] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.convertTables.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[convertTables] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[convertTables] / tabStopInterval,
      repaint: FALSE];
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvAscii.MessageKey.simulateFrames.ORD,
      line: line,
      preMargin: CvAscii.GetPreMargin[simulateFrames] MOD tabStopInterval,
      tabStop: CvAscii.GetPreMargin[simulateFrames] / tabStopInterval,
      repaint: FALSE];

};


--/* Change Procs */

LineLenXProc: FormWindow.ChoiceChangeProc = {
<< = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey, calledBecauseOf: FormWindow.ChangeReason, oldValue:
FormWindow.ChoiceIndex, newValue: FormWindow.ChoiceIndex];
>>
    IF newValue = oldValue THEN RETURN;
    IF newValue = CvAscii.limited THEN
        {
        FormWindow.SetVisibility[
          window: window,
          item: CvAscii.MessageKey.charsSuffix.ORD,
          visibility: visible,
          repaint: FALSE];
        FormWindow.SetVisibility[
          window: window,
          item: CvAscii.MessageKey.wordWrap.ORD,
          visibility: visible,
          repaint: TRUE];
        }
    ELSE
        {
        FormWindow.SetVisibility[
          window: window,
          item: CvAscii.MessageKey.charsSuffix.ORD,
          visibility: invisible,
          repaint: FALSE];
        FormWindow.SetVisibility[
          window: window,
          item: CvAscii.MessageKey.wordWrap.ORD,
          visibility: invisible,
          repaint: TRUE];
        };
};

END...

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:58:56 - Erickson - Changed paraEndsWith default to <CR> instead of <CR><LF>
17-Dec-87 15:48:52 - Erickson - AR 16414 - Added to ApplyChanges in the CvAscii.MessageKey.charsSuffix section.  The value read from the
prop sheet was expected to be a valid number.  If text was entered, the converter crashed the system.  I added signal checking for
InvalidNumber and Overflow.  If text is entered, the InvalidNumber signal is raised by FormWindow.GetIntegerItemValue, and is caught
here.  The user's input is then highlighted, that field of the propsheet is made the input focus, and a message is posted indicating the
problem.  This message was placed in the extraErr1 position in CvAsciiMsgFileImpl.mesa.  While I was here, I added a catch phrase for the
Overflow signal also, this simply sets the input value to zero and allows the already existing code to treat this as input out of range.
```

```
-- File: CvAsciiMainImpl.mesa
-- Last Revised by: Caro                        30-Jun-87 12:39:53
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom
    USING [MakeAtom],
  Attention
    USING [Post],
  BWSZone
    USING [Permanent],
  Context
    USING [Create, Error, Find, NopDestroyProc, Type, UniqueType],
  Converter
    USING [DestinationOptions, GetEventType, Register, Status, SourceOptions],
  ConverterMsg
    USING [Get, kvpDocument],
  ConverterPFOptions
    USING [conASCII],
  CvAscii
    USING [AsciiToVP, AsciiToVPDstOps, AsciiToVPSrcOps, Common,
    GetMessage, leadingMargin, MessageKey,
    pointsBetweenItems, ProblemType, VPToAscii, VPToAsciiDstOps],
  Event
    USING [AddDependency, AgentProcedure, EventType],
  Process
    USING [Detach, Pause, SecondsToTicks],
  ProductFactoring
    USING [Enabled],
  SimpleTextDisplay
    USING [MeasureString],
  StarFileTypes
    USING [document, text, unspecified],
  Window
    USING [Handle],
  XString
    USING [ReaderBody];
<<
--============================================================================
-- OVERVIEW:

Main code for ascii conversion.  Registations done here.
--============================================================================
>>

CvAsciiMainImpl: PROGRAM
IMPORTS
  Atom, Attention, BWSZone, Context, Converter, ConverterMsg,
  CvAscii, Event, Process, ProductFactoring, SimpleTextDisplay
EXPORTS
  CvAscii =
BEGIN

--====================
-- CONSTANTS
--====================

--====================
-- TYPES
--====================

Globals: TYPE = RECORD [
    leads: ItemLeads,
    ctype: Context.Type,
    z: UNCOUNTED ZONE];

ItemLeads: TYPE = ARRAY CvAscii.MessageKey[paraEndsWith..lastPsheetItem] OF CARDINAL;

--====================
-- GLOBALS
--====================

g: Globals;

--====================
-- PUBLIC SIGNALS
--====================

Problem: PUBLIC SIGNAL [err: CvAscii.ProblemType] = CODE;

--====================
-- PUBLIC PROCEDURES
--====================

DataFromWindow: PUBLIC PROC [w: Window.Handle] RETURNS [my: CvAscii.Common] = {
    my ← Context.Find[type: g.ctype, window: w ! Context.Error => {my ← NIL; CONTINUE}];
};


DataToWindow: PUBLIC PROC [my: CvAscii.Common, w: Window.Handle] = {
    Context.Create[
      type: g.ctype,
      data: my,
      proc: Context.NopDestroyProc,
      window: w ! Context.Error => CONTINUE];
```

```
};

GetPreMargin: PUBLIC PROC [item: CvAscii.MessageKey] RETURNS [leads: CARDINAL] = {
    RETURN[g.leads[item]];
};

--====================
-- PROCEDURES
--====================


Init: PROC = {
    z: UNCOUNTED ZONE = BWSZone.Permanent[];

    g ← [
      leads: ALL[CARDINAL.LAST],
      ctype: Context.UniqueType[],
      z: z];

    MeasureTags[];

    --/* register with converter icon */
    Register[];
};


MeasureTags: PROC = {
    lmarg: CARDINAL = CvAscii.leadingMargin;
    max: CARDINAL ← 0;
    --/* local proc */
    Length: PROC [key: CvAscii.MessageKey] RETURNS [width: CARDINAL] =
        {
        rb: XString.ReaderBody ← CvAscii.GetMessage[key];
        [width: width] ← SimpleTextDisplay.MeasureString[string: @rb];
        RETURN [width];
        };

    --/* begin code */
    g.leads ← [
      paraEndsWith: Length[paraEndsWith],
      asciiEncoding: Length[asciiEncoding],
      asciiEncodingChoices: 0,
      font: Length[font],
      fontSuffix: 0,
      fontChoices: 0,
      ignoreTrailing: 1,  -- no tag
      ascii3wayChoices: Length[asciiEncoding],
      lineLen: Length[lineLen],
      lineLenChoices: 0,
      charsSuffix: CARDINAL.LAST,
      wordWrap: CARDINAL.LAST,
      endLine: Length[endLine],
      endPara: Length[endPara],
      replaceUnknown: Length[replaceUnknown],
      replaceOffice: Length[replaceOffice],
      convertTables: 1,  -- no tag
      simulateFrames: 1,  -- no tag
      spare0: 0,
      spare1: 0,
      spare2: 0,
      spare3: 0,
      lastPsheetItem: 0];

    --/* now determine max */
    FOR i: CvAscii.MessageKey IN CvAscii.MessageKey[paraEndsWith..lastPsheetItem] DO
        IF g.leads[i] = CARDINAL.LAST THEN LOOP;
        max ← MAX[max, g.leads[i]];
    ENDLOOP;

    --/* now adjust */
    max ← max + lmarg;
    FOR i: CvAscii.MessageKey IN CvAscii.MessageKey[paraEndsWith..lastPsheetItem] DO
        SELECT g.leads[i] FROM
            0 => LOOP;
            1 => g.leads[i] ← max + 8;  -- compensate for no tag
            CARDINAL.LAST => g.leads[i] ← CvAscii.pointsBetweenItems;
            ENDCASE => g.leads[i] ← max - g.leads[i];
    ENDLOOP;
};


RegisterNow: PROC [first: BOOLEAN] RETURNS [allOk: BOOLEAN ← TRUE] = {
    doc: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];
    asciiDoc: XString.ReaderBody ← CvAscii.GetMessage[asciiSrcDoc];
    status: Converter.Status;
    --/* local proc */
    Check: PROC [status: Converter.Status] =
        {
        SELECT status FROM
            registered, alreadyExisted, overridden => NULL;
            busy =>
                {
                IF first THEN
                    {
                    et: Event.EventType ← Converter.GetEventType[];
```

```
                    --/* tell user registration will be done later */
                    --+$$$ not implemented

                    [] ← Event.AddDependency[
                       agent: RetryRegistration,
                       myData: NIL,
                       event: et];
                    first ← FALSE;  --/* only add once! */
                    };
                allOk ← FALSE;
                };
            error => allOk ← FALSE;  --+$$$ should post a message
            ENDCASE;
        };

    --/* begin code */
    status ← Converter.Register[
      srcType: StarFileTypes.text,
      srcFormat: @asciiDoc,
      destFormat: @doc,
      convertProc: CvAscii.AsciiToVP,
      sizeChange: 190,
      forkable: TRUE].status;

    Check[status];

    status ← Converter.Register[
      srcType: StarFileTypes.unspecified,
      srcFormat: @asciiDoc,
      destFormat: @doc,
      convertProc: CvAscii.AsciiToVP,
      sizeChange: 190,
      forkable: TRUE].status;

    Check[status];

    asciiDoc ← CvAscii.GetMessage[asciiDstDoc];
    status ← Converter.Register[
      srcType: StarFileTypes.document,
      srcFormat: @doc,
      destFormat: @asciiDoc,
      convertProc: CvAscii.VPToAscii,
      sizeChange: 63,
      forkable: TRUE].status;

    Check[status];

    --/* register ops */

    IF NOT allOk THEN RETURN;

    status ← Converter.DestinationOptions[
      srcFormat: @doc,
      destFormat: @asciiDoc,
      dependentOptions: CvAscii.VPToAsciiDstOps,
      override: TRUE].status;

    Check[status];

    asciiDoc ← CvAscii.GetMessage[asciiSrcDoc];
    status ← Converter.SourceOptions[
      srcFormat: @asciiDoc,
      destFormat: @doc,
      dependentOptions: CvAscii.AsciiToVPSrcOps,
      override: TRUE].status;

    Check[status];

    status ← Converter.DestinationOptions[
      srcFormat: @asciiDoc,
      destFormat: @doc,
      dependentOptions: CvAscii.AsciiToVPDstOps,
      override: TRUE].status;

    Check[status];

};


RetryRegistration: Event.AgentProcedure = {
    IF RegisterNow[first: FALSE].allOk THEN remove ← TRUE;
};


RetryProductFactoring: Event.AgentProcedure = {
    IF NOT ProductFactoring.Enabled[option: ConverterPFOptions.conASCII] THEN
        {
        msg: XString.ReaderBody ← CvAscii.GetMessage[notPF];

        Attention.Post[@msg];
        remove ← FALSE;
        }
    ELSE
        {
        Process.Detach[FORK AvoidDeadlock[]];
        remove ← TRUE;
        };
```

```
};


<<
--=============================================================================
AvoidDeadlock
* Finish doing registrations in another process, to make sure we don't try to AddDependency from inside of an AgentProcedure.
--=============================================================================
>>
AvoidDeadlock: PROC = {
    Process.Pause[Process.SecondsToTicks[2]];  --/* give other process a chance */
    [] ← RegisterNow[first: TRUE];
};


Register: PROCEDURE = {
    IF NOT ProductFactoring.Enabled[option: ConverterPFOptions.conASCII] THEN
        {
        msg: XString.ReaderBody ← CvAscii.GetMessage[notPF];
        logon: Event.EventType ← Atom.MakeAtom["LogonCompleted"L];

        Attention.Post[@msg];
        [] ← Event.AddDependency[
          agent: RetryProductFactoring,
          myData: NIL,
          event: logon];
        }
    ELSE [] ← RegisterNow[first: TRUE]; -- OK
};

--/* MAIN code */

Init[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
30-Jun-87 12:39:59 - Caro - MDS relief, RetryProductFactoring
```

```
-- File: CvAsciiMsgFileImpl.mesa
-- Last Revised by: Erickson                    17-Dec-87 16:06:35
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  ApplicationFolderExtra
    USING [InitMessages],
  CvAscii,
  NSFile
    USING [Error],
  Runtime
    USING [UnboundProcedure],
  XMessage
    USING [AllocateMessages, Get, Handle, MsgEntry, RegisterMessages],
  XString
    USING [FromSTRING, nullReaderBody, ReaderBody];

CvAsciiMsgFileImpl: PROGRAM
IMPORTS
  ApplicationFolderExtra, NSFile, Runtime, XMessage, XString
EXPORTS CvAscii =
BEGIN

--====================
-- GLOBALS
--====================

h: XMessage.Handle ← NIL;

--====================
-- SIGNALS
--====================

NoMessageFile: ERROR = CODE;

--====================
-- PUBLIC PROCEDURES
--====================

GetMessage: PUBLIC PROCEDURE [msg: CvAscii.MessageKey] RETURNS [msgRb: XString.ReaderBody] = {
    IF h # NIL THEN RETURN[h.Get[msg.ORD]];
    RETURN[XString.nullReaderBody];
};

--====================
-- PROCEDURES
--====================

InitMessages: PROCEDURE = {
    internalName: XString.ReaderBody ← XString.FromSTRING["FC ASCII Documents"L];
    messageFile: XString.ReaderBody ← XString.FromSTRING["MessageFile"L];

    h ← ApplicationFolderExtra.InitMessages[
      internalName: @internalName,
      label: @messageFile,
      domainIndex: 0 ! ANY => {h ← NIL; CONTINUE}];
    IF h = NIL THEN
        InitFromArray[];
};


InitFromArray: PROC = {
    h ← XMessage.AllocateMessages["Ascii Conversion"L, CvAscii.MessageKey.LAST.ORD.SUCC, NIL, NIL];

    Init0to24[];
    Init25toLAST[];
};


Init0to24: PROC = {
    msgArray: ARRAY CvAscii.MessageKey[asciiSrcDoc..lastPsheetItem] OF XMessage.MsgEntry ← [
      asciiSrcDoc: [
        msgKey: CvAscii.MessageKey.asciiSrcDoc.ORD,
        msg: XString.FromSTRING["ASCII Document"L],
        type: userMsg,
        translationNote: "Label for source of conversion"L,
        translatable: FALSE,
        id: 0],
      asciiDstDoc: [
        msgKey: CvAscii.MessageKey.asciiDstDoc.ORD,
        msg: XString.FromSTRING["ASCII Document (7 bit, 8 bit, or PC ASCII)"L],
        type: userMsg,
        translationNote: "Label for destination of conversion"L,
        translatable: FALSE,
        id: 1],
      paraEndsWith: [
        msgKey: CvAscii.MessageKey.paraEndsWith.ORD,
        msg: XString.FromSTRING["Paragraph Ends With"L],
        type: pSheetItem,
        translationNote: "Tag for text item, should read as if user were filling in the blank/completing the sentence"L,
        translatable: TRUE,
        id: 2],
      asciiEncoding: [
        msgKey: CvAscii.MessageKey.asciiEncoding.ORD,
        msg: XString.FromSTRING["ASCII Encoding"L],
```

```
        type: pSheetItem,
        translationNote: "Choice item tag"L,
        translatable: TRUE,
        id: 3],
      asciiEncodingChoices: [
        msgKey: CvAscii.MessageKey.asciiEncodingChoices.ORD,
        msg: XString.FromSTRING["7 bit or 8 bit ISO/ASCII:0@PC ASCII:1"L],
        type: argList,
        translationNote: "Choices that go with id#3"L,
        translatable: FALSE,
        id: 4],
      font: [
        msgKey: CvAscii.MessageKey.font.ORD,
        msg: XString.FromSTRING["Font"L],
        type: pSheetItem,
        translationNote: "Choice item tag"L,
        translatable: TRUE,
        id: 5],
      fontSuffix: [
        msgKey: CvAscii.MessageKey.fontSuffix.ORD,
        msg: XString.FromSTRING["spacing"L],
        type: pSheetItem,
        translationNote: "Choice item suffix for choices from id#7"L,
        translatable: TRUE,
        id: 6],
      fontChoices: [
        msgKey: CvAscii.MessageKey.fontChoices.ORD,
        msg: XString.FromSTRING["Proportional:0@Fixed:1"L],
        type: argList,
        translationNote: "Choices that go with id#5 and id#6"L,
        translatable: TRUE,
        id: 7],
      ignoreTrailing: [
        msgKey: CvAscii.MessageKey.ignoreTrailing.ORD,
        msg: XString.FromSTRING["IGNORE TRAILING WHITE SPACE"L],
        type: pSheetItem,
        translationNote: "Boolean item"L,
        translatable: TRUE,
        id: 8],
      ascii3wayChoices: [
        msgKey: CvAscii.MessageKey.ascii3wayChoices.ORD,
        msg: XString.FromSTRING["7 bit ISO/ASCII:0@8 bit ISO/ASCII:1@PC ASCII:2"L],
        type: argList,
        translationNote: "Another form for the choice item that goes with id#3"L,
        translatable: TRUE,
        id: 9],
      lineLen: [
        msgKey: CvAscii.MessageKey.lineLen.ORD,
        msg: XString.FromSTRING["Line Length"L],
        type: pSheetItem,
        translationNote: "Choice item tag for choices in id#11"L,
        translatable: TRUE,
        id: 10],
      lineLenChoices: [
        msgKey: CvAscii.MessageKey.lineLenChoices.ORD,
        msg: XString.FromSTRING["Unlimited:0@Limited:1"L],
        type: argList,
        translationNote: "Choices that go with id#10"L,
        translatable: TRUE,
        id: 11],
      charsSuffix: [
        msgKey: CvAscii.MessageKey.charsSuffix.ORD,
        msg: XString.FromSTRING["characters"L],
        type: pSheetItem,
        translationNote: "Suffix for number item -- to be read e.g. '[80] characters'"L,
        translatable: TRUE,
        id: 12],
      wordWrap: [
        msgKey: CvAscii.MessageKey.wordWrap.ORD,
        msg: XString.FromSTRING["WORD WRAP"L],
        type: pSheetItem,
        translationNote: "Boolean item, indicating that text lines should break only on the white space between words"L,
        translatable: TRUE,
        id: 13],
      endLine: [
        msgKey: CvAscii.MessageKey.endLine.ORD,
        msg: XString.FromSTRING["End Line With"L],
        type: pSheetItem,
        translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
        translatable: TRUE,
        id: 14],
      endPara: [
        msgKey: CvAscii.MessageKey.endPara.ORD,
        msg: XString.FromSTRING["End Paragraph With"L],
        type: pSheetItem,
        translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
        translatable: TRUE,
        id: 15],
      replaceUnknown: [
        msgKey: CvAscii.MessageKey.replaceUnknown.ORD,
        msg: XString.FromSTRING["Replace Unknown Char. With"L],
        type: pSheetItem,
        translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence.  'Character' is abbreviated
to make the psheet fit"L,
        translatable: TRUE,
        id: 16],
      replaceOffice: [
```

```
        msgKey: CvAscii.MessageKey.replaceOffice.ORD,
        msg: XString.FromSTRING["Replace Office Keyboard Char. With"L],
        type: pSheetItem,
        translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence.  'Character' is abbreviated
to make the psheet fit"L,
        translatable: TRUE,
        id: 17],
      convertTables: [
        msgKey: CvAscii.MessageKey.convertTables.ORD,
        msg: XString.FromSTRING["CONVERT TABLES INTO COLUMNAR TEXT"L],
        type: pSheetItem,
        translationNote: "Boolean item"L,
        translatable: TRUE,
        id: 18],
      simulateFrames: [
        msgKey: CvAscii.MessageKey.simulateFrames.ORD,
        msg: XString.FromSTRING["SIMULATE FRAMES"L],
        type: pSheetItem,
        translationNote: "Boolean item.  Used to indicated whether frames in a document are simulated in the ASCII document or not."L,
        translatable: TRUE,
        id: 19],
      spare0: [
        msgKey: CvAscii.MessageKey.spare0.ORD,
        msg: XString.FromSTRING[" "L],
        type: others,
        translationNote: "DO NOT TRANSLATE -- spare key"L,
        translatable: TRUE,
        id: 20],
      spare1: [
        msgKey: CvAscii.MessageKey.spare1.ORD,
        msg: XString.FromSTRING[" "L],
        type: others,
        translationNote: "DO NOT TRANSLATE -- spare key"L,
        translatable: TRUE,
        id: 21],
      spare2: [
        msgKey: CvAscii.MessageKey.spare2.ORD,
        msg: XString.FromSTRING[" "L],
        type: others,
        translationNote: "DO NOT TRANSLATE -- spare key"L,
        translatable: TRUE,
        id: 22],
      spare3: [
        msgKey: CvAscii.MessageKey.spare3.ORD,
        msg: XString.FromSTRING[" "L],
        type: others,
        translationNote: "DO NOT TRANSLATE -- spare key"L,
        translatable: TRUE,
        id: 23],
      lastPsheetItem: [
        msgKey: CvAscii.MessageKey.lastPsheetItem.ORD,
        msg: XString.FromSTRING[" "L],
        type: others,
        translationNote: "DO NOT TRANSLATE -- spare key"L,
        translatable: TRUE,
        id: 24]
      ];

    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};


Init25toLAST: PROC = {
    msgArray: ARRAY CvAscii.MessageKey[left..CvAscii.MessageKey.LAST] OF XMessage.MsgEntry ← [
      left: [
        msgKey: CvAscii.MessageKey.left.ORD,
        msg: XString.FromSTRING["<"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 25],
      right: [
        msgKey: CvAscii.MessageKey.right.ORD,
        msg: XString.FromSTRING[">"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 26],
      cr: [
        msgKey: CvAscii.MessageKey.cr.ORD,
        msg: XString.FromSTRING["CR"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 27],
      lf: [
        msgKey: CvAscii.MessageKey.lf.ORD,
        msg: XString.FromSTRING["LF"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 28],
      nl: [
        msgKey: CvAscii.MessageKey.nl.ORD,
        msg: XString.FromSTRING["NL"L],
        type: others,
        translationNote: "do not translate"L,
```

```
        translatable: FALSE,
        id: 29],
      ff: [
        msgKey: CvAscii.MessageKey.ff.ORD,
        msg: XString.FromSTRING["FF"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 30],
      tab: [
        msgKey: CvAscii.MessageKey.tab.ORD,
        msg: XString.FromSTRING["TAB"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 31],
      createError: [
        msgKey: CvAscii.MessageKey.createError.ORD,
        msg: XString.FromSTRING["The source object was not converted due to an error while creating the output file."L],
        type: errorMsg,
        translationNote: "Posted to attention window"L,
        translatable: TRUE,
        id: 32],
      notPF: [
        msgKey: CvAscii.MessageKey.notPF.ORD,
        msg: XString.FromSTRING["ASCII Conversion cannot be activated because required Software Option not enabled.  Please enable
Software Option, End Session, then Logon again."L],
        type: errorMsg,
        translationNote: "posted to attention window"L,
        translatable: TRUE,
        id: 33],
      paginating: [
        msgKey: CvAscii.MessageKey.paginating.ORD,
        msg: XString.FromSTRING[" paginating ... "L],
        type: userMsg,
        translationNote: "posted to attention window following 'Converting xyz ... ' converter icon message.  The leading and trailing
spaces are REQUIRED"L,
        translatable: TRUE,
        id: 34],
      skippedTableData: [
        msgKey: CvAscii.MessageKey.skippedTableData.ORD,
        msg: XString.FromSTRING[" Some data in '<>' was skipped ... "L],
        type: template,
        translationNote: "Some table data skipped.  Leading and trailing blanks REQUIRED."L,
        translatable: TRUE,
        id: 35],
      dfltMeta: [
        msgKey: CvAscii.MessageKey.dfltMeta.ORD,
        msg: XString.FromSTRING["<CR><LF>"L],
        type: others,
        translationNote: "do not translate, default value for text items"L,
        translatable: FALSE,
        id: 36],
      dfltChar: [
        msgKey: CvAscii.MessageKey.dfltChar.ORD,
        msg: XString.FromSTRING["?"L],
        type: others,
        translationNote: "do not translate, default value for text items"L,
        translatable: FALSE,
        id: 37],
      prefix: [
        msgKey: CvAscii.MessageKey.prefix.ORD,
        msg: XString.FromSTRING["CvAscii"L],
        type: others,
        translationNote: "do not translate, internal file name prefix"L,
        translatable: FALSE,
        id: 38],
      doneFailed: [
        msgKey: CvAscii.MessageKey.doneFailed.ORD,
        msg: XString.FromSTRING["Unrecoverable error writing ASCII conversion properties.  Cancel the property sheet and use a new
converter icon."L],
        type: errorMsg,
        translationNote: "Posted when user selects Done on property sheet, if there is an NSFile or other error"L,
        translatable: TRUE,
        id: 39],
      backstop: [
        msgKey: CvAscii.MessageKey.backstop.ORD,
        msg: XString.FromSTRING["Problem: the details section could not be created."L],
        type: pSheetItem,
        translationNote: "For some reason, creation of the client details window failed.  This string is put in the formwindow
instead."L,
        translatable: TRUE,
        id: 40],
      metaError: [
        msgKey: CvAscii.MessageKey.metaError.ORD,
        msg: XString.FromSTRING["The selected text item contains an error.  Please correct it."L],
        type: errorMsg,
        translationNote: "This message is posted to the Attention window when the user tries to Done or Start a sheet with a text/syntax
error.  Text syntax is described in the Reference Library documentation for ASCII."L,
        translatable: TRUE,
        id: 41],
      charsOutOfBounds: [
        msgKey: CvAscii.MessageKey.charsOutOfBounds.ORD,
        msg: XString.FromSTRING["The line length limit must be between 10 and 256 characters, inclusive. Please reenter."L],
        type: errorMsg,
        translationNote: "Posted when user tries to Done or Start a sheet with an invalid numeric value."L,
        translatable: TRUE,
```

```
        id: 42],
      fatalError: [
        msgKey: CvAscii.MessageKey.fatalError.ORD,
        msg: XString.FromSTRING[" conversion failed with an unrecoverable error "L],
        type: errorMsg,
        translationNote: "Posted if NSFile or other error in conversion.  Note that leading and trailing blanks are required."L,
        translatable: TRUE,
        id: 43],
      extraErr0: [
        msgKey: CvAscii.MessageKey.extraErr0.ORD,
        msg: XString.FromSTRING[" Unrecoverable ASCII conversion error: damaged converter icon. "L],
        type: errorMsg,
        translationNote: "Blanks are required.  Posted if the conversion cannot read properties from the converter icon."L,
        translatable: TRUE,
        id: 44],
      extraErr1: [
        msgKey: CvAscii.MessageKey.extraErr1.ORD,
        msg: XString.FromSTRING["The number in the highlighted field is invalid. Please reenter."L],
        type: errorMsg,
        translationNote: "Posted when the user tries to Done or Start a sheet with text in a numeric field."L,
        translatable: TRUE,
        id: 45],
      aToVDfltMeta: [
        msgKey: CvAscii.MessageKey.aToVDfltMeta.ORD,
        msg: XString.FromSTRING["<CR>"L],
        type: others,
        translationNote: "do not translate, default value for text items"L,
        translatable: FALSE,
        id: 46]

<<
      : [
        msgKey: CvAscii.MessageKey.USEAGAINTOREPLACETHISSTRING.ORD,
        msg: XString.FromSTRING[""L],
        type: ,
        translationNote: ""L,
        translatable: TRUE,
        id: ],
>>
      ];

    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};



--/* MAIN line code */

InitMessages[! NSFile.Error, Runtime.UnboundProcedure => NoMessageFile];

END...


LOG
24-Apr-85 12:12:27 - MSchneider - CREATED from SampleBWSApplicationMsgFileImpl
10-May-85 10:56:18 - MSchneider - used correct ApplicationFolder name
28-May-85  9:28:54 - MSchneider - moved localZone into procedure, added use of BWSZone
24-Jun-85 14:33:55 - MSchneider - made "MessageFile" be "MessageFile" in entry name
 9-Jul-85 11:12:31 - MSchneider - added ERROR NoMessageFile
26-Feb-87 14:59:12 - Caro - Upgraded to VP 2.0 (delete 90% of code)
 8-Apr-87 11:43:56 - Caro - Catch ANY error raised from InitMessages
26-Jun-87 11:10:51 - Caro - Made #44 a real error
19-Aug-87 10:51:37 - Caro - Reworded several messages and transNotes
24-Nov-87 17:01:04 - Erickson - added aToVDfltMeta (ID = 46) to change default for ascii to ViewPoint treatment of paraEndsWith.
17-Dec-87 16:04:02 - Erickson - AR 16414 - made #45 a real error, bad number input.
```

```
-- File: CvAsciiParseImpl.mesa
-- Last Revised by: Caro                        29-Jun-87 11:31:40
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, TAB],
  Attention
    USING [Post],
  CvAscii
    USING [Common, GetMessage, MessageKey],
  FormWindow
    USING [SetSelection, SetInputFocus],
  String
    USING [AppendChar, CopyToNewString, MakeString, StringBoundsFault],
  XChar
    USING [Character, Code, not],
  XString
    USING [AppendChar, ClearWriter, CopyToNewReaderBody,
    Empty, Equal, First, FromSTRING, FreeReaderBytes, FreeWriterBytes,
    InvalidEncoding, Lop, NewWriterBody,
    Reader, ReaderBody, ReaderFromWriter, ValidateReader, Writer, WriterBody];

<<
--============================================================================
-- OVERVIEW:

Parse text items containing meta characters into strings.
--============================================================================
>>

CvAsciiParseImpl: PROGRAM
IMPORTS
  Attention, CvAscii, FormWindow, String, XChar, XString
EXPORTS
  CvAscii =
BEGIN

--====================
-- CONSTANTS
--====================

max: CARDINAL = 10;
maxAbbr: CARDINAL = 3;  --/* abbreviations only up to 3 characters */
maxOctals: CARDINAL = 3;  --/* need exactly 3 octal digits */

--====================
-- TYPES
--====================

ParseStates: TYPE = {
    entry,
    beginMeta,
    doOctal,
    doAbbrev
    };
--====================
-- SIGNALS
--====================

ParseError: SIGNAL [err: ErrType ← syntaxError, start, pos: CARDINAL] = CODE;

ErrType: TYPE =
    {
    syntaxError,
    invalidMeta,
    unknownAbbr,
    invalidOctal,
    invalidEncoding
    };

--====================
-- PUBLIC PROCEDURES
--====================

ParseItem: PUBLIC PROC [my: CvAscii.Common, r: XString.Reader, item: CvAscii.MessageKey, buf: XString.Writer ← NIL] RETURNS [ok: BOOLEAN,
ls: LONG STRING] = {
    bufRb: XString.WriterBody;
    tmpRb: XString.ReaderBody;
    msgRb: XString.ReaderBody;
    clientBuf: BOOLEAN;

    IF buf = NIL THEN
        {
        bufRb ← XString.NewWriterBody[maxLength: 30, z: my.z];
        buf ← @bufRb;
        clientBuf ← FALSE;
        }
    ELSE
        clientBuf ← TRUE;

    BEGIN
      ENABLE ParseError =>
        {
        msgRb ← CvAscii.GetMessage[metaError];
```

```
        IF my.window = NIL OR item = CvAscii.MessageKey.FIRST THEN GOTO notOK;

        FormWindow.SetSelection[
          window: my.window,
          item: item.ORD,
          firstChar: start,
          lastChar: pos];
        FormWindow.SetInputFocus[
          window: my.window,
          item: item.ORD,
          beforeChar: pos];
        Attention.Post[@msgRb];
        ls ← NIL;
        GOTO notOK;
        };

    tmpRb ← XString.CopyToNewReaderBody[r: r, z: my.z];
    ls ← ParseToLS[text: @tmpRb, z: my.z, buf: buf];

    --/* test for invalid encoding */
    IF my.owner = AtoVdst THEN
        {
        msgRb ← XString.FromSTRING[ls];
        XString.ValidateReader[@msgRb ! XString.InvalidEncoding =>
           SIGNAL ParseError[
             err: invalidEncoding,
             start: 0,
             pos: CARDINAL.LAST]];
         };

    ok ← TRUE;
    EXITS notOK => ok ← FALSE;
    END;

    IF NOT clientBuf THEN
        XString.FreeWriterBytes[buf];
    XString.FreeReaderBytes[r: @tmpRb, z: my.z];
};

--====================
-- PROCEDURES
--====================


ParseToLS: PROC [text: XString.Reader, z: UNCOUNTED ZONE, buf: XString.Writer] RETURNS [ls: LONG STRING ← NIL] = {
    rb: XString.ReaderBody ← CvAscii.GetMessage[left];
    state: ParseStates ← entry;
    start,
    pos: CARDINAL ← 0;
    octals,
    abbrs: CARDINAL ← 0;
    cr: XString.ReaderBody;
    lf: XString.ReaderBody;
    nl: XString.ReaderBody;
    ff: XString.ReaderBody;
    tab: XString.ReaderBody;
    left: XChar.Character;
    right: XChar.Character;
    xc: XChar.Character;
    c: CHARACTER;
    octalValue: CARDINAL[0..255];

    --/* get < and > */
    left ← XString.First[@rb];
    rb ← CvAscii.GetMessage[right];
    right ← XString.First[@rb];

    --/* initialize strings */
    IF XString.Empty[text] THEN
        RETURN[ls: NIL]
    ELSE
        ls ← String.MakeString[z: z, maxlength: max];
    cr ← CvAscii.GetMessage[cr];
    lf ← CvAscii.GetMessage[lf];
    nl ← CvAscii.GetMessage[nl];
    ff ← CvAscii.GetMessage[ff];
    tab ← CvAscii.GetMessage[tab];

    --/* lop through string */
    DO
      ENABLE
        {
        String.StringBoundsFault =>
          {
          ns ← String.CopyToNewString[s: ls, z: z, longer: max];
          z.FREE[@ls];
          ls ← ns;
          RESUME[ns];
          };
        UNWIND =>
          {
          IF ls # NIL THEN z.FREE[@ls];
          };
        };

        xc ← XString.Lop[text];
        IF xc = XChar.not THEN
```

```
        {
        IF state = entry THEN
            EXIT
        ELSE
            SIGNAL ParseError[err: syntaxError, start: start, pos: pos];
        };

    SELECT state FROM
      entry =>
        {
        IF xc = left THEN
            state ← beginMeta
        ELSE
            {
            c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
            String.AppendChar[s: ls, c: c];
            state ← entry;
            };
        pos ← pos + 1;
        };
      beginMeta =>
        {
        start ← pos;
        c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
        SELECT c FROM
            IN ['0..'3] =>
                {
                state ← doOctal;
                octals ← 1;
                octalValue ← c - '0;
                };
            'C, 'F, 'L, 'N, 'T, '< =>
                {
                state ← doAbbrev;
                XString.ClearWriter[buf];  --/* collect abbreviation here */
                XString.AppendChar[to: buf, c: xc];
                abbrs ← 1;
                };
            ENDCASE =>
              SIGNAL ParseError[err: invalidMeta, start: start, pos: pos];
        pos ← pos + 1;
        };
      doOctal =>
        {
        c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
        IF xc = right THEN
            {
            IF start = pos THEN
                SIGNAL ParseError[err: invalidMeta, start: start, pos: pos + 1];
            IF octals < maxOctals OR octalValue > 377B THEN
                SIGNAL ParseError[err: invalidOctal, start: start, pos: pos];
            c ← LOOPHOLE[octalValue, CHARACTER];
            String.AppendChar[s: ls, c: c];
            state ← entry;
            }
        ELSE IF octals >= maxOctals THEN
            SIGNAL ParseError[err: invalidOctal, start: start, pos: pos]
        ELSE IF NOT c IN ['0..'7] THEN
            SIGNAL ParseError[err: invalidOctal, start: start, pos: pos]
        ELSE
            {
            octalValue ← (octalValue * 8) + (c - '0);
            octals ← octals + 1;
            state ← doOctal;
            };
        pos ← pos + 1;
        };
      doAbbrev =>
        {
        IF xc = right THEN
            {
            tmp: XString.Reader ← XString.ReaderFromWriter[buf];

            IF start = pos THEN
                SIGNAL ParseError[err: invalidMeta, start: start, pos: pos + 1];
            IF abbrs > maxAbbr THEN
                SIGNAL ParseError[err: unknownAbbr, start: start, pos: pos];
            SELECT TRUE FROM
                XString.Equal[r1: tmp, r2: @cr] =>
                    String.AppendChar[s: ls, c: Ascii.CR];
                XString.Equal[r1: tmp, r2: @lf] =>
                    String.AppendChar[s: ls, c: Ascii.LF];
                XString.Equal[r1: tmp, r2: @nl] =>
                    {
                    String.AppendChar[s: ls, c: Ascii.CR];
                    String.AppendChar[s: ls, c: Ascii.LF];
                    };
                XString.Equal[r1: tmp, r2: @tab] =>
                    String.AppendChar[s: ls, c: Ascii.TAB];
                XString.Equal[r1: tmp, r2: @ff] =>
                    String.AppendChar[s: ls, c: Ascii.FF];
                abbrs = 1 AND c = '< =>
                    String.AppendChar[s: ls, c: '<];
                ENDCASE =>
                    SIGNAL ParseError[err: unknownAbbr, start: start, pos: pos];
            state ← entry;
            }
```

```
                ELSE
                    {
                    XString.AppendChar[to: buf, c: xc];
                    abbrs ← abbrs + 1;
                    state ← doAbbrev;
                    };
                pos ← pos + 1;
                };
            ENDCASE;
    ENDLOOP;
};


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:28:54 - Caro - Added test for MessageKey.FIRST to ParseItem
29-Jun-87 11:33:00 - Caro - Added validation to ParseItem
```

```
-- File: CvAsciiToVPImpl.mesa
-- Last Revised by: Shinsato                    12-Feb-88 13:00:11
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (C) 1987, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, NUL, SP, TAB],
  BackgroundProcess
    USING [ResetUserAbort, UserAbort],
  BWSZone
    USING [Permanent],
  Converter
    USING [ConvertProc, CvData, DependentOptionProc, GetPOption, PostMessage],
  ConverterMsg,
  CvAscii,
  DocInterchangeDefs
    USING [AppendNewParagraph, AppendPageBreak, AppendText, CheckAbortProc,
    Doc, Error, FinishCreation, FinishCreationStatus,
    PaginateOption,
    SetCurrentParagraphProps, StartCreation, StartCreationStatus],
  DocInterchangePropsDefs
    USING [Family, FontPropsRecord, GetFontPropsDefaults, GetPagePropsDefaults,
    GetParaPropsDefaults, PagePropsRecord, ParaPropsRecord],
  Environment
    USING [Block, Byte, bytesPerPage, wordsPerPage],
  NSFile
    USING [Close, Error, GetReference, Handle, Logoff,
    nullHandle, OpenByReference, Reference, Session],
  NSFileStream
    USING [Create, Handle],
  Space
    USING [ScratchMap, Unmap],
  Stream
    USING [CompletionCode, Delete, GetBlock],
  TIP
    USING [ResetUserAbort, UserAbort],
  XCharSet0
    USING [Make],
  XString
    USING [AppendChar, ByteLength,
    Character, CharacterLength, ClearWriter, FreeWriterBytes,
    InvalidEncoding, NewWriterBody, Reader, ReaderBody, ReaderFromWriter,
    Writer, WriterBody, WriterInfo];

<<
--============================================================================
-- OVERVIEW:

Ascii to VP conversion.
--============================================================================
>>

CvAsciiToVPImpl: PROGRAM
IMPORTS
  BackgroundProcess, BWSZone, Converter, ConverterMsg, CvAscii,
  DocInterchangeDefs, DocInterchangePropsDefs,
  NSFile, NSFileStream, Space, Stream, TIP, XCharSet0, XString
EXPORTS
  CvAscii =
BEGIN

--====================
-- CONSTANTS
--====================

lineHtInPoints: CARDINAL = 12;
accentLineHt: CARDINAL = 16;  --/* allow extra space for accents */
fixedWidthInPoints: CARDINAL = 7;

maxPara: CARDINAL = 8 * 1024;
bufPages: CARDINAL = (maxPara + Environment.bytesPerPage - 1) / Environment.bytesPerPage;
paraLen: CARDINAL = maxPara/4;

pcTerminal: DocInterchangePropsDefs.Family = VAL[54];

words: CARDINAL = SIZE[CtoVPCharMap];

stopsAt: CARDINAL = 8;  --/* tab stops every eight characters */
tabStopCount: CARDINAL = (132/stopsAt)+1;  --/* 132 columns max */
leftFixed: CARDINAL = 12;
rightFixed: CARDINAL = 1;

aHyphen: CHARACTER = 055C;

--====================
-- TYPES
--====================

AVData: TYPE = LONG POINTER TO AVDataObj;
AVDataObj: TYPE = RECORD [
    source: NSFile.Handle,
    input: NSFileStream.Handle,  --/* created from source */
    cvData: Converter.CvData,
    session: NSFile.Session,
    src: CvAscii.Common,  --/* common data distinguished by owning formwindow */
```

```
    dst: CvAscii.Common,
    background: BOOLEAN,
    fontProps: DocInterchangePropsDefs.FontPropsRecord,
    paraProps: DocInterchangePropsDefs.ParaPropsRecord,
    pageProps: DocInterchangePropsDefs.PagePropsRecord,
    doc: DocInterchangeDefs.Doc,
    blk: Environment.Block,   --/* primary input buffer */
    state: AVState,
    z: UNCOUNTED ZONE];

--/* the various states of the StateMachine */
AVState: TYPE =
    {
    entry,
    append,
    ignoreTrailing,
    maxExceeded,
    endPara
    };


CtoVPCharMap: TYPE = ARRAY CHARACTER OF XString.Character;

--====================
-- GLOBALS
--====================

Global: TYPE = RECORD [
    pcmap: LONG POINTER TO CtoVPCharMap,
    isomap: LONG POINTER TO CtoVPCharMap,
    pz: UNCOUNTED ZONE];

g: Global;

--====================
-- PUBLIC PROCEDURES
--====================

AsciiToVP: PUBLIC Converter.ConvertProc = {
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];
>>
  ENABLE CvAscii.Problem, NSFile.Error, XString.InvalidEncoding =>
      {
      msgRb: XString.ReaderBody ← CvAscii.GetMessage[fatalError];

      Post[msgRb, cvData];
      CONTINUE;
      };

    IF source = NSFile.nullHandle THEN RETURN;
    dest ← AtoV[source, cvData, session, srcInstance, dstInstance, background];
};

<<
--============================================================================
Both DependentOptionProcs create instance data with CreateCommon.  The data is distinguished by the owner variable.  The CommonObj within
CvAscii.CommonData is the data structure written to the client file stored as the icon properties.  Only those fields pertaining to the
owner are used.
--============================================================================
>>

AsciiToVPSrcOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvAscii.Owners ← AtoVsrc;

    menuItemProc ← CvAscii.CommonMenu;
    destroy ← CvAscii.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvAscii.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvAscii.Problem => {owner ← backstop; instance ←
NIL; CONTINUE}]
    ELSE
        {
        my: CvAscii.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };

    --/* make formwindow */
    CvAscii.CreateFW[instance, window, owner];
};


AsciiToVPDstOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvAscii.Owners ← AtoVdst;

    menuItemProc ← CvAscii.CommonMenu;
    destroy ← CvAscii.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvAscii.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvAscii.Problem => {owner ← backstop; instance ←
```

```
NIL; CONTINUE}]
    ELSE
        {
        my: CvAscii.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };

    --/* make formwindow */
    CvAscii.CreateFW[instance, window,  owner];
};

--------------------
-- PROCEDURES
--------------------

AtoV: Converter.ConvertProc = {
    aborted: BOOLEAN ← FALSE;
    start: DocInterchangeDefs.StartCreationStatus ← lastAvailable;
    finish: DocInterchangeDefs.FinishCreationStatus ← lastAvailable;
    avData: AVDataObj;
    pOption: DocInterchangeDefs.PaginateOption;
    docSession: NSFile.Session;
    dst,
    src: CvAscii.Common ← NIL;
    --/* local proc */
    POption: PROCEDURE RETURNS [DocInterchangeDefs.PaginateOption] = INLINE
        {
        SELECT Converter.GetPOption[] FROM
          compress => RETURN[compress];
          simple => RETURN[simple];
          none => RETURN[none];
        ENDCASE => ERROR;
        };

    --/* begin code */
    --/* initialize instance data */
    IF dstInstance = NIL THEN  --/* ASSERT: srcInstance also NIL */
        {
        ENABLE NSFile.Error, CvAscii.Problem =>
            {
            msgRb: XString.ReaderBody ← CvAscii.GetMessage[extraErr0];  --" Unrecoverable ASCII conversion error: damaged converter icon.
"
            Converter.PostMessage[
              msg: @msgRb,
              cvData: cvData,
              cr: FALSE,
              clear: FALSE];
            IF src # NIL THEN CvAscii.DestroyCommon[src];
            GOTO terminate;
            };
        key: CvAscii.MessageKey ← CvAscii.MessageKey.FIRST;

        --/* assume both are NIL */
        src ← CvAscii.CreateCommon[cvData, FALSE, NIL, AtoVsrc];
        dst ← CvAscii.CreateCommon[cvData, FALSE, NIL, AtoVdst];

        src.text[paraEndsWith] ← CvAscii.ParseItem[
          my: src,
          r: @src.textRb[paraEndsWith],
          item: key].ls;

        dst.text[atovReplaceUnknown] ← CvAscii.ParseItem[
          my: dst,
          r: @dst.textRb[atovReplaceUnknown],
          item: key].ls;
        EXITS terminate => RETURN;
        }
    ELSE
        {
        src ← srcInstance;
        dst ← dstInstance;
        };

    avData ← [
         source: source,
         input: [NIL],
         cvData: cvData,
         session: session,
         src: src,
         dst: dst,
         background: background,
         fontProps: TRASH,
         paraProps: TRASH,
         pageProps: TRASH,
         doc: TRASH,
         blk: [Space.ScratchMap[count: bufPages], 0, maxPara],
         state: entry,
         z: dst.z];

    BEGIN
      ENABLE
        {
        DocInterchangeDefs.Error => GOTO err;
        UNWIND =>
            {
            avData.blk.blockPointer ← Space.Unmap[pointer: avData.blk.blockPointer];
```

```
            IF srcInstance = NIL THEN CvAscii.DestroyCommon[src];
            IF dstInstance = NIL THEN CvAscii.DestroyCommon[dst];
            src ← dst ← NIL;
            };
        };

    --/* open stream on source */
    avData.input ← NSFileStream.Create[
      file: avData.source,
      closeOnDelete: FALSE,
      session: avData.session ! NSFile.Error => {avData.input ← [NIL]; GOTO err}];

    --/* initialize */
    pOption ← POption[];
    DocInterchangePropsDefs.GetFontPropsDefaults[@avData.fontProps];
    DocInterchangePropsDefs.GetParaPropsDefaults[@avData.paraProps];
    DocInterchangePropsDefs.GetPagePropsDefaults[@avData.pageProps];

    --/* initialize tabs */
    avData.paraProps.basicProps.defaultTabStopSpacing ← (stopsAt*fixedWidthInPoints);

    --/* apply initial parms */
    SELECT avData.dst.f.font FROM
        CvAscii.proportional => NULL;
        CvAscii.fixed =>
            {
            avData.fontProps.fontDesc.family ← IF avData.src.f.atovAsciiEncoding = CvAscii.pcAscii THEN
                pcTerminal
            ELSE
                terminal;
            avData.fontProps.fontDesc.pointSize ← 12;
            avData.pageProps.leftMarginWidth ← leftFixed;
            avData.pageProps.rightMarginWidth ← rightFixed;
            };
        ENDCASE;

    --/* set "AFTER" para spacing by counting CRs in paraEndsWith string */
    BEGIN
    lcount: CARDINAL ← 0;
    eop: LONG STRING ← avData.src.text[paraEndsWith];

    IF eop # NIL THEN
      FOR i: CARDINAL IN [0..eop.length) DO
        IF eop[i] = Ascii.CR THEN lcount ← lcount + 1;
      ENDLOOP;
    --/* lcount = 0 => default */
    --/* lcount = 1 => single spacing */
    --/* lcount = 2 => double, etc. */
    IF lcount > 1 THEN
        avData.paraProps.basicProps.postLeading ← lineHtInPoints * (lcount - 1);
    END;

    --/* StartCreation checks process priority to determine forkedness */
    [doc: avData.doc, status: start] ← DocInterchangeDefs.StartCreation[
        paginateOption: pOption,
        initialFontProps: @avData.fontProps,
        initialParaProps: @avData.paraProps,
        initialPageProps: @avData.pageProps ! NSFile.Error => {
          IF error = [space[mediumFull]] THEN
            start ← notEnoughDiskSpace
          ELSE
            start ← lastAvailable;
          CONTINUE}];

    SELECT start FROM
        ok => NULL;
        notEnoughDiskSpace =>
            {
            Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], avData.cvData];
            GOTO err;
            };
        ENDCASE =>
            {
            Post[ConverterMsg.Get[ConverterMsg.kunknownProblem], avData.cvData];
            GOTO err;
            };

    --/* enter state graph */
    BEGIN
      ENABLE ABORTED => {aborted ← TRUE; CONTINUE};
        StateMachine[@avData];
    END;

    --/* paginating */
    IF pOption # none THEN
        {
        mrb: XString.ReaderBody ← CvAscii.GetMessage[paginating];
        Converter.PostMessage[
          msg: @mrb,
          cvData: cvData,
          cr: FALSE,
          clear: FALSE];
        };
    --/* user may have partial doc after an abort, so allow paginate/finish */
    --/* reset abort tests.  User must abort paginate separately. */
    IF aborted THEN
```

```
        {
        IF avData.background THEN
            BackgroundProcess.ResetUserAbort[]
        ELSE
            TIP.ResetUserAbort[NIL];
        };

    --/* paginate and finish */
    [docFile: dest, session: docSession, status: finish] ← DocInterchangeDefs.FinishCreation[
      docPtr: @avData.doc,
      checkAbortProc: UserAbortsPaginate,
      checkAbortClientData: @avData];

    IF finish = aborted THEN
        {
        aborted ← TRUE;
        Post[ConverterMsg.Get[ConverterMsg.kuserAbort], cvData];
        };

    --/* re-open dest in session */
    IF dest # NSFile.nullHandle THEN
        {
          ENABLE NSFile.Error =>
            {
            NSFile.Close[dest, docSession ! NSFile.Error => CONTINUE];
            dest ← NSFile.nullHandle;
            CONTINUE;
            };
        tmpRef: NSFile.Reference;
        tmp: NSFile.Handle ← dest;

        tmpRef ← NSFile.GetReference[file: dest, session: docSession];
        dest ← NSFile.OpenByReference[reference: tmpRef, session: avData.session];
        NSFile.Close[tmp, docSession];
        --/* if this process is clientBackground, docSession must be logged off */
        IF background THEN NSFile.Logoff[docSession ! NSFile.Error => CONTINUE];
        };

    EXITS err => NULL;
    END;
    IF avData.input # NIL THEN Stream.Delete[avData.input];
    IF avData.blk.blockPointer # NIL THEN
        avData.blk.blockPointer ← Space.Unmap[avData.blk.blockPointer];
    --/* destroy instance data if created by this proc call */
    IF srcInstance = NIL AND src # NIL THEN CvAscii.DestroyCommon[src];
    IF dstInstance = NIL AND dst # NIL THEN CvAscii.DestroyCommon[dst];
    IF finish # ok OR aborted THEN
       Post[ConverterMsg.Get[ConverterMsg.kdataSkipped], cvData];
};


CheckAbort: PROC [background: BOOLEAN] RETURNS [yes: BOOLEAN] = INLINE {
    yes ← (background AND BackgroundProcess.UserAbort[]) OR
      (NOT background AND TIP.UserAbort[NIL]);
};


FlushText: PROC [av: AVData, para: XString.Writer] = {
    r: XString.Reader ← XString.ReaderFromWriter[para];

    IF CheckAbort[av.background] THEN ERROR ABORTED;
    IF XString.ByteLength[r] > 0 THEN
        {
        DocInterchangeDefs.AppendText[
          to: [doc[av.doc]],
          text: r,
          textEndContext: XString.WriterInfo[para].endContext,
          fontProps: @av.fontProps];
        XString.ClearWriter[para];
        };
};


Post: PROC [msgRb: XString.ReaderBody, cvData: Converter.CvData] = {
    Converter.PostMessage[
      msg: @msgRb,
      cvData: cvData,
      cr: TRUE,
      clear: FALSE];
};



<<
--============================================================================
StateMachine
This procedure implements a state graph, which is depicted in auxiliary documentation.  The state machine handles the input data
character by character, although the i/o is optimized using block buffers.  Note that the XString.Writer "para" is the output buffer
that gets appended to the document every time text is flushed (see FlushText).  Hereafter are described, briefly, the states, the entry
conditions, exit conditions, and special circumstances:

- entry
The state machine is always entered here.  The entry conditions are that the index "n" references the next character to be handled.  The
next state is determined by the value of the character "c".  The mode "ignore" determines whether white space is treated as standard
text, or as should be handled by the special ignoreTrailing state.  If the character "c" matches the first character of the end of
paragraph string "eop0", then the next state is endPara; otherwise, the next state is "append".  Note that the variable "nextState"
does NOT refer to the state executed after entry, but rather the state that the next state RETURNS TO.  Although this violates strict
```

state machine implementation algorithms, it saves logic.

- append
The state is entered with the character "c", and a valid nextState. It translates the character "c" to a VP character, and appends it to the output buffer "para". Certain special cases are handled. The exit condition is a valid nextState, which becomes "state".

- ignoreTrailing
The purpose of this state is to implement deletion of white space that precedes an end of line sequence, if the user so desires. The state is entered either from entry with "c" being whitespace, or from ignoreTrailing, with "n" indicating the next character to handle. Variables are initialized to indicate the beginning of whitespace characters. The state is exited if eop0 is found, or a nonwhitespace character is found before the end of line.

- maxExceeded
This state handles an overflow exception. It is entered if "para" is about to exceeded its limits. A new paragraph is forced if this state is entered. It returns to entry.

- endPara
This state tries to determine if the end of a paragraph has been found. It is entered if the character "c" matched eop0, or (from endPara itself) if the input text continues to match the string "s". If a paragraph ending is found, the paragraph is flushed. The state returns to entry either if there is a complete match, or of there is a mismatch. Several special cases are handled.

The state machine loops until input is exhausted.
--**********
>>

```
StateMachine: PROC [av: AVData] = {
    lastBlock: BOOLEAN ← FALSE;
    flushed: BOOLEAN ← FALSE;  --/* controls appending text to doc */
    ignore: BOOLEAN ← av.dst.f.ignoreTrailing.value;
    eop: CARDINAL ← 0;  --/* index into paraEndsWith string */
    para: XString.WriterBody ← XString.NewWriterBody[maxLength: paraLen, z: av.z];
    state: AVState ← entry;
    blankCount: CARDINAL ← 0;  --/* count of "white" characters in buffer */
    blkCount: CARDINAL ← 0;  --/* number of blocks read */
    lastBlkCount: CARDINAL;  --/* for saving "blkCount" */
    nextState: AVState;  --/* the state a state goes back to */
    getNextBlock: BOOLEAN;
    bytes: CARDINAL;
    why: Stream.CompletionCode;
    eop0: CHARACTER;   --/* first character of end-of-paragraph text */
    unknown: LONG STRING;  --/* copy of user defined replacement text */
    blanksStart: CARDINAL;  --/* index into buffer for beginning of blanks */
    oldHt: CARDINAL;
    map: LONG POINTER TO CtoVPCharMap;
    blk: LONG POINTER TO PACKED ARRAY INTEGER[0..0) OF Environment.Byte;
    n: CARDINAL;  --/* current character in blk */
    last,
    c: CHARACTER;
    accentFirst,
    accentLast,
    lowGraphFirst,
    lowGraphLast,
    hiGraphFirst,
    hiGraphLast: CHARACTER;  --/* character segment range limits */

    --/* initialize */
    IF av.src.f.atovAsciiEncoding = CvAscii.pcAscii THEN
        {
        map ← g.pcmap;
        accentFirst ← 200C;             accentLast ← 245C;
        lowGraphFirst ← 1C;             lowGraphLast ← 037C;
        hiGraphFirst ← 246C;            hiGraphLast ← 377C;
        }
    ELSE
        {
        map ← g.isomap;
        accentFirst ← 301C;             accentLast ← 376C;
        lowGraphFirst ← 241C;           lowGraphLast ← 277C;
        hiGraphFirst ← 40C;             hiGraphLast ← 40C;
        };
    --/* para is a buffer of VP characters that gets appended to the doc */
    XString.ClearWriter[@para];
    eop0 ← IF av.src.text[paraEndsWith] # NIL THEN
        av.src.text[paraEndsWith][0]
    ELSE
        Ascii.NUL;
    last ← Ascii.NUL;
    unknown ← av.dst.text[atovReplaceUnknown];
    IF unknown = NIL THEN
        {
        --/* so we don't have to test for NIL again */
        unknown ← "?"L;
        unknown.length ← 0;
        };
    oldHt ← av.paraProps.basicProps.lineHeight;
    --/* make sure getNextBlock is TRUE first time */
    n ← av.blk.stopIndexPlusOne;
    blk ← av.blk.blockPointer;

    --/* enter state graph */
    DO
        getNextBlock ← n >= av.blk.stopIndexPlusOne;
        IF getNextBlock THEN
            {
```

```
        IF lastBlock THEN
            {
            --/* might have one last character pending */
            IF state = append THEN
                {
                nextState ← entry;
                GOTO oneLastLoop;
                };
            FlushText[av, @para];
            EXIT;  --/* state graph */
            };
        IF CheckAbort[av.background] THEN ERROR ABORTED;
        av.blk.stopIndexPlusOne ← maxPara;
        [bytesTransferred: bytes, why: why] ← Stream.GetBlock[
          sH: av.input,
          block: av.blk];
        lastBlock ← why # normal;
        av.blk.stopIndexPlusOne ← bytes;
        blk ← av.blk.blockPointer;
        n ← 0;
        --/* guard against blkCount overflow */
        blkCount ← IF blkCount = CARDINAL.LAST THEN 0 ELSE blkCount + 1;
        EXITS oneLastLoop => NULL;
        };

    SELECT state FROM
        entry =>
            {
            --/* get next character */
            c ← LOOPHOLE[blk[n], CHAR];
            --/* set up next state */
            SELECT c FROM
                Ascii.SP, Ascii.TAB => IF ignore THEN
                    {
                    state ← ignoreTrailing;
                    blanksStart ← n;
                    blankCount ← 0;
                    lastBlkCount ← blkCount;
                    }
                  ELSE
                    {
                    state ← append;
                    nextState ← entry;
                    n ← n + 1;
                    };
                eop0 =>
                    {
                    state ← endPara;
                    };
                ENDCASE =>
                    {
                    state ← append;
                    nextState ← entry;
                    n ← n + 1;
                    };
            };
        append =>
            {
            --/* ASSERT: order of select arms is critically important */
            SELECT c FROM
                IN [40C..176C],
                IN [hiGraphFirst..hiGraphLast] =>
                    {
                    --/* standard characters or high graphics */
                    XString.AppendChar[to: @para, c: map[c], extra: paraLen];
                    state ← nextState;
                    };
                IN [accentFirst..accentLast] =>
                    {
                    --/* accents & foreign characters */
                    XString.AppendChar[to: @para, c: map[c], extra: paraLen];

                    --/* make sure line is bigger */
                    IF av.paraProps.basicProps.lineHeight = oldHt THEN
                        {
                        av.paraProps.basicProps.lineHeight ← accentLineHt;
                        DocInterchangeDefs.SetCurrentParagraphProps[
                          textContainer: [doc[av.doc]],
                          paraProps: @av.paraProps];
                        };

                    state ← nextState;
                    };
                Ascii.CR =>
                    {
                    IF CheckAbort[av.background] THEN ERROR ABORTED;
                    IF nextState = entry THEN
                        {
                        --/* smart white space */
                        SELECT last FROM
                            Ascii.SP,
                            Ascii.TAB,
                            Ascii.CR,
                            Ascii.LF,
                            aHyphen => NULL; --/* just drop CR */
                            ENDCASE =>
                                {
```

```
                        XString.AppendChar[
                          to: @para,
                          c: map[Ascii.SP],
                          extra: paraLen];
                        };
                    };
                --/* CR is skipped if we came from endPara */
                state ← nextState;
                };
            Ascii.LF =>
                {
                IF last # Ascii.CR AND nextState # endPara THEN
                    {
                    --/* append newline */
                    XString.AppendChar[
                      to: @para,
                      c: XCharSet0.Make[newLine],
                      extra: paraLen];
                    };
                --/* LF is skipped if we came from endPara */
                --/* or if last = CR */
                state ← nextState;
                };
            Ascii.TAB =>
                {
                --/* tab */
                XString.AppendChar[to: @para, c: map[c], extra: paraLen];
                state ← nextState;
                };
            244C =>
                {
                --/* map to dollar sign */
                XString.AppendChar[to: @para, c: map['$], extra: paraLen];
                state ← nextState;
                };
            Ascii.FF =>
                {
                --/* flush page */
                FlushText[av, @para];
                DocInterchangeDefs.AppendPageBreak[
                  to: av.doc,
                  fontProps: @av.fontProps];
                state ← nextState;
                };
            Ascii.NUL =>
                {
                --/* skip */
                state ← nextState;
                };
            IN [lowGraphFirst..lowGraphLast] =>
                {
                --/* low graphics characters */
                XString.AppendChar[to: @para, c: map[c], extra: paraLen];
                state ← nextState;
                };
            <<
            246C, 250C, 300C, IN [326C..333C], 345C =>
                {
                };
            >>
            ENDCASE =>
                {
                IF av.src.f.atovAsciiEncoding = CvAscii.pcAscii
                  AND c = 177C THEN
                    XString.AppendChar[to: @para, c: map[c], extra: paraLen]
                ELSE
                    --/* exceptions */
                    FOR i: CARDINAL IN [0..unknown.length) DO
                        XString.AppendChar[to: @para, c: map[unknown[i]], extra: paraLen];
                    ENDLOOP;
                state ← nextState;
                };
        last ← c;
        <<
        * XString.CharacterLength is an expensive operation.
        * We make the observation that
        * ByteLength >= CharacterLength ALWAYS. Therefore
        * use faster ByteLength to determine if CharacterLength should
        * be called
        >>
        IF XString.ByteLength[XString.ReaderFromWriter[@para]] > maxPara THEN
            {
            IF XString.CharacterLength[XString.ReaderFromWriter[@para]] > maxPara
              AND nextState # endPara THEN
                state ← maxExceeded;
            };
        };
    ignoreTrailing =>
        {
        --/* get next char if other than first entry */
        IF blanksStart # n THEN
            {
            last ← c;
            c ← LOOPHOLE[blk[n], CHAR];
            };
        SELECT c FROM
```

```
            Ascii.SP, Ascii.TAB =>
                {
                state ← ignoreTrailing;
                n ← n + 1;
                blankCount ← blankCount + 1;
                };
            eop0 =>
                {
                --/* end found, so skip all trailing blanks */
                state ← endPara;
                };
            Ascii.CR =>
                {
                --/* NOTE: this arm must follow the eop0 arm */
                --/* ASSERT: eop0 # Ascii.CR by order of execution */
                --/* replace CR with space, and skip blanks */
                XString.AppendChar[
                  to: @para,
                  c: map[Ascii.SP],
                  extra: paraLen];
                state ← entry;
                blankCount ← 0;
                n ← n + 1;
                };
            ENDCASE =>
                {
                IF CheckAbort[av.background] THEN ERROR ABORTED;
                --/* whoops!  Not eol, so append */
                IF lastBlkCount # blkCount THEN
                    {
                    --/* blanks straddle blocks */
                    THROUGH [1..blankCount] DO
                        XString.AppendChar[
                          to: @para,
                          c: map[Ascii.SP],
                          extra: paraLen];
                    ENDLOOP;
                    }
                ELSE
                    FOR i: CARDINAL IN [blanksStart..n) DO
                        XString.AppendChar[
                          to: @para,
                          c: map[LOOPHOLE[blk[i], CHAR]],
                          extra: paraLen];
                    ENDLOOP;
                blankCount ← 0;
                state ← entry;
                };
        };
    maxExceeded =>
        {
        FlushText[av, @para];
        --/* restore old line height */
        av.paraProps.basicProps.lineHeight ← oldHt;
        DocInterchangeDefs.AppendNewParagraph[
          to: [doc[av.doc]],
          paraProps: @av.paraProps,
          fontProps: @av.fontProps,
          nToAppend: 1];
        state ← entry;
        };
    endPara =>
        {
        s: LONG STRING ← av.src.text[paraEndsWith];

        IF s = NIL THEN
            {
            state ← entry;
            nextState ← entry;
            n ← n + 1;
            flushed ← FALSE;
            GOTO restart;
            };

        IF eop # 0 THEN
            {
            last ← c;
            c ← LOOPHOLE[blk[n], CHAR];
            };

        --/* if we are at the end of s, then match! */
        IF eop >= s.length THEN
            {
            IF NOT flushed THEN
                --/* flush all text */
                FlushText[av, @para];

            --/* restore old line height */
            av.paraProps.basicProps.lineHeight ← oldHt;
            DocInterchangeDefs.AppendNewParagraph[
              to: [doc[av.doc]],
              paraProps: @av.paraProps,
              fontProps: @av.fontProps,
              nToAppend: 1];

            IF flushed THEN
                --/* flush following text */
```

```
            FlushText[av, @para];

        eop ← 0;
        state ← entry;
        nextState ← entry;
        flushed ← FALSE;
        GOTO restart;
        };

    --/* c match with end-of-paragraph? */
    IF s[eop] = c THEN
        {
        eop ← eop + 1;
        n ← n + 1;
        }
    ELSE
        {
        --/* false alarm */
        IF ignore THEN
            --/* ouch, we interrupted ignoreTrailing */
            FOR j: CARDINAL IN [0..eop) DO
                IF s[j] = Ascii.CR THEN GOTO oneCR;
                IF s[j] # Ascii.SP OR s[j] # Ascii.TAB THEN
                  GOTO notWhite;
            REPEAT
                oneCR =>
                    {
                    --/* replace CR with one blank */
                    XString.AppendChar[
                      to: @para,
                      c: map[Ascii.SP],
                      extra: paraLen];
                    --/* other blanks ignored */
                    blankCount ← 0;
                    };
                notWhite =>
                    {
                    --/* flush blankCount characters */
                    IF lastBlkCount # blkCount THEN
                        {
                        --/* blanks straddle blocks */
                        THROUGH [1..blankCount] DO
                            XString.AppendChar[
                              to: @para,
                              c: map[Ascii.SP],
                              extra: paraLen];
                        ENDLOOP;
                        }
                    ELSE
                        FOR i: CARDINAL IN [blanksStart..blanksStart+blankCount) DO
                            XString.AppendChar[
                              to: @para,
                              c: map[LOOPHOLE[blk[i], CHAR]],
                              extra: paraLen];
                        ENDLOOP;
                    blankCount ← 0;
                    };
                FINISHED =>
                    {
                    --/* include current chars in blankCount */
                    blankCount ← blankCount + (MAX[eop,1] - 1);
                    state ← ignoreTrailing;
                    };
            ENDLOOP;

        --/* set up for next state */
        IF (c = Ascii.SP OR c = Ascii.TAB)
          AND ignore
          AND state # ignoreTrailing THEN
            {
            state ← ignoreTrailing;
            blanksStart ← n;
            blankCount ← 0;
            lastBlkCount ← blkCount;
            }
        ELSE
            {
            state ← append;
            n ← n + 1;
            --/* account for any CRs */
            --/* IF last = CR, then kludge handled it */
            IF last # Ascii.CR THEN
                FOR j: CARDINAL IN [0..eop) DO
                    IF s[j] = Ascii.CR THEN GOTO foundCR;
                REPEAT
                    foundCR =>
                        {
                        --/* replace one or more CRs with one blank */
                        XString.AppendChar[
                          to: @para,
                          c: map[Ascii.SP],
                          extra: paraLen];
                        };
                    FINISHED => NULL;
                ENDLOOP;
            };
        eop ← 0;
```

```
                    nextState ← entry;
                    flushed ← FALSE;
                    GOTO restart;
                    --/* end of false alarm */
                    };

                --/* continue looking for eop */
                IF c = Ascii.CR THEN
                    {
                    --/* flush preceding text, clear buffer */
                    FlushText[av, @para];
                    flushed ← TRUE;
                    };

                --/* translate character */
                state ← append;
                nextState ← endPara;

                --/* special look-ahead kludge to make naked CR's work */
                IF c = Ascii.CR
                  AND NOT ignore
                  AND eop < s.length
                  AND n < av.blk.stopIndexPlusOne
                  AND s[eop] # LOOPHOLE[blk[n], CHAR] THEN
                    {
                    --/* smart white space */
                    SELECT last FROM
                        Ascii.SP,
                        Ascii.TAB,
                        Ascii.CR,
                        Ascii.LF,
                        aHyphen => NULL; --/* just drop CR */
                        ENDCASE =>
                            {
                            XString.AppendChar[
                              to: @para,
                              c: map[Ascii.SP],
                              extra: paraLen];
                            };
                    };
                EXITS restart => NULL;
                };
            ENDCASE;
    ENDLOOP;

    --/* clean up */
    XString.FreeWriterBytes[@para];
};


UserAbortsPaginate: DocInterchangeDefs.CheckAbortProc = {
<< = PROCEDURE [clientData: LONG POINTER] RETURNS [abort: BOOL];
>>
    data: AVData = clientData;

    abort ← CheckAbort[data.background];
};


ZzInit: PROC = {
    pz: UNCOUNTED ZONE = BWSZone.Permanent[];

    --/* these Spaces should not be unmapped while this application is loaded */
    g ← [
      pcmap: Space.ScratchMap[(words + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      isomap: Space.ScratchMap[(words + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      pz: pz];

    --/* initialize pc ascii map */
    g.pcmap↑ ← [
    --  NS chset        code       PC hex: description
        256*    0B+     40B,     -- 00: Null space
        256*    357B+   337B,    -- 01: Smile face="have a nice day"
        256*    356B+   337B,    -- 02: Dark (357B | 337B) smile face
        256*    356B+   314B,    -- 03: Solid (357B | 314B) heart
        256*    356B+   315B,    -- 04: Solid (357B | 315B) diamond
        256*    357B+   316B,    -- 05: Clubs
        256*    357B+   313B,    -- 06: Spades
        256*    357B+   146B,    -- 07: Centered bullet
        256*    356B+   101B,    -- 08: Inverse (357B | 146B) centered bullet
        256*    0B+     11B,     -- 09: Codes0[tab]
        256*    0B+     15B,     -- 0A: Ascii.LF=Codes0[newline]
        256*    41B+    151B,    -- 0B: Male=Mars
        256*    41B+    152B,    -- 0C: Female=Venus
        256*    0B+     15B,     -- 0D: Ascii.CR=Codes0[newline]
        256*    356B+   325B,    -- 0E: Double sixteenth note
        256*    356B+   317B,    -- 0F: Compass symbol, NOT sun (357B | 347B)
        256*    356B+   277B,    -- 10: Forward indicator arrow
        256*    356B+   276B,    -- 11: Backward indicator arrow
        256*    356B+   265B,    -- 12: North-south arrow
        256*    356B+   172B,    -- 13: Double exclamation mark
        256*    0B+     266B,    -- 14: Paragraph sign=pilcrow
        256*    0B+     247B,    -- 15: Section sign
        256*    356B+   336B,    -- 16: Solid horizontal rectangle
        256*    356B+   266B,    -- 17: North-south arrow perpendicular
        256*    0B+     255B,    -- 18: North arrow
        256*    0B+     257B,    -- 19: South arrow
```

```
256*    0B+     256B,   -- 1A: East arrow
256*    0B+     254B,   -- 1B: West arrow
256*    356B+   335B,   -- 1C: Right angle symbol
256*    357B+   122B,   -- 1D: Double arrow
256*    42B+    46B,    -- 1E: Black point-up triangle
256*    42B+    47B,    -- 1F: Black point-down triangle
256*    0B+     40B,    -- 20: Space
256*    0B+     41B,    -- 21: Exclamation point
256*    0B+     42B,    -- 22: Neutral double quote
256*    0B+     43B,    -- 23: Number sign
256*    0B+     244B,   -- 24: Dollar sign
256*    0B+     45B,    -- 25: Percent sign
256*    0B+     46B,    -- 26: Ampersand
256*    0B+     47B,    -- 27: Apostrophe
256*    0B+     50B,    -- 28: Opening parenthesis
256*    0B+     51B,    -- 29: Closing parenthesis
256*    0B+     52B,    -- 2A: Asterisk
256*    0B+     53B,    -- 2B: Plus sign
256*    0B+     54B,    -- 2C: Comma
256*    0B+     55B,    -- 2D: Neutral dash; Also hyphen/minus
256*    0B+     56B,    -- 2E: Period=full stop
256*    0B+     57B,    -- 2F: Slash
256*    0B+     60B,    -- 30: Digit 0
256*    0B+     61B,    -- 31: Digit 1
256*    0B+     62B,    -- 32: Digit 2
256*    0B+     63B,    -- 33: Digit 3
256*    0B+     64B,    -- 34: Digit 4
256*    0B+     65B,    -- 35: Digit 5
256*    0B+     66B,    -- 36: Digit 6
256*    0B+     67B,    -- 37: Digit 7
256*    0B+     70B,    -- 38: Digit 8
256*    0B+     71B,    -- 39: Digit 9
256*    0B+     72B,    -- 3A: Colon
256*    0B+     73B,    -- 3B: Semicolon
256*    0B+     74B,    -- 3C: Less than
256*    0B+     75B,    -- 3D: Equals
256*    0B+     76B,    -- 3E: Greater than
256*    0B+     77B,    -- 3F: Question mark
256*    0B+     100B,   -- 40: Commercial at
256*    0B+     101B,   -- 41: Uppercase Latin letter A
256*    0B+     102B,   -- 42: Uppercase Latin letter B
256*    0B+     103B,   -- 43: Uppercase Latin letter C
256*    0B+     104B,   -- 44: Uppercase Latin letter D
256*    0B+     105B,   -- 45: Uppercase Latin letter E
256*    0B+     106B,   -- 46: Uppercase Latin letter F
256*    0B+     107B,   -- 47: Uppercase Latin letter G
256*    0B+     110B,   -- 48: Uppercase Latin letter H
256*    0B+     111B,   -- 49: Uppercase Latin letter I
256*    0B+     112B,   -- 4A: Uppercase Latin letter J
256*    0B+     113B,   -- 4B: Uppercase Latin letter K
256*    0B+     114B,   -- 4C: Uppercase Latin letter L
256*    0B+     115B,   -- 4D: Uppercase Latin letter M
256*    0B+     116B,   -- 4E: Uppercase Latin letter N
256*    0B+     117B,   -- 4F: Uppercase Latin letter O
256*    0B+     120B,   -- 50: Uppercase Latin letter P
256*    0B+     121B,   -- 51: Uppercase Latin letter Q
256*    0B+     122B,   -- 52: Uppercase Latin letter R
256*    0B+     123B,   -- 53: Uppercase Latin letter S
256*    0B+     124B,   -- 54: Uppercase Latin letter T
256*    0B+     125B,   -- 55: Uppercase Latin letter U
256*    0B+     126B,   -- 56: Uppercase Latin letter V
256*    0B+     127B,   -- 57: Uppercase Latin letter W
256*    0B+     130B,   -- 58: Uppercase Latin letter X
256*    0B+     131B,   -- 59: Uppercase Latin letter Y
256*    0B+     132B,   -- 5A: Uppercase Latin letter Z
256*    0B+     133B,   -- 5B: Opening bracket
256*    0B+     134B,   -- 5C: Reverse slant
256*    0B+     135B,   -- 5D: Closing bracket
256*    0B+     136B,   -- 5E: Circumflex accent (spacing character)
256*    0B+     137B,   -- 5F: Low bar (spacing character)
256*    0B+     140B,   -- 60: Grave accent (spacing character)
256*    0B+     141B,   -- 61: Lowercase Latin letter a
256*    0B+     142B,   -- 62: Lowercase Latin letter b
256*    0B+     143B,   -- 63: Lowercase Latin letter c
256*    0B+     144B,   -- 64: Lowercase Latin letter d
256*    0B+     145B,   -- 65: Lowercase Latin letter e
256*    0B+     146B,   -- 66: Lowercase Latin letter f
256*    0B+     147B,   -- 67: Lowercase Latin letter g
256*    0B+     150B,   -- 68: Lowercase Latin letter h
256*    0B+     151B,   -- 69: Lowercase Latin letter i
256*    0B+     152B,   -- 6A: Lowercase Latin letter j
256*    0B+     153B,   -- 6B: Lowercase Latin letter k
256*    0B+     154B,   -- 6C: Lowercase Latin letter l
256*    0B+     155B,   -- 6D: Lowercase Latin letter m
256*    0B+     156B,   -- 6E: Lowercase Latin letter n
256*    0B+     157B,   -- 6F: Lowercase Latin letter o
256*    0B+     160B,   -- 70: Lowercase Latin letter p
256*    0B+     161B,   -- 71: Lowercase Latin letter q
256*    0B+     162B,   -- 72: Lowercase Latin letter r
256*    0B+     163B,   -- 73: Lowercase Latin letter s
256*    0B+     164B,   -- 74: Lowercase Latin letter t
256*    0B+     165B,   -- 75: Lowercase Latin letter u
256*    0B+     166B,   -- 76: Lowercase Latin letter v
256*    0B+     167B,   -- 77: Lowercase Latin letter w
256*    0B+     170B,   -- 78: Lowercase Latin letter x
256*    0B+     171B,   -- 79: Lowercase Latin letter y
256*    0B+     172B,   -- 7A: Lowercase Latin letter z
```

```
    256*    0B+     173B,   -- 7B: Opening brace
    256*    0B+     174B,   -- 7C: Vertical bar
    256*    0B+     175B,   -- 7D: Closing brace
    256*    0B+     176B,   -- 7E: Tilde (spacing character)
    256*    356B+   72B,    -- 7F: Small house
    256*    361B+   55B,    -- 80: Cedilla C
    256*    361B+   345B,   -- 81: Diaeresis u
    256*    361B+   261B,   -- 82: Grave e
    256*    361B+   243B,   -- 83: Circumflex a
    256*    361B+   247B,   -- 84: Diaeresis a
    256*    361B+   241B,   -- 85: Grave a
    256*    361B+   250B,   -- 86: Ring a
    256*    361B+   255B,   -- 87: Cedilla c
    256*    361B+   262B,   -- 88: Circumflex e
    256*    361B+   265B,   -- 89: Diaeresis e
    256*    361B+   260B,   -- 8A: Grave e
    256*    361B+   304B,   -- 8B: Diaeresis i
    256*    361B+   300B,   -- 8C: Circumflex i
    256*    361B+   276B,   -- 8D: Grave i
    256*    361B+   47B,    -- 8E: Diaeresis A
    256*    361B+   50B,    -- 8F: Ring A
    256*    361B+   61B,    -- 90: Acute E
    256*    0B+     361B,   -- 91: Lowercase ae digraph
    256*    0B+     341B,   -- 92: Uppercase AE digraph
    256*    361B+   321B,   -- 93: Circumflex o
    256*    361B+   324B,   -- 94: Diaeresis o
    256*    361B+   317B,   -- 95: Grave o
    256*    361B+   341B,   -- 96: Circumflex u
    256*    361B+   337B,   -- 97: Grave u
    256*    361B+   355B,   -- 98: Diaeresis y
    256*    361B+   124B,   -- 99: Diaeresis O
    256*    361B+   146B,   -- 9A: Diaeresis U
    256*    0B+     242B,   -- 9B: Cent sign
    256*    0B+     243B,   -- 9C: Pound-Sterling sign
    256*    0B+     245B,   -- 9D: Yen
    256*    357B+   244B,   -- 9E: Pesetas
    256*    357B+   242B,   -- 9F: Florin
    256*    361B+   242B,   -- A0: Acute a
    256*    361B+   277B,   -- A1: Acute i
    256*    361B+   320B,   -- A2: Acute o
    256*    361B+   340B,   -- A3: Acute u
    256*    361B+   314B,   -- A4: Tilde n
    256*    361B+   114B,   -- A5: Tilde N
    256*    0B+     343B,   -- A6: Feminine Spanish ordinal indicator
    256*    0B+     353B,   -- A7: Masculine Spanish ordinal indicator
    256*    0B+     277B,   -- A8: Inverted question mark
    256*    356B+   152B,   -- A9: Start of line symbol
    256*    357B+   152B,   -- AA: Logical not=end of line symbol
    256*    0B+     275B,   -- AB: Fraction one half
    256*    0B+     274B,   -- AC: Fraction one quarter
    256*    0B+     241B,   -- AD: Inverted exclamation point
    256*    0B+     253B,   -- AE: Left double guillemet
    256*    0B+     273B,   -- AF: Right double guillemet
    256*    356B+   140B,   -- B0: Light shade
    256*    357B+   176B,   -- B1: Shade
    256*    356B+   141B,   -- B2: Dark shade
    256*    357B+   344B,   -- B3: Thin vertical line=Center box bar vertical
    256*    50B+    51B,    -- B4: y -x -y = Right middle box side
    256*    50B+    120B,   -- B5: Right box side double to single
    256*    50B+    121B,   -- B6: Right box side single to double
    256*    50B+    122B,   -- B7: Upper right box corner single to double
    256*    50B+    123B,   -- B8: Upper right box corner double to single
    256*    50B+    124B,   -- B9: Right box side double
    256*    50B+    125B,   -- BA: Center box bar vertical double
    256*    50B+    126B,   -- BB: Upper right box corner double
    256*    50B+    127B,   -- BC: Lower right box corner double
    256*    50B+    130B,   -- BD: Lower right box corner single to double
    256*    50B+    131B,   -- BE: Lower right box corner double to single
    256*    50B+    44B,    -- BF: -x -y = Upper right box corner
    256*    50B+    46B,    -- C0: y +x = Lower left box corner
    256*    50B+    52B,    -- C1: x +y -x = Middle box bottom
    256*    50B+    50B,    -- C2: -x -y +x = Middle box top
    256*    50B+    47B,    -- C3: y +x -y = Left middle box side
    256*    357B+   345B,   -- C4: Thin horizontal line=Center box bar horizontal
    256*    357B+   346B,   -- C5: Thin intersecting lines=Box intersection
    256*    50B+    132B,   -- C6: Left box side single to double
    256*    50B+    133B,   -- C7: Left box side double to single
    256*    50B+    134B,   -- C8: Lower left box corner double
    256*    50B+    135B,   -- C9: Upper left box corner double
    256*    50B+    136B,   -- CA: Middle box bottom double
    256*    50B+    137B,   -- CB: Middle box top double
    256*    50B+    140B,   -- CC: Left box side double
    256*    50B+    141B,   -- CD: Center box bar horizontal double
    256*    50B+    142B,   -- CE: Box intersection double
    256*    50B+    143B,   -- CF: Middle box bottom single to double
    256*    50B+    144B,   -- D0: Middle box bottom double to single
    256*    50B+    145B,   -- D1: Middle box top double to single
    256*    50B+    146B,   -- D2: Middle box top single to double
    256*    50B+    147B,   -- D3: Lower left box corner double to single
    256*    50B+    150B,   -- D4: Lower left box corner single to double
    256*    50B+    151B,   -- D5: Upper left box corner single to double
    256*    50B+    152B,   -- D6: Upper left box corner double to single
    256*    50B+    153B,   -- D7: Box intersection single to double
    256*    50B+    154B,   -- D8: Box intersection double to single
    256*    50B+    45B,    -- D9: -x -y = Lower right box corner
    256*    50B+    43B,    -- DA: x -y = Upper left box cornert
    256*    356B+   271B,   -- DB: Solid fill character
```

```
        256*    356B+   272B,   -- DC: Solid fill character, bottom half
        256*    356B+   274B,   -- DD: Solid fill character, left half
        256*    356B+   273B,   -- DE: Solid fill character, right half
        256*    356B+   275B,   -- DF: Solid fill character, top half
        256*    46B+    141B,   -- E0: Small letter Alpha
        256*    0B+     373B,   -- E1: Double s = sharp s
        256*    46B+    104B,   -- E2: Capital letter Gamma
        256*    46B+    163B,   -- E3: Small letter Pi
        256*    46B+    126B,   -- E4: Capital letter Sigma
        256*    46B+    166B,   -- E5: Small letter Sigma
        256*    46B+    157B,   -- E6: Small letter Mu
        256*    46B+    170B,   -- E7: Small letter Tau
        256*    46B+    132B,   -- E8: Capital letter Phi
        256*    46B+    113B,   -- E9: Capital letter Theta
        256*    46B+    135B,   -- EA: Capital letter Omega
        256*    46B+    145B,   -- EB: Small letter Delta
        256*    41B+    147B,   -- EC: Infinity
        256*    46B+    172B,   -- ED: Small letter Phi
        256*    46B+    140B,   -- EE: Small letter Epsilon
        256*    357B+   120B,   -- EF: Intersection
        256*    357B+   163B,   -- F0: Equal by definition
        256*    0B+     261B,   -- F1: Plus/minus sign
        256*    41B+    146B,   -- F2: Greater than or equal to
        256*    41B+    145B,   -- F3: Less than or equal to
        256*    356B+   355B,   -- F4: Top portion of integral sign
        256*    356B+   356B,   -- F5: Bottom portion of integral sign
        256*    0B+     270B,   -- F6: Divide sign
        256*    357B+   171B,   -- F7: Asymptotic to = approximately equal, type 2
        256*    0B+     260B,   -- F8: Degree sign
        256*    0B+     267B,   -- F9: Centered dot
        256*    0B+     307B,   -- FA: Over-dot accent
        256*    357B+   174B,   -- FB: Alternate rendition of "radical = root"
        256*    375B+   250B,   -- FC: Superscript n
        256*    0B+     262B,   -- FD: Superscript 2 as independent character from 2
        256*    356B+   52B,    -- FE: Histogram sign
        256*    357B+   41B     -- FF: Non-breaking space
    ];

    FOR c: CHARACTER IN CHARACTER DO
        g.isomap[c] ← XCharSet0.Make[LOOPHOLE[c]];
    ENDLOOP;

};

--/* main line code */

ZzInit[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:21:47 - Caro - Added error catcher in ConvertProc over CreateCommon,
                            Caught NSFile.Error in Logoff
29-Jun-87 13:13:00 - Caro - Added lineHtInPoints, AFTER setting
10-Jul-87 10:55:05 - Caro - Added aHyphen testing for smart spacing
19-Aug-87 11:01:32 - Caro - Fixed AR 13535 by updating oldInstance window
16-Sep-87 13:48:21 - Caro - isomap accentFirst from 241C to 301C
                            isomap lowGraphFirst from 0 to 241C
                            isomap lowGraphLast from 0 to 277C
                            pcmap accentLast from 257C to 245C
                            pcmap hiGraphFirst from 260C to 246C
12-Feb-88 12:58:57 - Shinsato - In AtoV, made sure eop # NIL before counting CR
                                in eop.
```

```
-- File: OfflineDiagInterface.mesa
-- Last edited on  20-Jun-85 10:12:24  by KL
--
-- Copyright (C) Xerox Corporation 1984, 1985. All rights reserved.
--
-- This interface module provides the common utilities and definitions used by
-- Dove offline diagnostic programmers.
--
DIRECTORY
  TTY   USING [Handle];

OfflineDiagInterface: DEFINITIONS =

BEGIN

--~~~~~~~~~~~~~~~~~~~~~~~~~~
-- PUBLIC TYPE definitions
--~~~~~~~~~~~~~~~~~~~~~~~~~~

--~~~~~~~~~~~~~~~~~~~~~~~~~~
--  AbortCurrentTest Signal
--~~~~~~~~~~~~~~~~~~~~~~~~~~

--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- A user can hit the Abort/Stop Key any time. This generates AbortCurrentTest.
-- The consequence of hitting the Abort/Stop Key, as far as the Control Module
-- is concerned, is one of the following:
--
--      1) If the user is running a test, the test is aborted. The menu is
--         redisplayed if the aborted test hogged the entire screen.
--      2) If the client is in the process of selecting a menu item, then the
--         parent menu will be re-entered after exiting the current menu.
--
-- If a client has the need to do some clean up work before exiting, then this
-- signal can be intercepted by the client. After performing the last minute
-- tasks, the client should REJECT this signal; unless, of course, the client
-- disallows aborting (ie. the test must run to a certain point in the program
-- before aborting is allowed). If the latter is the case, then the client can
-- simply ignore the signal or take whatever local action is appropriate; for a
-- client has complete control as to what to do with this signal
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  AbortCurrentTest: SIGNAL;  -- User has hit the STOP key.


--~~~~~~~~~~~~~~~~~~~
-- Procedural types
--~~~~~~~~~~~~~~~~~~~

-- Offine diagnostic procedures take the following form:

  OfflineDiagnosticProc: TYPE = PROCEDURE RETURNS [result: ResultType];


--~~~~~~~~~~~~~~~~~~~
-- Enumerated Types
--~~~~~~~~~~~~~~~~~~~

-- Every test returns a test result. The result is given to the user as follows:
--
--      none - Nothing (blanks)
--      passed - "P" printed next to the test selection number
--      failed - "F" printed next to the test selection number
--      ambiguous - "?" printed next to the test selection number

  ResultType: TYPE = { none, passed, failed, ambiguous };

--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- PUBLIC data structures and their allocation PROCEDUREs exported by the Control
-- Module: These data structures are allocated from a private heap which will be
-- deleted when one exits a test category.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- Space is premium. In order to minimize the amount of allocated memory, the
-- Control Module creates its own private heap. Common structures needed by the
-- diagnostic programmers are allocated from this private heap, which can be
-- readily destroyed or recreated by the Control Module. This mechanism gives
-- the Control Module the ability to contain dynamic memory usage and access
-- client structures and procedures.
--
-- The following are the structures and the PUBLIC procedures for allocating them
-- from the Control Module's private heap.

--~~~~~~~~~~
-- TestItem
--~~~~~~~~~~

-- A TestItem identifies a menu selection and its attributes. It can identify an
-- actual test or a submenu.

  TestItem: TYPE = RECORD [
            itemName: LONG STRING ← NIL,  -- Name displayed in the menu.
            test: OfflineDiagnosticProc,  -- The actual test procedure.
            -- When no helpExplanation is provided, the Control Module
            -- will supply a generic help message such as "Please enter 1 to 9".
            itemExplanation: LONG STRING ← NIL,
            -- A TestItem can select another menu of tests.
```

```
            subMenu: LONG POINTER TO AMenuOfSelections ← NIL ];

    -- The procedure to get an instance of the structure is:

    GetATestItem: PROCEDURE RETURNS [LONG POINTER TO TestItem];


--~~~~~~~~~~~~~~~~~~~~~~
-- TestItemsForThisNode
--~~~~~~~~~~~~~~~~~~~~~~

-- TestItemsForThisNode contain all the TestItems associated with this menu level.
-- GetATestItemsForThisMenu allocates as much space as needed to hold all the
-- TestItem that the various users can select at this node.

  TestItemsForThisNode: TYPE = RECORD [
    count:  CARDINAL,
    nodeItems: SEQUENCE numOfTestItemsInNode: CARDINAL OF LONG POINTER TO TestItem
    ];

  -- The procedure to get an instance of the structure is:

  GetATestItemsForThisNode: PROCEDURE [numberOfTestItemsInNode: CARDINAL ← 0]
                            RETURNS [LONG POINTER TO TestItemsForThisNode];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- SubMenus for different classes of users
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- Users are divided into 5 classes: Normal Users
--                                   System Administrators
--                                   Services Engineers
--                                   Manufacturing Personnel
--                                   Programmers
--
-- Not all the TestItems in a node are selectable by all the classes of users.
-- userSelections↑, adminSelections↑, seSelections↑, manufacturingSelections↑ and
-- programmerSelections↑ contain series of CARDINALs designating
-- the TestItems each class of users can select and in what order these
-- selections will appear when displayed for selection.
--
-- Each number corresponds to the index of an entry in TestItemsForThisNode, which
-- contains the pointers to the actual TestItems. The order in which the
-- TestItemsForThisMenu indices are entered is the order in which the selections
-- will be presented on the display. For example, if seSelections↑ has [9,7,1,8,5]
-- as its sequence, then the displayed selection menu will appear as follows:
--
--    Available Selections
--
--        1)  <TestItemsForThisNode[9]>
--        2)  <TestItemsForThisNode[7]>
--        3)  <TestItemsForThisNode[1]>
--        4)  <TestItemsForThisNode[8]>
--        5)  <TestItemsForThisNode[5]>
--
--    Please Enter Selection:
--
-- To save space, a common structure is used to contain the selections of a user
-- class to avoid the use of variant records. This structure is Selections. An
-- instance of this structure is needed for each user class for each menu.

  Selections: TYPE = RECORD -- MACHINE DEPENDENT RECORD
                 [count:  CARDINAL,  -- The number of entries in the SEQUENCE.
                  selections: SEQUENCE numberOfSelections: CARDINAL OF CARDINAL];

--~~~~~~~~~~~~~~~~~~~
-- User Selections
--~~~~~~~~~~~~~~~~~~~

-- GetASelectionArray allocates space for an array to hold all the indices of
-- pointers to TestItems stored in TestItemsForThisMenu. These indices designate
-- the tests that a class of users can run.

  GetASelectionArray: PROCEDURE [numberOfSelections: CARDINAL ← 0]
                      RETURNS [selectionsForClass: LONG POINTER TO Selections];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- All the information that describe the contents of a menu node are contained
-- in AMenuOfSelections, which is passed to the Control Module. Based on this,
-- the Control Module builds an appropriate menu for the user.
--
-- Run<mumble>Tests[pAMenuOfSelections] passes the pAMenuOfSelections for the
-- AMenuOfSelections describing a client's top node.
--
-- menuHelp is optional. The help text is displayed when a ? is entered alone.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  AMenuOfSelections: TYPE = RECORD [
        menuTitle: LONG STRING ← NIL,  -- Optional
        menuHelp: LONG POINTER TO HelpText ← NIL, -- Optional
        userSelections: LONG POINTER TO Selections,
        adminSelections: LONG POINTER TO Selections,
        seSelections: LONG POINTER TO Selections,
        manufacturingSelections: LONG POINTER TO Selections,
        programmerSelections: LONG POINTER TO Selections,
```

```
        testItemsForThisNode: LONG POINTER TO TestItemsForThisNode ];


    -- The procedure to get an instance of the structure is:

    GetAMenuOfSelections: PROCEDURE  RETURNS -- AMenuOfSelections defines a node.
      [pMenuOfSelections: LONG POINTER TO  AMenuOfSelections ← NIL];


  --~~~~~~~~~~
  -- HelpText
  --~~~~~~~~~~

  -- HelpText contains STRINGs of help text for interacting with users. A user can
  -- receive help by typing in a question mark to read this text.

    HelpText: TYPE = RECORD [
              count:  CARDINAL,
              helpTitle: LONG STRING ← NIL,
              textBody: SEQUENCE numberOfLines: CARDINAL OF LONG STRING];

    -- The procedure to get an instance of HelpText is

    GetAHelpText: PROCEDURE [numberOfLines: CARDINAL ← 0]
                    RETURNS [LONG POINTER TO HelpText];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- FixedPositionDisplayRecord
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Fixed-position display data are possible. The data to be dispalyed and the
  -- name associated with each datum are passed to the Control Module in the
  -- record FixedPositionDisplayRecord. The format of the display is as follows for
  -- a sample display with two-itemed rows:
  --
  --      < Title for the displayed record >
  --
  --        Item1Title: Item1Value      Item2Title: Item2Value
  --        Item3Title: Item3Value      Item4Title: Item4Value
  --              ...                         ...
  --
  --        ItemnTitle: ItemnValue      Item(n+1)Title: Item(n+1)Value
  --
  -- The fixed-position data are displayed in the Data Area by default, like all
  -- other data.
  --
  -- Each displayed item can optionally have its own name.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    FixedPositionDisplayRecord: TYPE = RECORD [
          displayTitle: LONG STRING ← NIL,  -- Heading for displayed data.
          numberOfRows: CARDINAL,  -- The number of rows of displayed data.
          rows: SEQUENCE rowCount: CARDINAL OF LONG POINTER TO ARow];


    -- The procedure to get an instance of FixedPositionDisplayRecord is:

    -- Client must specify the number of items on a line and the number of lines.
    -- The LONG POINTER returned by GetARow is entered into desired SEQUENCE slot.
    -- For example:  pFixedPositionDisplayRecord.rows[0] ← ptrToRow0
    --               pFixedPositionDisplayRecord.rows[1] ← ptrToRow1 ....
    --
    GetAFixedPositionDisplayRecord: PROCEDURE [rowCount: CARDINAL]
          RETURNS [LONG POINTER TO FixedPositionDisplayRecord];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- A line of display can contain several items. In deciding how many items to
  -- put on a line, keep in mind the width of the 15" display.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    ARow: TYPE = RECORD [
          itemsInARow:  CARDINAL ← 1,  -- Number of items per row.
          rowItems: SEQUENCE numberOfDisplayItemsInARow: CARDINAL OF DisplayItem];

    -- The procedure to get an instance of ARow is:

    GetARow: PROCEDURE [numberOfDisplayItemsInARow: CARDINAL ← 1]
                -- Store returned pointer in FixedPositionDisplayRecord.rows[n]
                RETURNS [LONG POINTER TO ARow];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- A displayed item is abstracted by DisplayItem. It allows a client to
  -- customize his/her display format.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    DisplayItem: TYPE = RECORD [
          -- Positional values must be non-zero.
          namePosition: CARDINAL ← 0,  -- Starting position... Must NOT be zero.
          name: LONG STRING ← NIL,  -- Optional name for this item.
          -- A value can be a STRING. If non-NIL, then it is printed after the name
          -- with a space separating the two.
          stringValue: LONG STRING ← NIL,
          valuePosition: CARDINAL ← 0,  -- Must be non-zero. 0 => no value
```

```
        value: LONG CARDINAL];  -- The value for the displayed item.

  --~~~~~~~~~~~~~~~~
  -- OptionsRecord
  --~~~~~~~~~~~~~~~~

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- There are occasions when it is desireable to have the user select an option
  -- from a list of many possible options. GetAnOption does this.
  --
  -- The Option Menu occupies the top portion of the display. While the Option
  -- Menu is displayed, the public Message and client Data areas retain their
  -- normal properties.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    OptionsRecord: TYPE = RECORD [
          optionMenuTiTle: LONG STRING ← NIL, -- Heading for displayed options.
          numberOfOptionLines: CARDINAL,  -- The number of lines of options.
          -- linesOfOptions holds the pointers returned by GetAnOptionLine
          linesOfOptions: SEQUENCE numberOfLines: CARDINAL
                          OF LONG POINTER TO AnOptionLine];

    -- The procedure to get an instance of an OptionsRecord is:

    GetAnOptionsRecord: PROCEDURE [numberOfLines: CARDINAL]
                          RETURNS [LONG POINTER TO OptionsRecord];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- A line of options may contain many items. Again , keep in mind the width of
  -- the 15" display.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    AnOptionLine: TYPE = -- MACHINE DEPENDENT -- RECORD
          [numberOfOptionsPerLine: CARDINAL ← 1,  -- The number of options per line.
           optionsOnALine: SEQUENCE optionsPerLine: CARDINAL OF OptionItem];

  -- The procedure to get an instance of an OptionLine is GetAnOptionLine.
  -- GetAnOptionLine allocates space for a line of options. The pointer to this
  -- line of displayed options should be assigned to the appropriate slot in the
  -- SEQUENCE of pointers in OptionsRecord.


    GetAnOptionLine: PROCEDURE [optionsPerLine: CARDINAL]
      -- Store the returned pointers in the sequence linesOfOptions in OptionsRecord
                        RETURNS [LONG POINTER TO AnOptionLine];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- An OptionItem is defined by OptionItem.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    OptionItem: TYPE = -- MACHINE DEPENDENT -- RECORD
          [position:  CARDINAL,  -- Starting position of this option.
           selectionNumberForThisItem: CARDINAL, -- Number for selecting this item.
           option: LONG STRING,  -- The name for this option.
           helpForThisOption: LONG STRING ← NIL];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  --  GetSpace Gives a block of contiguous memory of <pageCount> pages.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    GetSpace: PROCEDURE [pageCount: CARDINAL] RETURNS [LONG POINTER];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- PUBLIC PROCEDUREs exported by clients of the Offline Diagnostic Subsystem.
  -- Each client must export a single procedure to this interface in order to
  -- access its facilities.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    ClientPackage: TYPE = PROCEDURE RETURNS [LONG POINTER TO AMenuOfSelections];

    RunEthernetTests: ClientPackage;

    RunFloppyDiscTests: ClientPackage;

    RunFormatterScavengerBPU: ClientPackage;

    RunHardDiscTests: ClientPackage;

    RunKbdDsplMouseTests: ClientPackage;

    RunLaserDiscTests: ClientPackage;

    RunManufacturingTests: ClientPackage;

    RunPrinterTests: ClientPackage;

    RunRS232CTests: ClientPackage;

    RunTapeDriveTests: ClientPackage;

    -- System configuration utilities
    RunSystemConfigurator: ClientPackage;
```

```
  -- For miscellaneous uses
  RunMiscTests: ClientPackage;

  -- This is reserved for the self-test package that tests the various features
  -- of the Offline Diagnostic Subsystem.
  RunSelfTest: ClientPackage;


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- PUBLIC PROCEDUREs exported by the Control Module
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

<< Screen Lay Out:

  +=========================================================================+
  I   ID   Xerox (C) Xerox Corporation 1984, 1985. All rights reserved.      I
  I   Running: <Test Category>,  Test Selected: <Test name>                  I
  I                                                                          I
  I   O  Menu/System-Config/Options/TestParameter Area                       I
  I      GetAnOption, PutTestParameters(Fixed-Position Data)                 I
  I                                                                          I
  I   O  Interactive Selection Prompt Line - spX, spY, GetAnOption           I
  I   O  Auxiliary Prompt - auxPX, auxPY, GetANumber, GetYesNo, GetAString  I
  I                                                                          I
  I   O  Common Message Area - bOfMA, endOfMA,  msgX, msgY, dirtyML         I
  I      Help and error messages are displayed here (PutMessage)             I
  I                                                                          I
  I   O  Data Area - bOfDA, dX, dY, dirtyDL, endOfDA, dataOffSet             I
  I                  DisplayFixedPosition, PutData,                          I     I   Fixed-Position Data is displayed here.
      I
  I      This area is for client usage exclusively.                          I
  I                                                                          I
  +=========================================================================+

>>

--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- PutData allows a user to display data in the Data Area, which is under the
-- exclusive control of the client.
--
-- PutData displays the input string in the Data Region of the display. It is
-- meant to display dynamic test data. The defaults print data in a tightly
-- packed format.
--
-- numberAfterData is a LONG CARDINAL that will be appended immediately to the
-- data. If a separating space is desired, it is up to the client to add a space
-- as the last character in the immediately preceding data string.
--
-- If dataAreaHeading # NIL then the heading is printed and the rest of the data
-- area is cleared. Subsequent scrolling will not clear this heading until an
-- explicit "clearHeadingAndData: TRUE" is issued.
--
-- Once a heading is printed, subsequent invocation of PutData should have
-- dataAreaHeading set to NIL, until, of course, a new heading is desired. If not,
-- the data area will be cleared and the heading printed.
--
-- If fixed-position data is displayed, the entire Data Area is cleared when
-- the fixed-position data is cleared.
--
-- The Data Area is twice the size of the Message Area.
--
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  PutData: PROCEDURE [data: LONG STRING ← NIL, -- String to be printed.
                      -- numberAfterData = LAST[LONG CARDINAL] => no number to
                      -- be printed after the data STRING. Any other value will
                      -- be printed after data.
                      numberAfterData: LONG CARDINAL ← LAST[LONG CARDINAL],
                      dataAreaHeading: LONG STRING ← NIL, -- This is not cleared
                      -- clearHeadingAndData TRUE clears the entire Data Area
                      -- including dataAreaHeading. dataAreaHeading must be NIL.
                      clearHeadingAndData: BOOLEAN ← FALSE,
                      -- clearDataAreaOnly does not clear the heading. Only the
                      -- data below the heading is cleared.
                      clearDataAreaOnly: BOOLEAN ← FALSE,

                      -- Control variables for formatting dispaly data.
                      startWithNewLine: BOOLEAN ← FALSE, -- CRLF
                      numOfBlankLines: CARDINAL ← 0, -- CRLF numOfBlankLine times.
                      -- spaceBeforePrinting has effect only if startWithNewLine
                      -- and clearDataAreaOnly are both FALSE.
                      blankSpaces: CARDINAL ← 0,  -- Spaces before printing
                      -- xPosition = 0 => Let Control module and blankSpaces
                      -- specify the position on the line to print data.
                      -- Non-Zero xPosition overrides automatic positioning.
                      -- The Control Module will start printing at xPosition
                      -- after calculating the current line.
                      xPosition: CARDINAL ← 0, -- Position to start printing
                      -- Pause at the bottom of the Data Area so that the user
                      -- can review the data before they are cleared.
                      pauseAtBottomOfDataArea: BOOLEAN ← FALSE];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- PutMessage displays the input messages in the Message Region of the display.
-- This is a public interactive area: both the client and Control Module write
-- data into this region randomly.
```

```
--
-- The default format is a new line per message.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  PutMessage: PROCEDURE [message: LONG STRING ← NIL,
                         beep: BOOLEAN ← FALSE,
                         -- startWithNewLine TRUE invalidates spaceBeforePrinting
                         startWithNewLine: BOOLEAN ← TRUE,  -- CRLF
                         -- One can insert numOfBlankLine blank lines.
                         numOfBlankLines: CARDINAL ← 0,
                         blankSpaces: CARDINAL ← 1,
                         --clearMessageAreaFirst TRUE invalidates startWithNewLine
                         -- and spaceBeforePrinting
                         clearMessageAreaFirst: BOOLEAN ← FALSE];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- DisplayFixedPositionData is meant to give the client the ability to display
-- data related to topics whose headings are fixed positionally.
--
-- The fixed data are displayed at the beginning of the Data Area.
--
-- In addition to a heading associated with each value, a permanent title can
-- optionally be dispalyed at the very beginning of the Data Area. The title is
-- contained in the displayData.displayTitle field of displayData↑.
--
-- The first invocation should have upDateOnly set to FALSE (upDateOnly: FALSE).
-- Both the headings and values associated with each heading will be printed.
--
-- Subsequent invocations should only update the VALUEs. The heading then would
-- not be reprinted each time. This is achieved by setting "upDateOnly: TRUE",
-- which is the default.
--
-- clearDataArea clears the displayed data and re-initialize the Data Area,
-- regardless what upDateOnly is. It gives the client the means to clear the
-- Data Area without affecting the rest of the screen.
--
-- While fixed-position data is displayed, the remainder of the Data Area is
-- accessible to PutData and retains all its properties.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  DisplayFixedPositionData: PROCEDURE
        [displayData: LONG POINTER TO FixedPositionDisplayRecord ← NIL,
         clearDataArea: BOOLEAN ← FALSE, -- TRUE clears before printing.
         upDateOnly:  BOOLEAN ← TRUE];  -- TRUE => Print values only.


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- PutTestParameters displays and updates test parameters in the menu/option
-- area. Each datum has two fields, a title (STRING) and a value (LONG CARDINAL).
--
-- This procedures uses the same data structures as DisplayFixedPositionData.
--
-- upDateOnly should be set to FALSE for the first call. "upDateOnly: FALSE" will
-- cause the screen to be cleared and the title and name fields to be printed in
-- addition to the initial value fields. "upDateOnly: TRUE" prints only the new
-- values.
--
-- parameters = NIL will clear and reformat the screen, regardless what upDateOnly
-- is.
--
-- Both Data and Message areas retain their normal properties while parameters
-- are displayed.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  PutTestParameters: PROCEDURE [
        parameters: LONG POINTER TO FixedPositionDisplayRecord ← NIL,
        upDateOnly:  BOOLEAN ← TRUE];  -- TRUE => Print values only.


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- GetYesNo gets a yes/no response from the user. A YES returns TRUE. NO returns
-- FALSE.
-- Yes(Y/y) and No(N/n) are defined in the Mesasage Keys. They can, therefore,
-- take on any dissimilarly CHARACTER-valued pairs.
-- The default: value is returned if CR is the only input and defaultSpecified
-- is TRUE.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  GetYesNo: PROCEDURE [prompt: LONG STRING ← NIL,
                       help: LONG POINTER TO HelpText ← NIL,
                       defaultSpecified: BOOLEAN ← FALSE,
                       default: BOOLEAN ← FALSE]
              RETURNS [YesReturnsTrue: BOOLEAN];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- GetANumber gets a numeric input from the user.
-- + and - are also allowed. These, however, are directional indicators.
-- + sets foreward to TRUE, - sets foreward to FALSE.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  GetANumber: PROCEDURE [prompt: LONG STRING ← NIL,  -- Personalized prompt.
                         help: LONG POINTER TO HelpText ← NIL,  -- Explanation
                         lowLimit: LONG CARDINAL ← 1,
                         upperLimit: LONG CARDINAL ← LAST [LONG CARDINAL],
                         numberIsHexadecimal: BOOLEAN ← FALSE,
```

```
                    numberIsLong: BOOLEAN ← FALSE,
                    defaultNumber: LONG CARDINAL ← LAST[LONG CARDINAL]]
                RETURNS [longNumber: LONG CARDINAL,  -- 0 if number is not long
                    -- number is 0 if numberIsLong is TRUE.
                    number: CARDINAL,
                    -- Directional default is FOREWARD.
                    foreward: BOOLEAN ← TRUE, -- FALSE => reverse
                    numberInStringFormat: LONG STRING ]; -- In specified base


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- GetAnOption:
--
-- There are occasions when it is desireable to have the user select an option
-- from a list of many possible options. GetAnOption does this.
--
-- An invocation with "optionTable: NIL" will get another option input from
-- the user for the client from the currently displayed option list.
--
-- optionTable # NIL will cause the option table to be printed.
--
-- The client can specify a default option returned at the receipt of a CR.
--
-- The client can loop and get as many option selections as the client wishes.
-- To prevent the reprinting of the option table, set optionTable to NIL after
-- the first use of GetAnOption. The client determines when to terminate. This
-- can be done be having one of the options be:  n) Terminate option input.
--
-- The client can insert a customized help text for each use of GetAnOption.
-- This also applies to the prompt. By knowing what has been entered, the user
-- can be prevented from erroneous operations by inserting different prompts.
--
-- 0 can not be used by the client as a selection number.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  GetAnOption: PROCEDURE [optionTable: LONG POINTER TO OptionsRecord ← NIL,
                          defaultOption: CARDINAL ← 0, -- When CR = the only input
                          optionPrompt: LONG STRING ← NIL,  -- Prompt for input.
                          optionHelp: LONG POINTER TO HelpText ← NIL,
                          justDisplayTable: BOOLEAN ← FALSE]
                 RETURNS [selectedOption: CARDINAL];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- GetAString is for getting a string input from the user. This procedure should
-- not be used as a means to get a command from the user. All commands should be
-- hidden behind menu/option selections. The idea is:
--
--       What you see is what you get... And if you don't see it, entering a
--       question mark should give you what you need.
--
-- This procedure should be used to get a name ( ie. Host or whatever )
-- from the user, not for getting a command.
--
--      prompt: a client-customizeable message given to the user when getting
--              an input from the user. If none is supplied, the following is
--              the default message:  "Please enter a name"
--
--      defaultString: an optional prompt appended to the above prompt as
--                     follows: [<defaultString>]
--
--      help: a block of text explaining the current action. If help = NIL,
--            "Please enter a name" is given as the default
--
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  GetAString:  PROCEDURE [prompt: LONG STRING ← NIL, -- Personalized prompt
                          defaultString: LONG STRING ← NIL,
                          help: LONG POINTER TO HelpText ← NIL, -- Help
                          echoWithStar: BOOLEAN ← FALSE]
                 RETURNS [LONG STRING];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- HitAnyKeyToContinue temporarily pauses the test until the user enters a key.
-- If the key is the STOP key, the test is terminated; all other keys cause the
-- test to continue at the point of pause.
--
-- If prompt is NIL, the default prompt of "Enter any key to continue:' is
-- displayed.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  HitAnyKeyToContinue: PROCEDURE [
     prompt: LONG STRING ← NIL, beep: BOOLEAN ← TRUE];


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- LookForAbort checks to see if the user has hit the "STOP" key. If the key is
-- hit, AbortCurrentTest is raised. This signal can be used internally by the
-- client by catching it; or, if eventually REJECTed by the client, passed back
-- to the Control Module as a genuine request to abort the current test.
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  LookForAbort: PROCEDURE;


--~~~~~~~~~~~~~~~~~~~~~~~~
```

```
  -- Miscellaneous exports
  --~~~~~~~~~~~~~~~~~~~~~~~~

    LoginType: TYPE = {
               NormalUser, Administrator, Services, Manufacturing, Programmer };

    userType: READONLY LoginType;  -- This is the class of the logged-in user.

    bOfDA: READONLY CARDINAL;  -- Beginning of Data Area of display.

    inputChar: READONLY CHAR;  -- Current user input

    inputSensed: BOOLEAN;  -- TRUE => user has just entered an input.

    abortSensed: BOOLEAN;  -- TRUE => user has hit the  STOP key.


  --~~~~~~~~~~~~~~~~~~~~~
  -- Handle to DiagTTY
  --~~~~~~~~~~~~~~~~~~~~~

    hDiagTTY: TTY.Handle;


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- DiagHeap - All space allocation should be from this Heap. This makes it
  -- possible for the Control Module to delete spaces when it knows they are no
  -- longer needed.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    DiagHeap: READONLY UNCOUNTED ZONE;


  <<
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Informational SIGNALS that will transfer control to the Debugger. These serve
  -- mainly to assist the client in interfacing with the Control Module.
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    NoTestsInUserSelections => No selections for Normal User.
    NoTestInAdminSelections => No selections for System Administrator
    NoTestInSESelections => No selections for Service Engineer .
    NoTestInManufacturingSelections => No selections for Manufacturing.
    NoTestInProgrammerSelections => No selections for Programmer.
    TestItemsForThisNodeIsEmpty => Node is void.
    LineIsTooLong => Line is too long. Plan for 15" display.
    MenuIsTooLargeForScreen => Menu is too large for display. Plan for 15" display.

  >>

    END.  -- OfflineDiagInterface


  LOG:

  Created: 10-Jul-84 by KL.
  19-Jun-85 by KL - Added defaults to GetYesNo
  20-Jun-85 by KL - Replaced Analyst with Manufacturing as a valid LoginType.
```

```
-- File: OfflineDiagInterfaceExtra.mesa
-- last edit on   26-Oct-87 10:06:27  by JMA
--
-- Copyright (C) 1985, 1986, 1987 by Xerox Corporation. All rights reserved.
--
-- This interface module is meant to contain all the miscellaneous pieces needed
-- by the offline diagnostic subsystem. Also, new clients can be added if
-- recompilation of clients of this module is not difficult.

DIRECTORY
  OfflineDiagInterface USING [ClientPackage];

OfflineDiagInterfaceExtra: DEFINITIONS =
  BEGIN

  -- This is exported by OFLFloppyExecImpl. Its purpose is to read the bcd files
  -- on a floppy disk used to build a multiple client offline diagnsotic system.
  --
  ReadDiagnosticsPackage: PROCEDURE;

  -- This cleans up the undesireable bcd's loaded from the second floppy disk.
  -- This is exported by OFLFloppyExecImpl.
  --
  EraseLoadedConfig: PROCEDURE;

  -- This is exported by OfflineDiagnosticVersionImpl, which exports a version
  -- number. This version number is to be bound with every config file that
  -- will be used to build an offline diagnsotic boot file
  --
  GetVersion: PROCEDURE RETURNS [version: LONG STRING];

  -- ValidateClients is exported by OfflineDiagnosticControlModuleB for the use of
  -- OFLFloppyExecImpl. This should only be used with "checkOnly" defaulted to
  -- TRUE. If false, an internal client is built. This may cause problems to the
  -- OfflineDiagnosticControlModule.
  --
  ValidateClients: PROCEDURE[checkOnly: BOOLEAN ← TRUE] RETURNS[valid: BOOLEAN];

  -- This package contains the floppy utilities.
  --
  RunFloppyDiscUtilities: OfflineDiagInterface.ClientPackage;

  -- This package contains the Tempest tests
  --
  RunSecureDeviceTests: OfflineDiagInterface.ClientPackage;

  -- This package contains the PCE diagnostics
  --
  RunPCETests: OfflineDiagInterface.ClientPackage;

  -- This package contains the laser printer tests
  --
  RunLaserPrinterTests: OfflineDiagInterface.ClientPackage;

  -- This package contains the Voice Option diagnostics
  --
  RunVoiceOptionTests: OfflineDiagInterface.ClientPackage;

  -- This package contains the Scanner diagnostics
  --
  RunScannerTests: OfflineDiagInterface.ClientPackage;

  -- This package contains the Cartridge Tape diagnostics
  --
  RunCartridgeTapeTests: OfflineDiagInterface.ClientPackage;


  END... OfflineDiagInterfaceExtra

LOG

Created: 19-Dec-85 10:06:37 by MXT and KL to hold miscellaneous components and
         new clients
8-Jan-86: Added the following clients:

          RunSecureDeviceTests - a package of Tempest tests
          RunPCETests - a package of PCE diagnostics
          RunLaserPrinterTests - a package of laser printer diagnsotics
          RunVoiceOptionTests - a package of diagnostics for the Voice box

21-Jan-87 Added scanner diagnostics (JMA)

26-Oct-87 Added cartridge tape diagnostics (JMA)
```

```
-- File: OFLFloppyExecImpl.mesa - last edit:
-- STC   1-Feb-88 13:25:58

-- Copyright (C) 1985, 1986, 1987, 1988 by Xerox Corporation. All rights reserved.


DIRECTORY
  BackingStore USING [Run],
  CMDiagMsgKeysDove USING[cmMsgKeys, DiagMessages],
  File USING [Type],
  FileTypes USING [tUntypedFile],
  Floppy USING [
    Close, Error, ErrorType, FileHandle, GetFileAttributes, GetNextFile,
    nullFileID, nullVolumeHandle, Open, Read, VolumeHandle],
  Environment USING [ bitsPerWord,  PageNumber],
  OfflineDiagInterface,
  OfflineDiagInterfaceExtra,
  Process USING [Pause, SecondsToTicks],
  Runtime USING [
    ConfigError, ControlLink,
    UnNewConfig, VersionMismatch],  --GetBuildTime, ],
  Space USING [
    Deallocate, Interval, LongPointerFromPage, nullInterval, PageFromLongPointer,
    virtualMemory],
  SpecialLoader USING [LoadConfig, MapProcType, UnmapProcType],
  SpecialSpace USING [AllocateForCode],
  VM USING [GetMapUnitAttributes, Interval, ScratchMap, Unmap];

OFLFloppyExecImpl: PROGRAM
  IMPORTS
    CMDiagMsgKeysDove, Floppy, OfflineDiagInterface, OfflineDiagInterfaceExtra,
    Process, Runtime, Space, SpecialLoader,
    SpecialSpace, VM
  EXPORTS OfflineDiagInterfaceExtra =
  BEGIN
  OPEN CMDiagMsgKeysDove, ODI: OfflineDiagInterface,
    OdiExtra: OfflineDiagInterfaceExtra;

  TryAgain: SIGNAL = CODE;
  GiveUp: SIGNAL = CODE;
  NoFile: SIGNAL = CODE;

  prevConfig: PROGRAM ← NIL;
  nullFile: Floppy.FileHandle = [Floppy.nullVolumeHandle, Floppy.nullFileID];

  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  -- PUBLIC PROCs for OdiExtra.
  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  EraseLoadedConfig: PUBLIC PROCEDURE = {
    IF prevConfig # NIL THEN {
      ODI.PutMessage[
        message: cmMsgKeys[unloading], clearMessageAreaFirst: TRUE];
      Runtime.UnNewConfig[LOOPHOLE[prevConfig]];
      ODI.PutMessage[message: cmMsgKeys[finished], startWithNewLine: FALSE]}
    ELSE {
      ODI.PutMessage[message: cmMsgKeys[cantErase], clearMessageAreaFirst: TRUE, beep: TRUE];
      BlackHole[quit]};
    };

  ReadDiagnosticsPackage: PUBLIC PROCEDURE   = {
    IF OdiExtra.ValidateClients[TRUE] THEN RETURN;
    -- At least one Diagnostic client already exists.

    ReadFloppy[];
    };

  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  -- Utility-Type Functions
  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  Confirm: PROC [] = {
    ODI.PutMessage[message: cmMsgKeys[insertDiskLabeled], numOfBlankLines: 1];
    ODI.PutMessage[message: cmMsgKeys[diskForWSDiag], numOfBlankLines: 1];
    ODI.PutMessage[message: cmMsgKeys[diskForScannerDiag], numOfBlankLines: 1];
    ODI.PutMessage[message: cmMsgKeys[diskForCartTapeDiag], numOfBlankLines: 1];
    UNTIL ODI.GetYesNo[prompt: cmMsgKeys[isDiskReady], defaultSpecified: TRUE, default: TRUE] DO ENDLOOP; };

  PutNoFileMsgs: PROC = {
    ODI.PutMessage[
      message: cmMsgKeys[noDiagPkg], beep: TRUE,
      clearMessageAreaFirst: TRUE];
    ODI.PutMessage[cmMsgKeys[checkDisk]];
    };

  GetFloppyBcds: PROC RETURNS[link: PROGRAM, count: CARDINAL ← 0, validVersion: BOOLEAN ← TRUE] =
    BEGIN
    file: Floppy.FileHandle;
    volume: Floppy.VolumeHandle ← Floppy.nullVolumeHandle;

    volume ← Floppy.Open[
      drive: 0 !
      Floppy.Error =>
        SELECT error FROM
          notReady => {
            ODI.PutMessage[cmMsgKeys[notReady]]; TryAgain[]};
          noSuchDrive => {ODI.PutMessage[cmMsgKeys[noSuchDrive]]; TryAgain[]};
          invalidFormat => {
            ODI.PutMessage[cmMsgKeys[invalidFormat]]; TryAgain[]};
```

```
        needsScavenging => {
          ODI.PutMessage[cmMsgKeys[needsScavenging]]; TryAgain[]};
        ENDCASE; ];
    ODI.PutMessage[
      message: cmMsgKeys[loading],
      clearMessageAreaFirst: TRUE];
    file ← [volume, Floppy.nullFileID];
    count ← 0;
    UNTIL (file ← GetNextFile[file]) = nullFile DO
      ENABLE ANY => {
        IF volume # Floppy.nullVolumeHandle THEN Floppy.Close[volume];
        volume ← Floppy.nullVolumeHandle;
        };
      [link, validVersion] ← GetFloppyBcdsInternal[file];
      count ← count + 1;
      IF NOT validVersion THEN EXIT; -- the file just loaded contains invalid version.
      ENDLOOP;
    Floppy.Close[volume];
    volume ← Floppy.nullVolumeHandle;
    END;


  GetFloppyBcdsInternal: PROC[file: Floppy.FileHandle] RETURNS [cLink: PROGRAM, valid: BOOLEAN] =
    BEGIN

    MapSpace: SpecialLoader.MapProcType =
      -- pageOffset, pageCount: CARDINAL,
      -- swapUnits: Space.SwapUnits, access: Space.Access]
      --RETURNS [mapUnitPtr: LONG POINTER]
      BEGIN
      offset: CARDINAL ← 1;
      interval: Space.Interval ← SpecialSpace.AllocateForCode[
        pageCount, Space.virtualMemory];
      -- read floppy file directly into resident VM
      VM.ScratchMap[
        interval: [Space.PageFromLongPointer[interval.pointer], interval.count] !
        UNWIND => Space.Deallocate[interval]];
      Floppy.Read[
        file: file, first: offset + pageOffset, count: pageCount,
        vm: interval.pointer !
        Floppy.Error =>
          SELECT error FROM
            fileNotFound => {
              ODI.PutMessage[cmMsgKeys[fileNotFound]]; GiveUp[]};
              -- The specified file was not found on floppy disk.
            hardwareError => {
              ODI.PutMessage[cmMsgKeys[hardwareError]]; GiveUp[]};
            ENDCASE => SIGNAL NoFile];
              -- There happened some problem accessing the floppy disk but
              -- allow user to replace and try again.
      RETURN[interval.pointer];
      END;

    UnmapSpace: SpecialLoader.UnmapProcType =
      -- targetAddress: LONG POINTER]
      -- RETURNS [mapUnit: Space.Interval]
      BEGIN
      mapUnit ← Space.nullInterval;
      IF targetAddress # NIL THEN {
        run: ARRAY [0..1) OF BackingStore.Run;
        page: Environment.PageNumber = Space.PageFromLongPointer[targetAddress];
        vmInterval: VM.Interval ← VM.GetMapUnitAttributes[
          page, DESCRIPTOR[run]].mapUnit;
        -- UnmapAt/Deallocate (vs. Unmap) done to get mapUnit.count
        mapUnit ← [Space.LongPointerFromPage[vmInterval.page], vmInterval.count];
        VM.Unmap[vmInterval.page];
        Space.Deallocate[mapUnit]};
      END;

    valid ← TRUE;
    cLink ← SpecialLoader.LoadConfig[MapSpace, UnmapSpace, TRUE !
      Runtime.VersionMismatch => {
        ODI.PutMessage[message: cmMsgKeys[cantFindConfig], clearMessageAreaFirst: TRUE];
        valid ← FALSE;
        RESUME}];
    END;  -- Of GetFloppyBcds

  GetNextFile: PROC [file: Floppy.FileHandle]
    RETURNS [Floppy.FileHandle] = {
    type: File.Type ← FileTypes.tUntypedFile;

    WHILE type = FileTypes.tUntypedFile DO
      file ← Floppy.GetNextFile[file];
      IF file.file = Floppy.nullFileID THEN RETURN[nullFile];
      [, type] ← Floppy.GetFileAttributes[file];
      ENDLOOP;
    RETURN[file]};

  CheckModuleExistence: PROC RETURNS [isBound: BOOLEAN ← FALSE] =
    BEGIN
    isBound ← Runtime.IsBound[LOOPHOLE[ODI.RunEthernetTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunFloppyDiscTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunHardDiscTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunRS232CTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunKbdDsplMouseTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunPrinterTests, Runtime.ControlLink]]
      OR Runtime.IsBound[
```

```
        LOOPHOLE[ODI.RunFormatterScavengerBPU, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunSystemConfigurator, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunSelfTest, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunLaserDiscTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunManufacturingTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunTapeDriveTests, Runtime.ControlLink]]
      OR Runtime.IsBound[LOOPHOLE[ODI.RunMiscTests, Runtime.ControlLink]];
    END;

  RunFloppyExec: PUBLIC ODI.ClientPackage = -- for Diagnotics menu.
    BEGIN
    menuPtr: LONG POINTER TO ODI.AMenuOfSelections ← NIL;
    menuEntriesPtr: LONG POINTER TO ODI.TestItemsForThisNode;
    menuEntryPtr: LONG POINTER TO ODI.TestItem;
    userTestPtr: LONG POINTER TO ODI.Selections;
    adminTestPtr: LONG POINTER TO ODI.Selections;
    serviceTestPtr: LONG POINTER TO ODI.Selections;
    manufacturingTestPtr: LONG POINTER TO ODI.Selections;
    programerTestPtr: LONG POINTER TO ODI.Selections;
    index: CARDINAL ← 0;

    menuEntriesPtr ← ODI.GetATestItemsForThisNode[2];

    menuEntryPtr ← ODI.GetATestItem[];
    menuEntryPtr.itemName ← cmMsgKeys[menuName];
    menuEntryPtr.test ← ReadDiagnosticsPackage;
    menuEntriesPtr.nodeItems[0] ← menuEntryPtr;

    menuEntryPtr ← ODI.GetATestItem[];
    menuEntryPtr.itemName ← "Erase Loaded Config";
    menuEntryPtr.test ← EraseLoadedConfig;
    menuEntriesPtr.nodeItems[1] ← menuEntryPtr;

    userTestPtr ← ODI.GetASelectionArray[1];
    adminTestPtr ← ODI.GetASelectionArray[1];
    serviceTestPtr ← ODI.GetASelectionArray[1];
    manufacturingTestPtr ← ODI.GetASelectionArray[1];
    programerTestPtr ← ODI.GetASelectionArray[2];

    userTestPtr.selections[0] ← 0;
    adminTestPtr.selections[0] ← 0;
    serviceTestPtr.selections[0] ← 0;
    manufacturingTestPtr.selections[0] ← 0;
    programerTestPtr.selections[0] ← 0;
    programerTestPtr.selections[1] ← 1;

    menuPtr ← ODI.GetAMenuOfSelections[];
    menuPtr.menuTitle ← cmMsgKeys[floppyExec];
    menuPtr.userSelections ← userTestPtr;
    menuPtr.adminSelections ← adminTestPtr;
    menuPtr.seSelections ← serviceTestPtr;
    menuPtr.manufacturingSelections ← manufacturingTestPtr;
    menuPtr.programmerSelections ← programerTestPtr;
    menuPtr.testItemsForThisNode ← menuEntriesPtr;
    RETURN[menuPtr];

    END;

  ReadFloppy: PROC =
    BEGIN
    ENABLE
      BEGIN
      TryAgain => {
        Process.Pause[Process.SecondsToTicks[3]];
        ODI.PutMessage[cmMsgKeys[tryAgain]];
        Confirm[ ! TryAgain => GOTO iQuit];
        CONTINUE};
      GiveUp => GOTO iQuit;
      ABORTED => GOTO aborted;
      END;

    count: CARDINAL ← 0;
    link: PROGRAM;
    found: BOOLEAN ← FALSE;
    versionIsValid: BOOLEAN ← FALSE;

  -- Intended to get the number of floppy disk to be read.
  -- Currently the number is 1 and I commented out this.
    --[number: count] ← ODI.GetANumber[
    --  prompt: "How many floppys to insert?"L,
    --  lowLimit: 0, upperLimit: 10, defaultNumber: 1];

  UNTIL found DO
    -- FOR i:CARDINAL IN [1..count] DO
    -- ODI.PutMessage[cmMsgKeys[insertDisk]];
    Confirm[];
    versionIsValid ← FALSE;
    [link, count, versionIsValid] ← GetFloppyBcds[
      !
      Runtime.ConfigError => { -- will be raised if the content is not the executable BCD.
        PutNoFileMsgs[];
        ODI.HitAnyKeyToContinue[];
        LOOP};
      NoFile => { -- Will be raised if no file with valid file type is on the floppy disk.
        PutNoFileMsgs[];
        LOOP}];
    --ENDLOOP; <-- -- of FOR i: CARDINAL IN [1..count] DO
```

```
    prevConfig ← link;


    IF count = 0 THEN { -- no file was found of floppy disk.
      PutNoFileMsgs[];
      LOOP}
    ELSE { -- count # 0
      -- Some files were loaded.
      -- first check the version is valid.
      IF NOT versionIsValid THEN -- loaded bcd is version mismatched.
        IF link = NIL THEN {ODI.PutMessage[message: cmMsgKeys[cantErase], beep: TRUE];
          ERROR GiveUp}
        ELSE {-- try to unload the loaded bcd.
          Runtime.UnNewConfig[LOOPHOLE[link]];
          ODI.PutMessage[cmMsgKeys[checkDisk]];
          ODI.HitAnyKeyToContinue[];
          LOOP}; -- of versionIsValid.

      -- Check loaded config instatnce.
      found ← OdiExtra.ValidateClients[TRUE]; --CheckModuleExistence[];
      IF NOT found THEN {
        -- Loaded BCD doesn't have OfflineDiagnostics package.
        PutNoFileMsgs[];
        IF link = NIL THEN {
        -- Unable to clean up the resident memory so the system must be rebooted.
          ODI.PutMessage[message: cmMsgKeys[cantErase], beep: TRUE];
          ERROR GiveUp}; -- of link = NIL
        Runtime.UnNewConfig[LOOPHOLE[link]];
        ODI.HitAnyKeyToContinue[];
        LOOP} -- of NOT found
      ELSE {-- at least one offline diagnostic package was loaded.
        ODI.PutMessage[
          message: cmMsgKeys[finishLoading],
          clearMessageAreaFirst: TRUE];
          EXIT
        }
      }; -- of count # 0
    ENDLOOP;

    EXITS
      iQuit => {BlackHole[quit]};
      aborted => {BlackHole[aborted]};
    END;

  BlackHole: PROCEDURE[msg: CMDiagMsgKeysDove.DiagMessages] = {
    ODI.PutMessage[cmMsgKeys[msg]]; DO ENDLOOP};

  END..

LOG
25-Aug-85 22:44:33       ET
   Created from Othello>VolumeInitCommandImpl.mesa
10-Sep-85 14:54:12       MXT
   Changed for OfflineDiagKernelDove
 5-Nov-85 15:32:20       MXT
   Added read multiplu files from one floppy disk function.

31-Jan-86 16:27:55       MXT     (Reason Not Given)
14-Jan-86 17:48:32       MXT     Changed to use OdiExtra.ValidateClients.
  5-Mar-87 11:39:37      KXW     Modified Confirm to post messages for scanner diagnostics.
27-Jan-88  9:54:26, STC, add diskForCartTapeDiag in Confirm
```

```
-- File: OfflineDiagnosticVersionImpl.mesa
-- Last edited by STC, 21-Jan-88  9:25:22
--
-- Copyright (C) 1985, 1986 by Xerox Corporation. All rights reserved.
--
-- This module contains the version number for the system being built. It should
-- be included in the top level config file. The Control Module will check its
-- version number against the client's version number.
--
DIRECTORY
  OfflineDiagInterfaceExtra       USING [];  -- Exporting it

OfflineDiagnosticVersionImpl: PROGRAM  EXPORTS OfflineDiagInterfaceExtra =

BEGIN

  -- This constant should be changed for each release to reflect its version
  -- number.
  --
  versionForThisRelease: LONG STRING = "2.0";

  GetVersion: PUBLIC PROCEDURE RETURNS [version: LONG STRING] =
              { RETURN [versionForThisRelease] };

  END... OfflineDiagnosticVersionImpl

LOG

Created on 19-Dec-85 by KL
Edited on 25-Aug-86 15:31:40  by SPL
Edited on  9-Jan-87 13:33:17  by STC, Experimental
Edited on  13-Jan-87 15:15:17  by SPL, Set to 1.3c
Edited on   3-Mar-87 14:36:20  by KXW, Set to 1.5b
Edited on   17-Mar-87  9:50:06  by KXW, Set to 1.5c
Edited on    6-Apr-87 14:19:36  by KXW, Set to 1.5d
6-Apr-87 14:19:36  by STC, Set to version 1.5e
15-May-87 10:26:01  by STC, Set to version 1.5f
23-Jul-87 16:28:08  by STC, Set to version 1.5s
27-Aug-87 13:34:44  by STC, Set to version 1.5g
17-Sep-87 11:07:35  by STC, Set to version 1.5
 7-Oct-87 13:43:37  by STC, Set to version 2.0e
17-Nov-87  9:00:20  by STC, Set to version 2.0f
 4-Dec-87  8:53:13  by STC, Set to version 2.0
17-Dec-87 11:23:25  by STC, Set to version 2.0g
21-Jan-88  9:25:12  by STC, Set to version 2.0, the official final 2.0
```

```
-- File: OfflineDiagTTYDove.mesa
-- Last edited on  11-Mar-87 12:10:57 by KXW
--
-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.
--
-- This interface module provides access to OfflineDiagTTYImplDove. This is
-- a stripped down version of the original DiagSimpleTTY. Only the essential
-- items are provided to save space.
--
DIRECTORY
  UserTerminal  USING [CursorArray],
  KeyStations   USING [DownUp, KeyStation];

OfflineDiagTTYDove: DEFINITIONS =

  BEGIN
    -- There are 3 possible screen sizes: 15", 17" and 19". The following are
    -- upper limits, ie maximum dimemsions, for the largest screen size (19").
    --
    maxNumOfCharsPerLine: CARDINAL = 144;  -- pixelsPerLine/font.maxWidth.
    maxLines: CARDINAL = 71;  -- screenHeight/fontHeight (Gacha12Strike).

    -- Ranges for character positions and line numbers.
    --
    xRange: TYPE = [0..maxNumOfCharsPerLine);   -- Possible character positions.
    yRange: TYPE = [0..maxLines);   -- Possible line numbers.

    -- Actual limits exported by the implementation module.
    --
    rightEdge: READONLY CARDINAL;  --  Character positions on a line.
    bottomLine: READONLY CARDINAL; -- Number of lines on screen.
    charHeight: READONLY CARDINAL;  -- This is the character height ir bits.
    bitsPerTextLine: READONLY CARDINAL;  -- Number of bits per text line.

    OutOfRange: ERROR;  -- Out of display limitation.

    -- Current character position and line number exported by implementation
    --                                                                  module.
    charPos: READONLY CARDINAL;      -- [0..rightEdge)
    currentLine: READONLY CARDINAL;  -- [0..maxLines)

    -- Keyboard information
    --
    oldKeys: READONLY LONG POINTER TO PACKED ARRAY KeyStations.KeyStation
             OF KeyStations.DownUp;
    newKeys: READONLY LONG POINTER TO PACKED ARRAY KeyStations.KeyStation
             OF KeyStations.DownUp;

    -- KeyDescriptor:
    --
    -- This is used for Keyboard diagnostics. Each key on the keyboard is
    -- described by a KeyDescriptor.
    --
    KeyDescriptor: TYPE = RECORD [
        x: CARDINAL, -- [0..UserTerminal.screenWidth) = Position on a raster
        y: CARDINAL ← 0, -- [0..UserTerminal.screenHeight) = raster line number
        width: CARDINAL ← 0, -- [0..UserTerminal.screenWidth)
        height: CARDINAL ← 0]; -- [0..UserTerminal.screenHeight)

    -- diagKeyBoard
    --
    -- Language-independent pointer to a particular keyboard implementation,
    -- described in KDMMsgesAndKBDsImplDove.mesa
    --
    diagKeyBoard: READONLY POINTER TO ARRAY KeyStations.KeyStation
                                       OF KeyDescriptor;

    -- DiagPutText:
    --
    --   xCoordinate = Horizontal position of character starting from 0. Unit of
    --                 measure is font.maxWidth.
    --   yCoordinate = Line number on bitmapped screen starting from 0. Unit of
    --                 measure is fontHeight (Gacha12Strike).
    --   text is pointer to an arbitrarily long/short text to be printed.
    --                 NIL pointer => No text to print. Just position the cursor.
    --   DiagPutText[] non-destructively homes cursor.
    --
    DiagPutText: PROC[xCoordinate: xRange ← 0, yCoordinate: yRange ← 0,
                      text: LONG STRING ← NIL];


    -- PutObject
    --
    --   Put an UserTerminal.CursorArray object (16X16 pixels) on the display
    --   at location (x,y). x is the pixel position on a scanline; y is the
    --   scanline on the screen.
    --
    PutObject: PROCEDURE [x, y: CARDINAL,
                      object: LONG POINTER TO UserTerminal.CursorArray];


    -- SetTTYMode and TTYMode:
    --
    --   The OfflineDiagTTY has three modes of operation. abnormalMode and
    --   kbdDiagnostic are used exclusively by Mouse, Dispaly and Keyboard
    --   diagnostics. All other clients should use the normal mode, which is the
    --   default mode set by the OfflineDiagnosticControlModule.
```

```
    --
    TTYMode: TYPE = {normal, abnormalMode, kbdDiagnostic};

    SetTTYMode: PROC [mode: TTYMode ← normal];


    -- FillScreenWithObject:
    --
    --   Paint the display with objects made up of arrays of 16 words
    --
    FillScreenWithObject: PROCEDURE [object: LONG POINTER TO
                                        UserTerminal.CursorArray];

    -- DrawALine:
    --
    --   Draws a horizontal line of arbitrary length and thickness on the screen.
    --   Ink is either black (all 1's) or white (all 0's).
    --
    DrawALine: PROCEDURE [x, y, thickness, lineLength, ink: CARDINAL];


    -- FillRectangle:
    --
    --  1) A key is represented by a rectangle with the dimensions as specified
    --     in its KeyDescriptor.
    --  2) "key" is a bit position in the keyboard bit map in the IOREGION.
    --  3) "paint" is a CARDINAL of all 1's or all 0's.
    --  4) When a key is down, it is painted black.
    --  5) When a key is up, it is painted white.
    --
    FillRectangle: PROCEDURE [key: KeyStations.KeyStation, paint: CARDINAL];


    -- DrawKBDLine: For the exclusive use of Display diagnostics.
    --
    --   This procedure is specifically designed for drawing the keyboard
    --   outlines in the Data Area of the display.
    --
    --          horizontal = TRUE => Horizontal line (Top and bottom edges)
    --          horizontal = FALSE => Vertical line (Left and right edges)
    --
    DrawKBDLine: PROCEDURE [x, yRelMiddle, lineLength, ink: CARDINAL,
                            horizontal: BOOLEAN];


    END...  OfflineDiagTTYDove

LOG

19-Jan-85: Creation by KL
18-Feb-85: Deleted Inline PROC's
24-Jan-86: Moved KeyDescriptor and diagKeyBoard here from KDMMsgesAndK8DsDove
11-Mar-87 12:11:02: Added ERROR OutOfRange.
```

```
-- File: OfflineDiagTTYImplDove.mesa.
-- Last edited on    20-Mar-87  9:49:22 by  KXW
--
-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.
--
-- Due to severe space constraints, unnecessary features are removed. This
-- module is a stripped down version of oldOfflineDiagTTYImplDove.mesa
--
DIRECTORY
  BitBlt                USING [AlignedBBTable, BBptr, BBTableSpace, BITBLT,
                               DstFunc, BitBltFlags, SrcDesc, GrayParm],
  Environment           USING [BitAddress, Block, Byte, bitsPerWord],
  Inline                USING [LongMult],
  Process               USING [Detach, EnableAborts, SetPriority, Yield],
  ProcessPriorities     USING [priorityIOHigh],
  Runtime               USING [GetTableBase, Interrupt],
  SpecialSpace          USING [MakeGlobalFrameResident,
                               MakeProcedureResident, SpecialError],
  Stream                USING [defaultObject, GetProcedure, Handle, Object,
                               PutProcedure, SendAttentionProcedure,
                               SetSSTProcedure, WaitAttentionProcedure],
  TTY                   USING [Handle, OutOfInstances],
  TTYConstants          USING [aborted, blinkDisplay, normal, notAborted,
                               removeChars],
  UserTerminal          USING [mouse, SetMousePosition, screenWidth, screenHeight,
                               BlinkDisplay, SetState, GetBitBltTable,
                               SetBackground, keyboard, WaitForScanLine,
                               Coordinate, CursorArray, SetCursorPosition,
                               SetCursorPattern, SetBorder],
  KeyStations           USING [KeyStation, KeyBits, DownUp],
  OfflineDiagTTYDove    USING [xRange, yRange, TTYMode, KeyDescriptor,
                               diagKeyBoard];

OfflineDiagTTYImplDove: MONITOR
  LOCKS m USING m: POINTER TO MONITORLOCK
  IMPORTS BitBlt, Inline, Process, Runtime, SpecialSpace, Stream, UserTerminal,
          TTY, OfflineDiagTTYDove
  EXPORTS TTY, OfflineDiagTTYDove =

BEGIN OPEN Dtty: OfflineDiagTTYDove,
           Env: Environment,
           Term: UserTerminal;

  screenLock:   MONITORLOCK;
  keyboardLock: MONITORLOCK;

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- Common Definitions
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  CHAR: TYPE = CHARACTER;
  ControlA:     CHARACTER = 'A - 100B;
  BS:           CHARACTER = 10C;        -- Non-destructive backspace.
  TAB:          CHARACTER = 11C;
  CR:           CHARACTER = 15C;
  SP:           CHARACTER = ' ;
  DEL:          CHARACTER = 177C;
  rubOut:       CHARACTER = 177C;       -- Destuctive back space (RubOut).
  ControlZ:     CHARACTER = 'Z - 100B; -- Home and clear screen (ClearScreen).
  CLFC:         CHARACTER = 34C;        -- ↑\ Clears line starting from carret

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- FONT Definitions and variables
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
font: LONG POINTER TO MACHINE DEPENDENT RECORD [
  -- Strike Header
  newStyle(0:0..0): BOOLEAN,
  indexed(0:1..1):  BOOLEAN,
  fixed(0:2..2):    BOOLEAN,
  kerned(0:3..3):   BOOLEAN,
  pad(0:4..15):     [0..7777B],
  min(1):           CHARACTER,   -- First char in font
  max(2):           CHARACTER,   -- Last char in font
  maxwidth(3):      CARDINAL,  -- Width of characters in font
  -- Strike Body
  length(4):        CARDINAL,  -- Total number of words in the Strike Body.
  ascent(5):        CARDINAL,  -- Ascent + Descent = Character height.
  descent(6):       CARDINAL,
  xoffset(7):       CARDINAL,  -- 0.
  raster(8):        CARDINAL,  -- Number of words per scan-line in the Strike.
  chars(9:0..63):  SELECT OVERLAID * FROM
    hasBoundingBox => [
      boundingBox(9:0..63): RECORD [
        FontBBox, FontBBoy, FontBBdx, FontBBDy: INTEGER],
      BBBitmap(13):         ARRAY [0..0) OF WORD],
    noBoundingBox => [
      -- A large bitmap of height scanlines
      bitmap(9):            ARRAY [0..0) OF WORD], -- (ascent+descent) scanlines
    ENDCASE] = GetFont[];

bitmap: LONG POINTER = IF font.kerned THEN @font.BBBitmap ELSE @font.bitmap;

-- Height of a character in Strike.

  charHeight: PUBLIC CARDINAL = font.ascent+font.descent;
  wordSize: CARDINAL = Environment.bitsPerWord;

-- Pointer into the Strike Body, indexed by character codes.
```

```
    xInSegment: LONG POINTER TO ARRAY CHARACTER [0C..0C) OF CARDINAL =
                bitmap + font.raster*charHeight - (font.min-0C);

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Keyboard Definitions and Constants
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    keyCount: CARDINAL = LAST[KeyStations.KeyStation];  -- Number of keys
    old, new: KeyStations.KeyBits;
    oldKeys: PUBLIC LONG POINTER TO KeyStations.KeyBits = @old;
    newKeys: PUBLIC LONG POINTER TO KeyStations.KeyBits = @new;
    keyRep: LONG POINTER TO ARRAY KeyStations.KeyStation OF Dtty.KeyDescriptor;

    KeyItem: TYPE = RECORD [
    Letter: BOOLEAN, ShiftCode: CHAR [0C..177C], NormalCode: CHAR [0C..377C]];

  -- Keyboard Info
    ctrl:       CARDINAL =  52;
    leftShift:  CARDINAL =  57;
    shiftLock:  CARDINAL =  72;
    rightShift: CARDINAL =  76;
    STOPkey:    CARDINAL =  77;
    RETURNkey:  CARDINAL =  60;
    spaceKey:   CARDINAL =  73;
    lastKey:    CARDINAL =  LAST[KeyStations.KeyStation];

  -- JLevel-IV info
    jShift1:    CARDINAL =  47;
    jShift2:    CARDINAL = 111;
    jExtraSP:   CARDINAL = 110;

  -- Range of normal keys as detailed in KeyTable.
  --  NormalKeyCount: TYPE = [16..77];
    NormalKeyCount: TYPE = [0..111];

  --*****************************************************************************
  -- This table transforms the bits from the keyBoard bitmap in the IORegion into
  -- an ASCII character code. The ASCII code is then used as indices to get a
  -- Gacha12Strike font representation for the ASCII code to be displayed.
  --
  -- Note: The numeric digits on the numeric keypad are "wire-ored" in this table
  --       with their corresponding digits on the main key array.
  --*****************************************************************************

  -- Table format: Letter/notLetter, Shifted ASCII code, Unshifted ASCII code

  KeyTable: ARRAY NormalKeyCount OF KeyItem = [

  --   Index[0..7]
    [FALSE,  00C,  00C],  -- null
    [FALSE,  00C,  00C],  -- Bullet
    [FALSE,  00C,  00C],  -- SuperSub
    [FALSE,  00C,  00C],  -- Case
    [FALSE,  00C,  00C],  -- Strikeout
    [FALSE,  62C,  62C],  -- KeypadTwo
    [FALSE,  63C,  63C],  -- KeypadThree
    [FALSE,  47C,  47C],  -- SingleQuote

  --   Index[8..15]
    [FALSE,  53C,  53C],  -- KeypadAdd
    [FALSE,  55C,  55C],  -- KeypadSubtract
    [FALSE,  52C,  52C],  -- KeypadMultiply
    [FALSE,   57C,  57C],  -- KeypadDivide
    [FALSE,  05C,  04C],  -- KeypadClear (unShifted=Clear MA, shitfed=Clear DA)
    [FALSE,  00C,  00C],  -- Point (left)
    [FALSE,  00C,  00C],  -- Adjust (right)
    [FALSE,  00C,  00C],  -- Menu

  --   Index [16..31]
    [FALSE,  45C,  65C],  -- %,5
    [FALSE,  44C,  64C],  -- $,4
    [FALSE, 176C,  66C],  -- ~,6
    [TRUE,  105C, 145C],  -- E
    [FALSE,  46C,  67C],  -- &,7
    [TRUE,  104C, 144C],  -- D
    [TRUE,  125C, 165C],  -- U
    [TRUE,  126C, 166C],  -- V
    [FALSE,  51C,  60C],  -- ),0
    [TRUE,  113C, 153C],  -- K
    [FALSE,  30C,  55C],  -- 30C,-
    [TRUE,  120C, 160C],  -- P
    [FALSE,  77C,  57C],  -- ?,/
    [FALSE, 174C, 134C],  -- |,\
    [FALSE,  12C,  12C],  -- LF
    [FALSE,  10C,  10C],  -- BS

  --   Index [32..47]
    [FALSE,  43C,  63C],  -- #,3
    [FALSE, 100C,  62C],  -- @,2
    [TRUE,  127C, 167C],  -- W
    [TRUE,  121C, 161C],  -- Q
    [TRUE,  123C, 163C],  -- S
    [TRUE,  101C, 141C],  -- A
    [FALSE,  50C,  71C],  -- (,9
    [TRUE,  111C, 151C],  -- I
    [TRUE,  130C, 170C],  -- X
```

```
  [TRUE,  117C, 157C],  -- O
  [TRUE,  114C, 154C],  -- L
  [FALSE,  74C,  54C],  -- <,,
  [FALSE,  42C,  47C],  -- ",'
  [FALSE, 175C, 135C],  -- },]
  [FALSE,   0C,   0C],  -- SPARE2
  [FALSE,   0C,   0C],  -- SPARE1 (jShift1)

--   Index [48..63]
  [FALSE,  41C,  61C],  -- !,1
  [FALSE,  33C,  33C],  -- ESCAPE
  [FALSE,  14C,  14C],  -- non-destructive forespace
  [TRUE,  106C, 146C],  -- F
  [FALSE,    0C,  0C],  -- CONTROL
  [TRUE,  103C, 143C],  -- C
  [TRUE,  112C, 152C],  -- J
  [TRUE,  102C, 142C],  -- B
  [TRUE,  132C, 172C],  -- Z
  [FALSE,    0C,  0C],  -- LEFT SHIFT
  [FALSE,  76C,  56C],  -- >,.
  [FALSE,  72C,  73C],  -- :,;
  [FALSE,  15C,  15C],  -- CR
  [FALSE, 136C, 137C],  -- ↑,←
  [FALSE, 177C, 177C],  -- DEL
  [FALSE,    0C,  0C],  -- NOT USED (FR5)

--    Index [64..79]
  [TRUE,  122C, 162C],  -- R
  [TRUE,  124C, 164C],  -- T
  [TRUE,  107C, 147C],  -- G
  [TRUE,  131C, 171C],  -- Y
  [TRUE,  110C, 150C],  -- H
  [FALSE,  52C,  70C],  -- *,8
  [TRUE,  116C, 156C],  -- N
  [TRUE,  115C, 155C],  -- M
  [FALSE,   0C,   0C],  -- LOCK
  [FALSE,  40C,  40C],  -- SPACE
  [FALSE, 173C, 133C],  -- {,[
  [FALSE,  53C,  75C],  -- +,=
  [FALSE,   0C,   0C],  -- RIGHT SHIFT
  [FALSE,   0C,   0C],  -- SPARE3 (stop)
  [FALSE,   0C,   0C],  -- MOVE
  [FALSE,   0C,  07C],  -- UNDO, generates BEL. Used to invert screen

--  Index [80..95]
  [FALSE,  00C,  00C],  -- MARGIN
  [FALSE,  67C,  67C],  -- 7 on numeric key pad
  [FALSE,  70C,  70C],  -- 8 on numeric key pad
  [FALSE,  71C,  71C],  -- 9 on numeric key pad
  [FALSE,  64C,  64C],  -- 4 on numeric key pad
  [FALSE,  65C,  65C],  -- 5 on numeric key pad
  [FALSE,  00C,  00C],  -- English
  [FALSE,  66C,  66C],  -- 6 on numeric key pad
  [FALSE,   0C,   0C],  -- Katakana
  [FALSE,   0C,   0C],  -- COPY
  [FALSE,   0C,   0C],  -- FIND
  [FALSE,   0C,   0C],  -- AGAIN
  [FALSE,   0C,   0C],  -- Help
  [FALSE,   0C,   0C],  -- EXPAND
  [FALSE,  61C,  61C],  -- 1 on numeric key pad
  [FALSE,  00C,  00C],  -- DiagnosticBitTwo

--  Index [96..111]
  [FALSE,  00C,  00C],  -- DiagnosticBitOne
  [FALSE,   0C,   0C],  -- CENTER
  [FALSE,  60C,  60C],  -- 0 on numeric key pad
  [FALSE,   0C,   0C],  -- BOLD
  [FALSE,   0C,   0C],  -- ITALIC
  [FALSE,   0C,   0C],  -- UNDERLINE
  [FALSE,  00C,  00C],  -- Superscript
  [FALSE,  00C,  00C],  -- Subscript
  [FALSE,   0C,   0C],  -- LARGER/SMALLER
  [FALSE,  56C,  56C],  -- Period on numeric key pad
  [FALSE,  54C,  54C],  -- Comma on numeric key pad
  [FALSE,   0C,   0C],  -- < %European, LeftDakuonShift %Japanese
  [FALSE,  42C,  42C],  -- DoubleQuote
  [FALSE,   0C,   0C],  -- DEFAULTS
  [FALSE,  00C,  00C],  -- Hiragana %Japanese
  [FALSE,  00C,  00C]];  -- RightHandDakuonShift %Japanese


-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--  System stuff.
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   useCount:        CARDINAL ← 0;
   simpleStream:   Stream.Object ← Stream.defaultObject;
   mode: Env.Byte ← TTYConstants.normal;
   modeVal: CARDINAL ← 0;
   charsSeen: BOOLEAN ← FALSE;
   inputBlock: LONG POINTER TO Env.Block ← NIL;
   diagTTYMode: Dtty.TTYMode  ← Dtty.TTYMode[normal];
   kAmDiagnosticStopTyped: BOOLEAN ← FALSE;


-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- DISPLAY DEFINITIONS
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
    BBTable: BitBlt.BBTableSpace;
    bbPtr:   BitBlt.BBptr = BitBlt.AlignedBBTable[@BBTable];
    bitsPerLine: CARDINAL ← Term.screenWidth;
    screenHeight: CARDINAL ← Term.screenHeight;
    bitsPerTextLine: PUBLIC CARDINAL;  -- screenWidth*font.height.
    bitsPerScreenLine: CARDINAL;  -- Width of screen
    pixelsFromOrigin: LONG CARDINAL;  -- Number of pixels from the origin (0,0)


 -- Character positioning  definitions:

    rightEdge: PUBLIC CARDINAL ← 0;  -- Right-most character position on a line.
                                 -- Its value depends on screen width and font.maxWidth.
    charPos: PUBLIC CARDINAL ← 0;  -- Current character position.
    oldCharPos: CARDINAL ← 0;  -- For keeping old character position.

 -- Line positioning definitions:

    bottomLine: PUBLIC CARDINAL;  --Number of permissible text lines.
    currentLine: PUBLIC CARDINAL; -- Current line carret is on.
    oldLine:  CARDINAL ← 0; -- Old line.
    firstLine: Env.BitAddress; -- Beginning of first display line for text
    trueOrigin: Env.BitAddress; -- (0,0) for bitmap
    thisLine, oldThisLine: Env.BitAddress; -- Beginning of curent line
    referenceLine: CARDINAL = Term.screenHeight/2;  -- Where to draw keyboard

 -- Carret control definitions:

    doBlink: BOOLEAN ← FALSE;  -- Blinking is enabled when TRUE.
    blinkerIsDark: BOOLEAN ← FALSE; -- Keeps state of carret (TRUE => it is dark).

 -- Error
    OutOfRange: PUBLIC ERROR = CODE;

 --~~~~~~~~~~~~~~~~
 -- Mouse stuff
 --~~~~~~~~~~~~~~~~

 --*****************************************************************************
 -- The mouse is represented by a 16X16 pixel square for diagnostics. The mouse
 -- buttons are represented by 2 sqares, each of which is inverted when depressed.
 --
 --       point => darken left square;
 --       adjust => darken right square;
 --       chord => darken both sqares.
 --
 --*****************************************************************************

   fifteenBits: CARDINAL = 15;  -- heightMinusOne

   textPointerMouse: Term.CursorArray = [  --
       100000B, 140000B, 160000B, 170000B, 174000B, 176000B, 177000B, 170000B,
       154000B, 114000B, 006000B, 006000B, 003000B, 003000B, 001400B, 001400B];

 << %Space
   upArrowMouse: Term.CursorArray = [  -- Fat up arrow
       200B, 700B, 1740B, 3760B, 7770B, 16734B, 30706B, 700B, 700B, 700B, 700B,
       700B, 700B, 700B, 700B, 700B];

   hourGlass: Term.CursorArray = [  -- Hour Glass
       177777B, 100001B, 040002B, 034034B, 017170B, 007560B, 003740B, 001700B,
       001100B, 002440B, 004220B, 010610B, 021704B, 047762B, 177777B, 177777B];

   questionMark: Term.CursorArray = [  -- ?
       017000B, 037600B, 060600B, 140300B, 140300B, 060300B, 000600B, 001400B,
       003000B, 006000B, 006000B, 006000B, 000000B, 000000B, 006000B, 006000B];

   bullseye: Term.CursorArray = [  --
       003700B, 007740B, 014060B, 030030B, 060014B, 140006B, 141606B, 141606B,
       141606B, 140006B, 060014B, 030030B, 014060B, 007740B, 003700B, 000000B];
 >>



 --~~~~~~~~~~~~~~~~~~
 -- Paints
 --~~~~~~~~~~~~~~~~~~

   blackPaint: CARDINAL = 177777B;
   whitePaint: CARDINAL ← 0;   -- For general purpose clearing

 --*****************************************************************************
 --  PUBLIC procedure for creating one and only one TTY instance.
 --*****************************************************************************
 CreateTTYInstance: PUBLIC PROCEDURE [name: LONG STRING, backingStream:
                                     Stream.Handle, tty: TTY.Handle]
                    RETURNS [ttyImpl: Stream.Handle, backing: Stream.Handle] = {
   CreateEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] RETURNS [CARDINAL]=
              INLINE {RETURN[useCount ← useCount+1]};
   IF CreateEntry[@keyboardLock] # 1 THEN ERROR TTY.OutOfInstances;
   RETURN[@simpleStream, NIL]};


 -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 --  TTY Procedures to be encapsulated in a Stream.Object.
 -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
--*****************************************************************************
--  For disposing the stream.
--*****************************************************************************
Delete: PROCEDURE [stream: Stream.Handle] = {
  DestroyEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] =
                INLINE {useCount ← useCount-1};
  DestroyEntry[@keyboardLock]};


--*****************************************************************************
--   Get the next byte from the stream and store it away.
--*****************************************************************************
GetBlock: Stream.GetProcedure = {
  c: CHAR = GetChar[];  -- Get input.
  sst ← TTYConstants.normal;
  why ← normal;
  bytesTransferred ← 1;
  block.blockPointer[block.startIndex] ← LOOPHOLE[c]};  -- Store in buffer.


--*****************************************************************************
--  Put a byte into the next available position in the stream from memory.
--*****************************************************************************
PutBlock: Stream.PutProcedure = {
  SELECT mode FROM
    TTYConstants.normal =>
      FOR i: CARDINAL IN [block.startIndex..block.stopIndexPlusOne) DO
          PutChar[LOOPHOLE[block.blockPointer[i]]] ENDLOOP;
    TTYConstants.removeChars =>
      BEGIN
        FOR i: CARDINAL IN [block.startIndex..block.stopIndexPlusOne) DO
          modeVal ← modeVal*256 + block.blockPointer[i];
          ENDLOOP;
        END;
    ENDCASE};


--*****************************************************************************
--   Wait for an attention to arrive.
--*****************************************************************************
WaitAttention: Stream.WaitAttentionProcedure = {
  DoIt: ENTRY PROCEDURE [m: POINTER TO MONITORLOCK]
              RETURNS [Env.Byte] =  INLINE {
                RETURN[IF stopTyped THEN TTYConstants.aborted
                       ELSE TTYConstants.notAborted]};
  RETURN[DoIt[@keyboardLock]]};


--*****************************************************************************
--   Send an attention down the stream.
--*****************************************************************************
SendAttention: Stream.SendAttentionProcedure = {
  DoIt: ENTRY PROCEDURE [m: POINTER TO MONITORLOCK] =  INLINE {
    SELECT byte FROM
      TTYConstants.aborted => stopTyped ← TRUE;
      TTYConstants.notAborted =>stopTyped ← FALSE;
      ENDCASE};
    DoIt[@keyboardLock]};


--*****************************************************************************
--   Change the current SubSequenceType.
--*****************************************************************************
SetSST: Stream.SetSSTProcedure =  {
  SELECT mode FROM
    TTYConstants.removeChars =>
      FOR i: CARDINAL IN [0..modeVal) DO
        PutChar[BS];
        PutChar[SP];
        PutChar[BS];
        ENDLOOP;
    TTYConstants.blinkDisplay => Term.BlinkDisplay[];
    ENDCASE;
  mode ← sst;
  modeVal ← 0};


--*****************************************************************
--  Change OfflineDiagTTY between the normal or diagnostic modes.
--*****************************************************************
SetTTYMode: PUBLIC PROC [mode: Dtty.TTYMode] = {
  -- First initialize all keys to "up"
  IF mode # normal THEN { -- Initialize keyboard
     old ← ALL[KeyStations.DownUp[up]];
     new ← ALL[KeyStations.DownUp[up]] };

  diagTTYMode ← mode;     -- Set TTY mode
  IF mode = normal
     THEN { doBlink ← TRUE;    -- Returning from KDM diagnostics
            charsSeen ← FALSE; in ← out  -- Flush out input buffer -- }
     ELSE { doBlink ← FALSE; PutChar [ControlZ] }};  -- Doing KDM diagnostics


-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--  Additional PUBLIC procedures
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
--*****************************************************************************
--   PutText allows a client to position the carret and optionally place a
--   string of text there. text = NIL => repositions carret only.
--*****************************************************************************
DiagPutText: PUBLIC PROC[xCoordinate: OfflineDiagTTYDove.xRange ← 0, -- Line pos.
                         yCoordinate: OfflineDiagTTYDove.yRange ← 0, -- Line no.
                         text: LONG STRING ← NIL] = {          -- Pointer to text.
  PutTextEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] = INLINE
    BEGIN
      inputTextChar: CHAR;
      yxBpl: LONG CARDINAL;
      IF blinkerIsDark THEN ClearBlinker[];

      -- thisLine and firstLine point to beginning of respective screen lines
      yxBpl ← LONG[(yCoordinate)]*bitsPerTextLine;  -- May be big number.
      -- Calculate the number of words to index into destination bitmap.
      yxBpl ← yxBpl/wordSize;  -- Number of words from beginning.
      thisLine.word ← firstLine.word + CARDINAL[yxBpl];  -- Desired  text line.
      bbPtr.dst ← thisLine;  -- Fix up BBTable to point to starting line and
      bbPtr.dst ← GetBitAddress[bbPtr.dst.word,   -- word and bit positions.
                    (bbPtr.dst.bit+(xCoordinate*font.maxwidth))];
      bbPtr.srcDesc ← [srcBpl[font.raster*wordSize]];
      bbPtr.height ← charHeight;  -- Font.ascent + Font.descent.
      bbPtr.width ← font.maxwidth;
<<
bbPtr.dst ← GetBitAddress[firstLine.word, LONG[(yCoordinate)]*bitsPerTextLine];
bbPtr.dst ← GetBitAddress[bbPtr.dst.word,   -- word and bit positions.
                    (bbPtr.dst.bit+(xCoordinate*font.maxwidth))];
>>

      charPos ← xCoordinate;  -- Update current char position counter.
      currentLine ← yCoordinate;   -- Update current-line-position counter.

      IF text = NIL THEN RETURN;        -- NIL pointer => Position carret only.
      FOR i: CARDINAL IN [0..text.length)  -- Output text.
        DO
          inputTextChar ← text[i];
          DisplayChar[inputTextChar];
          ENDLOOP;
      doBlink ← TRUE;
      END; --  of PutTextEntry

  IF xCoordinate > rightEdge OR yCoordinate > bottomLine THEN ERROR OutOfRange;
  PutTextEntry[@screenLock]};  -- End of DiagPutText

--~~~~~~~~~~~~~~~~~~~~~
--  Diagnostic stuffs
--~~~~~~~~~~~~~~~~~~~~~

--*****************************************************************************
--
--      1) A rectangle is a thick horizontal line representing a key.
--      2) "key" is the bit position of a key station in the keyboard bit map.
--      3) "paint" is a CARDINAL of all 1's or all 0's.
--      4) The inside edges of all rectangles overlap by 2 pixels.
--      5) Special keys:
--
--         13 (Point) contains the dimensions for lower half of the RETURN key.
--
--*****************************************************************************

-- Paints the internal portion of a rectangle as defined in a KeyDescriptor.

FillRectangle: PUBLIC PROC [key: KeyStations.KeyStation, paint: CARDINAL] =
  BEGIN
    anEdge: CARDINAL = 2;  -- Adjustment for one edge (2 pixels)
    twoEdges: CARDINAL = 4;  -- Adjustment for 2 edges
    RETURNkey: CARDINAL = 60;  -- Return key requires special handling
    RETURNkeyLowerHalf: CARDINAL = 13;  -- For painting RETURN key lower half
    pixelsToRefLine: LONG CARDINAL ← Inline.LongMult[referenceLine, bitsPerLine];
    rightEdgeFactor: CARDINAL = 1000;
    rightEdgeFudge: CARDINAL = 998;
    length: CARDINAL;

    keyRep ← Dtty.diagKeyBoard;
    length ← IF keyRep[key].width > rightEdgeFactor THEN
              keyRep[key].width - rightEdgeFudge ELSE keyRep[key].width;

    -- Calculate number of pixels from pixelsFromOrigin
    --                        Lines into Data Area                Bits into line
    pixelsFromOrigin ← LONG[keyRep[key].y + anEdge]*bitsPerLine + keyRep[key].x;
    pixelsFromOrigin ← pixelsFromOrigin + pixelsToRefLine + anEdge;

    DrawLineEntry  [m: @screenLock,
                    thickness: keyRep[key].height - anEdge,
                    lineLength: length - twoEdges,
                    ink: paint,
                    startingPixel: pixelsFromOrigin];
                    -- horizontal: TRUE] ;

    IF key = RETURNkey THEN { -- RETURN is represented by two rectangles.

       -- Draw lower half  RETURNlowX        RETURNtopX
       pixelsFromOrigin ← LONG[keyRep[RETURNkeyLowerHalf].y]*bitsPerLine
                          + keyRep[RETURNkeyLowerHalf].x;
       pixelsFromOrigin ← pixelsFromOrigin + pixelsToRefLine;
```

```
        DrawLineEntry [m: @screenLock,
                thickness: keyRep[RETURNkeyLowerHalf].height,
                lineLength: keyRep[RETURNkeyLowerHalf].width,
                ink: paint,
                startingPixel: pixelsFromOrigin] }
                -- horizontal: TRUE] }

    END;  -- FillRectangle


--*****************************************************************************
--  Fill screen with UserTerminal.CursorArray objects
--*****************************************************************************
FillScreenWithObject: PUBLIC PROCEDURE [object: LONG POINTER TO
                                                UserTerminal.CursorArray] =
  BEGIN

    FillScreenEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] = INLINE {
      -- Set up BITBLT table for painting the object
      bbPtr.dst ← trueOrigin;   -- firstLine;
      bbPtr.dstBpl ← bitsPerScreenLine;
      bbPtr.src ← [object, 0, 0];
      bbPtr.srcDesc ← [gray[[0, 0, 0, fifteenBits -- heightMinusOne --]]];
      bbPtr.width ← bitsPerLine;
      bbPtr.height ← screenHeight;
      bbPtr.flags ← [gray: TRUE];
      BitBlt.BITBLT[bbPtr];  -- Paint it
      bbPtr.flags ← [] }; -- End FillScreenEntry

    UserTerminal.SetCursorPattern [ALL[0]];
    PutChar [Control Z];   -- Clear screen
    FillScreenEntry [@screenLock];
    END;  -- FillScreenWithObject


--*****************************************************************************
-- Put an 16X16 object at (x,y)
--*****************************************************************************

PutObject: PUBLIC PROCEDURE [x, y: CARDINAL, -- In pixels/scanlines
                             object: LONG POINTER TO UserTerminal.CursorArray] =
  BEGIN

    PutObjectEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] = INLINE {
      bbPtr.dst ← GetBitAddress[firstLine.word, LONG[y]*bitsPerScreenLine];
      bbPtr.dst ← GetBitAddress[bbPtr.dst.word, (bbPtr.dst.bit + x)];
      bbPtr.src ← [object, 0, 0];
      bbPtr.srcDesc ← [gray[[0, 0, 0, fifteenBits -- heightMinusOne --]]];
      bbPtr.width ← 16;
      bbPtr.height ← 16;
      bbPtr.flags ← [gray: TRUE];
      BitBlt.BITBLT[bbPtr];  -- Paint it
      bbPtr.flags ← [] }; -- End PutObjectEntry

    IF x > bitsPerScreenLine OR y > screenHeight THEN ERROR OutOfRange;
    PutObjectEntry [@screenLock];
    END;  -- PutObject


--*****************************************************************************
--  Draws a horizontal line of arbitrary length and thickness on the
--  screen.  This is a general purpose routine, whereas DrawKBDLine is tailored
--  for drawing the keyboard lines.
--*****************************************************************************

DrawALine: PUBLIC PROCEDURE [x, y, thickness, lineLength, ink: CARDINAL] =
  BEGIN
  IF (x + lineLength > bitsPerLine) OR (y + thickness > screenHeight)
       THEN ERROR OutOfRange;  -- Just in case

    pixelsFromOrigin ← LONG[y]*bitsPerLine + x;
    DrawLineEntry [@screenLock, thickness, lineLength, ink, pixelsFromOrigin];
    END;  -- DrawALine


--*****************************************************************************
-- This procedure is specifically designed for drawing the keyboard in the Data
-- Area of the display. The value yRelMiddle is a relative offset from the middle
-- of the screen.
--*****************************************************************************

DrawKBDLine: PUBLIC PROCEDURE [x, yRelMiddle, lineLength, ink: CARDINAL,
                               horizontal: BOOLEAN] =
  BEGIN

    edgeThickness: CARDINAL = 2;
    referenceLine: CARDINAL = Term.screenHeight/2;  -- Where to draw keyboard

    -- Convert x and yRelMiddle to values in terms of bits from origin
    pixelsFromOrigin ← LONG[yRelMiddle]*bitsPerLine + x -- Offset
        + Inline.LongMult [referenceLine, bitsPerLine];  -- Reference
    -- Now draw the line
    DrawLineEntry [m: @screenLock,
                   thickness: IF horizontal THEN edgeThickness ELSE lineLength,
                   lineLength: IF horizontal THEN lineLength ELSE edgeThickness,
                   ink: blackPaint,
                   startingPixel: pixelsFromOrigin -- , orientation -- ];
```

```
    END;  -- DrawKBDLine


--*****************************************************************************
--  This ENTRY procedure draws a horizontal or vertical line on the screen.
--  A line can be of arbitrary length and thickness.  All graphics drawn here
--  are composed of lines as defined in this module.
--*****************************************************************************
DrawLineEntry: ENTRY PROC [m: POINTER TO MONITORLOCK,
                           thickness, lineLength, ink: CARDINAL,
                           startingPixel: LONG CARDINAL] =
  BEGIN
     erase: BitBlt.DstFunc = and;
     blend: BitBlt.DstFunc = or;
     longWordSize: CARDINAL = 2*wordSize;

     IF thickness = 0 OR lineLength = 0 THEN RETURN;

     -- First set up BITBLT arguments
     bbPtr.flags ← [gray: TRUE];    -- Use gray brick
     bbPtr.srcDesc ← [gray[[0, 0, 0, 0]]];  -- Gray brick

     -- Set up BITBLT table for drawing the line
     bbPtr.dst ← GetBitAddress [trueOrigin.word, startingPixel];
     bbPtr.src ← [word: @ink, reserved: 0, bit: 0];
     bbPtr.height ← IF thickness < Term.screenHeight THEN thickness ELSE Term.screenHeight;
     bbPtr.width ← IF lineLength < Term.screenWidth THEN lineLength ELSE Term.screenWidth;
     bbPtr.flags.dstFunc ← IF ink = whitePaint THEN erase ELSE blend;

--     bbPtr.height ← IF horizontal THEN thickness ELSE lineLength;
--     bbPtr.width ← IF horizontal THEN lineLength ELSE thickness;

     -- Draw line
     BitBlt.BITBLT[bbPtr];
     -- Clear flags
     bbPtr.srcDesc ← [srcBpl[font.raster*wordSize]];
     bbPtr.flags ← [];

     END; -- DrawLineEntry

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--  Keyboard Implementations
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   stopTyped:           BOOLEAN ← FALSE;
   charactersAvailable: CONDITION;


--*****************************************************************************
--   Keyboard RingBuffer is 50 characters deep.
--*****************************************************************************

  buffer:  PACKED ARRAY [0..50) OF CHAR;
  in: CARDINAL ← 0;  -- Points to ring buffer location for the next input char.
  out: CARDINAL ← 0; -- Points to ring buffer location holding the output.

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--  KEYBOARD PROCEDURES
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


--*****************************************************************************
--  All inputs from the keyboard are first buffered in the Keyboard RingBuffer.
--*****************************************************************************
StuffBuffer: ENTRY PROC [c: CHAR, m: POINTER TO MONITORLOCK] = INLINE {
  newin: CARDINAL;
  IF (newin←in+1) = LENGTH[buffer] THEN newin ← 0;  -- Wrap around.
  IF newin#out THEN {buffer[in] ← c; in ← newin}};  -- Put input into buffer.


--*****************************************************************************
--  Gets the input/inputs from the Keyboard RingBuffer.
--*****************************************************************************
GetChar: PROC RETURNS [c: CHAR] = {
  P: ENTRY PROC [m: POINTER TO MONITORLOCK] = INLINE {
    ENABLE UNWIND => NULL;
    WHILE in=out DO WAIT charactersAvailable ENDLOOP;  -- in=out => no new input.
    c ← buffer[out];
    IF (out←out+1) = LENGTH[buffer] THEN out ← 0};  -- Wrap around.
  P[@keyboardLock]};


--*****************************************************************************
--   Update mouse position.
--*****************************************************************************
TrackMouseCursor:  PROC = INLINE {
  mouse: Term.Coordinate ← Term.mouse↑;
  mouse.x ← MIN[MAX[0, mouse.x], bitsPerLine];
  mouse.y ← MIN[MAX[0, mouse.y], screenHeight];
  Term.SetCursorPosition[mouse];
  Term.SetMousePosition[mouse] };
--  UserTerminal.SetCursorPattern [textPointerMouse]};


--*****************************************************************************
--  This is the main idle loop. It is the detached process that constantly looks
--  for inputs from the keyboard.
```

```
--*************************************************************************
ProcessKeyboard: PROC =
  BEGIN
  pKeyarray: LONG POINTER TO KeyStations.KeyBits = LOOPHOLE [Term.keyboard];

  blinkCount:     CARDINAL ← 33;  -- Blink carret when count = 0.
  interruptState: KeyStations.DownUp ← up;

  Process.SetPriority [ProcessPriorities.priorityIOHigh];
    -- allows interpret calls from the debugger (using priorityPageFaultIO
    -- instead would cause a state vector deadlock if an interpret call
    -- took a page fault).
  old ← pKeyarray↑;  -- Initialize old Keyarray.

  DO  -- Input/Output handler loop.

    rubItOut: CARDINAL = 31;  -- "←" is RubOut here.
    Brdcst:  ENTRY PROC [m: POINTER TO MONITORLOCK] = INLINE {
      BROADCAST charactersAvailable};  -- Activate process waiting on queue.

    XmitChar:  PROC [c: CHARACTER] = {  -- Stuff input char in ring buffer.
      StuffBuffer[c, @keyboardLock]; charsSeen ← TRUE};

    -- Look for inputs.  Keyboard and mouse are polled during retrace.
    Term.WaitForScanLine[0];  --  Retrace and system update.
    new ← pKeyarray↑;  --  Get keyboard status after system update.

    -- See if any request for world swap.
    IF new[STOPkey]=down AND (new[rightShift]=down OR new[leftShift]=down) THEN {
      IF interruptState=up THEN Runtime.Interrupt[];  -- Do world swap.
      interruptState ← down; LOOP};
    interruptState ← up;

    TrackMouseCursor[];  -- Update mouse position.

    IF diagTTYMode = abnormalMode AND (new[spaceKey]=up AND old[spaceKey]=down)
       THEN { XmitChar[SP]; Brdcst[@keyboardLock] };

    IF diagTTYMode # normal THEN {
       IF old[STOPkey]=up AND new[STOPkey]=down THEN { stopTyped ← TRUE; LOOP };
       old ← new;
       Process.Yield[];
       LOOP};

    UserTerminal.SetCursorPattern [textPointerMouse];
    IF (blinkCount ← blinkCount - 1) <= 1 THEN {
          BlinkCarret[];
          blinkCount ← 34 };

    FOR j: CARDINAL IN [0 .. keyCount) DO
      char: CHAR; entry: KeyItem;

      -- Normal mode of operation
      IF new[j]=up OR old[j]=down THEN LOOP; -- Act on first instance of change
      IF j=STOPkey THEN {stopTyped ← TRUE; LOOP};

      -- ASCII NULL or keys that have no significance for diagnostics
      IF (char ← (entry ← KeyTable[j]).NormalCode) = 0C THEN LOOP;

      IF new[leftShift] =down OR new[rightShift]=down
         OR new[jShift1]=down OR new[jShift2]   =down
         OR (new[shiftLock]=down AND entry.Letter) THEN  -- Capitol letters.
         char ← entry.ShiftCode;

      IF j = rubItOut THEN char ← 177c;   -- Destructive backup

      XmitChar[char]; -- Stuffs input char in ring buffer.
      ENDLOOP;  -- End of NormalKeyCount loop.

    IF new[jExtraSP]=down AND old[jExtraSP] = up THEN XmitChar[SP];
    IF charsSeen THEN Brdcst[@keyboardLock];  --Activate process waiting on queue.
    old ← new;
    ENDLOOP
  END;  -- ProcessKeyboard

--- ~~~~~~~~~~~~~~~~~~~~
--  EXTERNAL PROCEDURES
--- ~~~~~~~~~~~~~~~~~~~~

--*************************************************************************
--  GetBitAddress sets up the src or dst pointers for the next BITBLT operation.
--  The word and bit position pointers are properly adjusted to point to the
--  beginning word and bit positions of the source or destination bitmap to be
--  operated on.
--*************************************************************************
GetBitAddress: PROCEDURE [reference: LONG POINTER, bitsFromOrigin: LONG CARDINAL]
                RETURNS [Env.BitAddress] = {
  RETURN [[reference + LOOPHOLE[CARDINAL [bitsFromOrigin/wordSize]], 0
           CARDINAL [bitsFromOrigin MOD wordSize]]] };


--*************************************************************************
--  The carret used here is a black rectangle located at font.max+1. Before
--  moving the carret, ensure the black rectangle is cleared first.
--*************************************************************************
BlinkCarret: PROC = INLINE {
  BlinkCarretEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] = {
```

```
    blinker: CHAR = font.max+1;
    IF doBlink THEN {
      bbPtr.src ← GetBitAddress[bitmap,xInSegment[blinker]];  -- Dark rectangle.
      blinkerIsDark ← ~ blinkerIsDark;  -- Complements blinkerIsDark.
      bbPtr.flags ← [dstFunc: xor];  -- XOR carret and displayed character.
      BitBlt.BITBLT[bbPtr];
      bbPtr.flags ← []};  -- Clear XOR
  IF doBlink THEN BlinkCarretEntry[@screenLock]};


--*****************************************************************************
--  External procedure for accessing screen.
--*****************************************************************************
PutChar: PROC [c: CHARACTER] = {
  PutCharEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] =
    INLINE {doBlink ← FALSE;
    DisplayChar[c]; IF diagTTYMode = normal THEN doBlink ← TRUE};
  PutCharEntry[@screenLock]};


--*****************************************************************************
--  Entry procedure for clearing screen.
--*****************************************************************************
ClearScreenEntry: ENTRY PROC [m: POINTER TO MONITORLOCK] = INLINE {ClearScreen[];
  doBlink ← TRUE};  -- Start blinking the carret.

--- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
---  INTERNAL DISPLAY PROCEDURES
--- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

--*****************************************************************************
--   ClearBlinker ensures that the carret is cleared before moving on.
--*****************************************************************************
ClearBlinker: INTERNAL PROC = BEGIN
  blinkerIsDark ← FALSE;  --  Blinker is clear.
  bbPtr.src ← GetBitAddress[bitmap,xInSegment[font.max+1]];
  bbPtr.flags ← [dstFunc: xor];
  BitBlt.BITBLT[bbPtr];
  bbPtr.flags ← [];
  doBlink ← FALSE  -- Disable blinking. Re-enabled by caller.
  END;


--*****************************************************************************
--   Backup does a non-destructive backspace.
--*****************************************************************************
Backup: INTERNAL PROC = {
  t: CARDINAL = bbPtr.dst.bit + wordSize - font.maxwidth;
  IF blinkerIsDark THEN ClearBlinker[]; --Ensure no dark rectangle is left behind.
  IF charPos = 0 THEN RETURN; -- Do nothing if at first char position of line.
  charPos ← charPos - 1;
  bbPtr.dst.word ← bbPtr.dst.word + t/wordSize - 1;
  bbPtr.dst.bit ← t MOD wordSize;
  doBlink ← TRUE};  -- Re-enable blinker.


--*****************************************************************************
--   ClearScreen clears the screen and homes the carret. Additionally, the
--   user terminal dimensions, which can vary from system to system, are obtained
--   from the system via UserTerminal.GetBitBltTable. The following constants are
--   initilaized here and are based on the information returned by
--   UserTerminal.GetBitBltTable:
--      bitsPerScreenLine
--      bitsPerTextLine
--      firstLine
--      rightEdge
--      bottomLine
--      trueOrigin
--*****************************************************************************
ClearScreen: INTERNAL PROC = {
  bbPtr↑ ← Term.GetBitBltTable[];
  trueOrigin ← bbPtr.dst;  -- Save for later use
  bitsPerScreenLine ← bbPtr.dstBpl;  -- Save destination bits per line
  bitsPerTextLine ← bitsPerScreenLine * charHeight;
  firstLine ← thisLine ← GetBitAddress[  -- Point to beginning of first line.
    bbPtr.dst.word, bbPtr.dst.bit+8+8*bbPtr.dstBpl];
  charPos ← 0; currentLine ← 0;
  --Last character position on line. Here, bbPtr.width is width of entire display.
  rightEdge ← (bbPtr.width-wordSize)/font.maxwidth;
  -- Max number of screen lines. bbPtr.height is screen height here.
  bottomLine ← (bbPtr.height - wordSize)/charHeight;
  bbPtr.src ← [@whitePaint, 0, 0];
  bbPtr.srcDesc ← [gray[[0, 0, 0, 0]]];
  bbPtr.flags ← [gray: TRUE];  -- Alter interpretation of srcBpl to srcDesc.
  BitBlt.BITBLT[bbPtr];
  -- set up standard arguments for character painting
  bbPtr.dst ← firstLine;
  --bbPtr.dstBpl set
  --bbPtr.src set when proc called
  bbPtr.srcDesc ← [srcBpl[font.raster*wordSize]];
  bbPtr.height ← charHeight;  -- Font.ascent + Font.descent.
  bbPtr.width ← font.maxwidth;
  bbPtr.flags ← []};  -- Clear the flags, especially the gray flag.


--*****************************************************************************
--   Clears the current screen position pointed to by the carret.
```

```
--*****************************************************************************
ClearThisChar: INTERNAL PROC = {
  bbPtr.src ← [@whitePaint, 0, 0];
  bbPtr.srcDesc ← [gray[[0, 0, 0, 0]]];
  bbPtr.flags ← [gray: TRUE];
  BitBlt.BITBLT[bbPtr];
  bbPtr.srcDesc ← [srcBpl[font.raster*wordSize]];
  bbPtr.flags ← []};


--*****************************************************************************
--   rubOut (177C) destructively backs up by one position
--*****************************************************************************
RubOut: INTERNAL PROC = {
  ClearThisChar[];
  IF charPos = 0 THEN RETURN;          -- Do nothing if at beginning of line
  Backup[];
  doBlink ← TRUE};


--*****************************************************************************
--   ↑\ (34c) invokes ClearLineFromCarret, which clears the remainder  of the
--   line starting from the current carret position.
--*****************************************************************************
ClearLineFromCarret: INTERNAL PROC = {
  -- First, save the current position.
  oldThisLine ← thisLine;  -- Current line
  oldCharPos ← charPos;    -- Current character position.

  -- Clear rest of current line starting from current position.
  FOR i: CARDINAL IN [charPos..rightEdge) DO
    ClearThisChar[];
    IF i < (rightEdge-1) THEN
      bbPtr.dst ← GetBitAddress[bbPtr.dst.word, bbPtr.dst.bit+font.maxwidth];
    ENDLOOP;

  -- Restore the old position.
  bbPtr.dst ← thisLine ← oldThisLine;
  bbPtr.dst ← GetBitAddress[bbPtr.dst.word, bbPtr.dst.bit+                    (oldCharPos*font.maxwidth)];
  charPos ← oldCharPos;
  doBlink ← TRUE};


--*****************************************************************************
--   Newline does a Carriage Return and a Line Feed.
--*****************************************************************************
Newline: INTERNAL PROC = {
  doBlink ← FALSE;  -- Disable Blinking.
  IF currentLine<(bottomLine-1)THEN {   --Not currently at the bottom of screen.
    thisLine ← GetBitAddress[thisLine.word, thisLine.bit+bitsPerTextLine];
    currentLine ← currentLine+1}
  ELSE {  -- At bottom of screen. Need to move the entire screen up 1 line.
    sBBTable: BitBlt.BBTableSpace;
    sbbPtr: BitBlt.BBptr = BitBlt.AlignedBBTable[@sBBTable];
    sbbPtr↑ ← [
      dst: firstLine, dstBpl: bbPtr.dstBpl,  -- New destination bitmap.
      -- New source bitmap is old bitmap minus the bottom line.
      src: GetBitAddress[firstLine.word, firstLine.bit+bitsPerTextLine],
      srcDesc: [srcBpl[bbPtr.dstBpl]], flags: [direction: forward],
      width: -- CARDINAL ← -- rightEdge*font.maxwidth,
      height: -- CARDINAL ← -- charHeight*(bottomLine-1)];
    BitBlt.BITBLT[sbbPtr];
    sbbPtr↑ ← [
      dst: thisLine, src: [@whitePaint, 0, 0],
      dstBpl: bbPtr.dstBpl, srcDesc: [gray[[0, 0, 0, 0]]],
      width: rightEdge*font.maxwidth, height: charHeight,
      flags: [gray: TRUE]];
    BitBlt.BITBLT[sbbPtr]};
  bbPtr.dst ← thisLine;
  charPos ← 0;
  doBlink ← TRUE}; -- Re-enable blinking. End NewLine


--*****************************************************************************
--   DisplayChar interprets keyboard inputs and cause the correct sequence of
--   events to happen. Be carefull with RETURN's !!!
--*****************************************************************************
DisplayChar: INTERNAL PROC [c: CHARACTER] = {
  -- Ensure carret is cleared. Blinking is also disabled.
  IF blinkerIsDark THEN ClearBlinker[];
  -- Implement commands.
  SELECT c FROM
    IN (SP..'~] => {  -- Normal characters. Output to screen.
      IF c ~IN [font.min..font.max] THEN c ← font.max+1;
      bbPtr.src ← GetBitAddress[bitmap, xInSegment[c]];
      BitBlt.BITBLT[bbPtr] };
    rubOut => { RubOut[]; RETURN };
    SP =>  { ClearThisChar[] };  -- Enter a space.
    CR =>  { Newline[]; RETURN };  -- Does a carriage return and line feed.
    BS =>  { Backup[]; RETURN };  -- Non-destructive.
    ControlZ => { ClearScreen[]; RETURN };    -- Home carret and clear screen.
    CLFC => { ClearLineFromCarret[]; RETURN };
    IN [0C..SP) => RETURN;
    ENDCASE => RETURN;
  IF (charPos ← charPos+1) >= rightEdge THEN Newline[]
  ELSE bbPtr.dst ← GetBitAddress[bbPtr.dst.word, bbPtr.dst.bit+font.maxwidth];
  doBlink ← TRUE };  -- Enable Blinking.
```

```
--- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
---  Gacha12Strike Font Definition.
--- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

--*****************************************************************************
--  GetFont provides the pointer to the Gacha12Strike font file hard coded here.
--*****************************************************************************
GetFont: PROC RETURNS [LONG POINTER] = {
  Gacha12Strike: ARRAY [0..1470B) OF WORD = [
    --   0-- 120000B,     0B,   176B,    10B,  1464B,    13B,     3B,     0B,
    --  10--     61B,     0B,     0B,     0B, 10000B,     0B,  2040B,     0B,
    --  20--      0B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --  30--      0B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --  40--      0B,     0B,     0B,     0B,     0B,     0B,     0B,    74B,
    --  50--     74B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --  60--      0B,     0B,     0B,     0B,     0B,     0B,     0B,  7010B,
    --  70--  70000B,     0B,     0B,    10B, 22012B, 34144B, 14010B,  4020B,
    -- 100--   4000B,     0B,     2B, 36010B, 36074B,  6076B, 36176B, 36074B,
    -- 110--      0B,  2000B, 20074B, 14020B, 74034B, 74176B, 77034B, 41076B,
    -- 120--  17102B, 40306B, 41074B, 76074B, 76074B, 77502B, 40501B, 41101B,
    -- 130--  77040B, 20004B,     0B,   100B,     2B,    14B,   100B,  4004B,
    -- 140--  40070B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    -- 150--  10010B,  4000B,     0B,     0B,    10B, 22012B, 52244B, 22010B,
    -- 160--  10010B, 25000B,     0B,     2B, 41030B, 41102B,  6040B, 41002B,
    -- 170--  41102B,     0B,  4000B, 10102B, 22020B, 42042B, 42100B, 40042B,
    -- 200--  41010B,  1104B, 40306B, 61102B, 41102B, 41102B,  4102B, 40511B,
    -- 210--  41101B,  1040B, 20004B,     0B,   100B,     2B,    22B,   100B,
    -- 220--   4004B, 40010B,     0B,     0B,     0B,    20B,     0B,     0B,
    -- 230--      0B, 10010B,  4000B,     0B,     0B,    10B, 22024B, 52250B,
    -- 240--  22010B, 10010B, 16010B,     0B,     4B, 41050B, 41102B, 12040B,
    -- 250--  40002B, 41102B,     0B, 10000B,  4102B, 41050B, 42102B, 41100B,
    -- 260--  40102B, 41010B,  1110B, 40252B, 61102B, 41102B, 41102B,  4102B,
    -- 270--  21111B, 22042B,  2040B, 10004B,  4000B,   100B,     2B,    20B,
    -- 300--    100B,     0B, 40010B,     0B,     0B,     0B,    20B,     0B,
    -- 310--      0B,     0B, 10010B,  4000B, 74000B,     0B,    10B, 22076B,
    -- 320--  50310B, 14010B, 20004B, 25010B,     0B,     4B, 43010B,  1002B,
    -- 330--  12174B, 76004B, 41102B,  4010B, 20000B,  2002B, 47050B, 42100B,
    -- 340--  41100B, 40100B, 41010B,  1120B, 40252B, 51102B, 41102B, 41040B,
    -- 350--   4102B, 21111B, 22042B,  4040B, 10004B, 16020B, 34134B, 36072B,
    -- 360--  36174B, 35134B, 34074B, 42010B,167134B, 36134B, 35054B, 36174B,
    -- 370--  41102B,101104B, 41174B, 10010B,  4062B, 74000B,     0B,    10B,
    -- 400--     24B, 34010B, 10000B, 20004B,  4010B,   176B,    10B, 45010B,
    -- 410--   2034B, 22102B, 41004B, 36102B,  4010B, 40176B,  1004B, 51104B,
    -- 420--  76100B, 41174B, 76100B, 77010B,  1160B, 40252B, 51102B, 41102B,
    -- 430--  76030B,  4102B, 21052B, 14024B,  4040B,  4004B, 25040B, 42142B,
    -- 440--  41106B, 41020B, 43142B,  4004B, 44010B,111142B, 41142B, 43062B,
    -- 450--  41020B, 41102B,101104B, 41004B, 10010B,  4132B, 74000B,     0B,
    -- 460--    10B,    50B, 12020B, 24400B, 20004B,   177B,     0B,    10B,
    -- 470--  51010B,  4002B, 22002B, 41010B, 41102B,     0B, 40000B,  1010B,
    -- 500--  51104B, 41100B, 41100B, 40116B, 41010B,  1110B, 40222B, 45102B,
    -- 510--  76102B, 44004B,  4102B, 12052B, 14024B, 10040B,  4004B,  4177B,
    -- 520--   2102B, 40102B, 41020B, 41102B,  4004B, 50010B,111102B, 41102B,
    -- 530--  41040B, 40020B, 41044B,111050B, 21010B, 60010B,  3114B, 74000B,
    -- 540--  35400B,    10B,   174B, 12023B, 45000B, 20004B,    10B,     0B,
    -- 550--     20B, 61010B, 10002B, 42002B, 41010B, 41076B,     0B, 20176B,
    -- 560--   2010B, 46104B, 41100B, 41100B, 40102B, 41010B,  1104B, 40222B,
    -- 570--  45102B, 40102B, 42002B,  4102B, 12052B, 22010B, 20040B,  2004B,
    -- 600--   4040B, 36102B, 40102B, 77020B, 41102B,  4004B, 70010B,111102B,
    -- 610--  41102B, 41040B, 36020B, 41044B,111020B, 22020B, 10010B,  4000B,
    -- 620--  74000B,     0B,     0B,    50B, 52025B, 42000B, 20004B,    10B,
    -- 630--      0B,    20B, 41010B, 20102B, 77102B, 41020B, 41002B,     0B,
    -- 640--  10000B,  4000B, 40376B, 41102B, 41100B, 40102B, 41010B, 41104B,
    -- 650--  40222B, 43102B, 40102B, 42102B,  4102B, 12024B, 22010B, 20040B,
    -- 660--   2004B,  4020B, 42102B, 40102B, 40020B, 41102B,  4004B, 44010B,
    -- 670-- 111102B, 41102B, 41040B,  1020B, 41044B,111050B, 12040B, 10010B,
    -- 700--   4000B, 74000B,     0B,    10B,   120B, 52045B, 42000B, 10010B,
    -- 710--     10B, 14000B,  4040B, 41010B, 40102B,  2102B, 41020B, 41102B,
    -- 720--   4030B,  4000B, 10010B, 21202B, 41042B, 42100B, 40046B, 41010B,
    -- 730--  41102B, 40202B, 43102B, 40102B, 41102B,  4102B,  4024B, 41010B,
    -- 740--  40040B,  1004B,  4000B, 42142B, 41106B, 41020B, 43102B,  4004B,
    -- 750--  42010B,111102B, 41142B, 43040B, 41022B, 43030B,111104B, 14100B,
    -- 760--  10010B,  4000B, 74000B,     0B,    10B,   120B, 34046B, 35400B,
    -- 770--  10010B,     0B,  4000B,  4040B, 36076B, 77074B,  2074B, 36020B,
    --1000--  36074B,  4010B,  2000B, 20010B, 16202B, 76034B, 74176B, 40032B,
    --1010--  41076B, 36102B, 77202B, 41074B, 40074B, 41074B,  4074B,  4024B,
    --1020--  41010B, 77040B,  1004B,     0B, 35134B, 36072B, 36020B, 35102B,
    --1030--   4004B, 41010B,111102B, 36134B, 35040B, 36014B, 35030B, 66104B,
    --1040--  41764B, 10010B,  4000B, 74000B,     0B,     0B,     0B, 10000B,
    --1050--     0B,  4020B,     0B,  4000B,     0B,     0B,     0B,     0B,
    --1060--     0B,     0B,    10B,     0B,     0B,     0B,     0B,     0B,
    --1070--     0B,     0B,     0B,     0B,     0B,    20B,     0B,     0B,
    -1100--     0B,     0B,    40B,     4B,     0B,     0B,     0B,     0B,
    --1110--  1000B,     4B,     0B,     0B,   100B,  1000B,     0B,     0B,
    --1120--     0B,  4000B, 10010B,  4000B, 74000B,     0B,     0B,     0B,
    --1130--     0B,     0B,  2040B,     0B, 10000B,     0B,     0B,     0B,
    --1140--     0B,     0B,     0B,    20B,     0B,     0B,     0B,     0B,
    --1150--     0B,     0B,     0B,     0B,     0B,     0B,   16B,     0B,
    --1160--     0B,     0B,     0B,    74B,    74B,     0B,     0B,     0B,
    --1170--     0B, 41000B,   104B,     0B,     0B,   100B,  1000B,     0B,
    --1200--     0B,     0B, 50000B,  7010B, 70000B, 74000B,   377B,     0B,
    --1210--     0B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --1220--     0B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --1230--     0B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --1240--     0B,     0B,     0B,     0B,     0B,     0B,     0B,     0B,
    --1250--     0B,     0B, 36000B,    70B,     0B,     0B,   100B,  1000B,
```

```
    --1260--      0B,     0B,     0B, 20000B,     0B,     0B, 74000B,     0B,
    --1270--     10B,    10B,    10B,    10B,    10B,    10B,    10B,    10B,
    --1300--     10B,    10B,    10B,    10B,    10B,    10B,    10B,    10B,
    --1310--     10B,    10B,    10B,    10B,    10B,    10B,    10B,    10B,
    --1320--     20B,    20B,    20B,    20B,    20B,    20B,    20B,    20B,
    --1330--     30B,    40B,    50B,    60B,    70B,   100B,   110B,   120B,
    --1340--    130B,   140B,   150B,   160B,   170B,   200B,   210B,   220B,
    --1350--    230B,   240B,   250B,   260B,   270B,   300B,   310B,   320B,
    --1360--    330B,   340B,   350B,   360B,   370B,   400B,   410B,   420B,
    --1370--    430B,   440B,   450B,   460B,   470B,   500B,   510B,   520B,
    --1400--    530B,   540B,   550B,   560B,   570B,   600B,   610B,   620B,
    --1410--    630B,   640B,   650B,   660B,   670B,   700B,   710B,   720B,
    --1420--    730B,   740B,   750B,   760B,   770B,  1000B,  1010B,  1020B,
    --1430--   1020B,  1030B,  1040B,  1050B,  1060B,  1070B,  1100B,  1110B,
    --1440--   1120B,  1130B,  1140B,  1150B,  1160B,  1170B,  1200B,  1210B,
    --1450--   1220B,  1230B,  1240B,  1250B,  1260B,  1270B,  1300B,  1310B,
    --1460--   1320B,  1330B,  1340B,  1350B,  1360B,  1370B,  1400B,  1410B];
    -- End of Gacha12Strike font file. This is used as a source bitmap.

  p: LONG POINTER TO ARRAY [0..1470B) OF WORD
    ← Runtime.GetTableBase[LOOPHOLE[OfflineDiagTTYImplDove]]; -- Start of frame.
  DO  -- Look for beginning of font table embedded in code.
    IF p[0] = Gacha12Strike[0] AND p↑ = Gacha12Strike THEN RETURN[p];
    p ← p+1
    ENDLOOP};  -- p points to beginning of the Gacha12Strike font table at RETURN.


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- MAINLINE CODE --
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
<<
  FontError: ERROR = CODE;

  IF ~font.newStyle OR font.indexed OR font.min ~IN [0C..177C]
    OR font.max+1 ~IN [0C..177C] THEN ERROR FontError;
>>

-- Encapsulate the simple TTY operations in a Stream.Object.

  simpleStream.get ← GetBlock;
  simpleStream.put ← PutBlock;
  simpleStream.delete ← Delete;
  simpleStream.waitAttention ← WaitAttention;
  simpleStream.sendAttention ← SendAttention;
  simpleStream.setSST ← SetSST;

  Process.EnableAborts[@charactersAvailable];

  [] ← Term.SetState[on];
  [] ← Term.SetBackground[white];
  Term.SetBorder [oddPairs: 252B, evenPairs: 1258];
  Term.SetCursorPattern [textPointerMouse];   -- Use the Tajo mouse normally

  ClearScreenEntry[@screenLock];  -- Clear display and home carret.

  SetTTYMode [normal];  -- Set mode to normal

  -- Pin down ProcessKeyboard and OfflineDiagTTYImplDove  ??????????
  SpecialSpace.MakeProcedureResident[ProcessKeyboard
    ! SpecialSpace.SpecialError =>
      IF error = alreadyResident THEN CONTINUE
      ELSE REJECT];
  SpecialSpace.MakeGlobalFrameResident[OfflineDiagTTYImplDove
    ! SpecialSpace.SpecialError =>
      IF error = alreadyResident THEN CONTINUE
      ELSE REJECT];

  -- This is the detached process for sensing inputs.
  Process.Detach[FORK ProcessKeyboard];

  END.... OfflineDiagTTYImplDove.mesa


LOG

Created by KL on 15-Nov-84.
This program is based on the SimpleTTY last edited by Johnson on 6-Apr-83.

24-Jan-86: Broke the tie to KDMMsgesAndKBDsDove. Moved diagKeyBoard and
           KeyDescriptor to OfflineDiagTTYDove.

 5-Jan-87 by KXW,  Fixed bug in DiagPutText.
11-Mar-87 14:16:12  by KXW,  Modified DiagPutText, PutObject and DrawALine to check arguments.
20-Mar-87  9:49:02  by KXW,  Fixed bug in PutObject.
```

```
-- File: CMDiagMsgKeysDove.mesa
-- MXT                  17-Dec-85 15:25:34
-- Last edited on       26-Oct-87 15:41:54 by JMA

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.
--
-- Abstract
--
<< The CMDiagMsgKeysDove.mesa interface module defines an array of indices
   (DiagMessages). Each index corespondings to a diagnostic message in
   CMDiagMsgKeysImplDove.mesa.

   The implementation module (CMDiagMsgKeysImplDove.mesa) is language dependent.
   Non-English messages can replace the English messages contained in
   the message array. To do so, simply replace "EnglishMessages" in
   CMDiagMsgKeysImplDove.mesa with the foreign language equivalent of
   "EnglishMessages" in the target language, and change all the English
   messages to messages written in the new language.

   Note: CMDiagMsgKeysImplDove.bcd must be started to make the text of the
         message keys available. >>

CMDiagMsgKeysDove: DEFINITIONS =

BEGIN

  -- cmMsgKeys is a pointer to DiagMessages, an arrray of indices. The indices
  -- in turn point to diagnostic messages in CMDiagMsgKeysImplDove.mesa a
  -- language-dependent message table.
  --
  cmMsgKeys: READONLY LONG DESCRIPTOR FOR EnglishMessages;

  EnglishMessages: TYPE = ARRAY DiagMessages OF LONG STRING;

  DiagMessages: TYPE = {   -- Table of enumerated indices for diagnostic messages.

  -- Title prompts:

     copyRight,
     OfflineDiagTitle,
     running,

  -- Log on and password prompts:

     passWordPrompt,
     inCorrectPassword,
     administratorPWD,
     techRepPWD,
     manufacturePWD,
     diagProgrammerPWD,

     userSelectionPrompt,
     normalUser,
     administrator,
     techRep,

     -- Help info for log in

     loginHelpTitle,
     normalUserExplanation,
     otherUserClassExplanation,

  -- Test subsystem prompts.

     runEthernet,
     runFloppy,
     runFloppyUtility,
     runHardDisc,
     runRS232,
     runKbdDsplMouse,
     runPrinter,
     runTapeDrive,
     runFSBPU,
     runLaserDisc,
     runManufacture,
     runMisc,
     runSecureDeviceTests,
     sysConfigUtility,
     runScannerTests,
     runCartridgeTapeTests,
     selfTests,
     noClients,

  -- Interactive prompts.

     availableSelections,
     enterSelectionPrompt,
     helpForSelection,
     enterYesNoPrompt,
     enterNamePrompt,
     PassedMark,
     FailedMark,
     blankIt,
     ambiguousMark,
     lastSelection,
     noTestAllowed,
     goToHigherMenu,
```

```
        exitCurrentMenu,
        noTests,
        moreText,
        yes,
        Yes,
        no,
        No,
        legalDigits,
        moreData,
        dash,
        decimalDigitsOnly,
        illegalInput,
        isOutOfRange,
        enterHexNumber,
        enterDecimalNumber,
        testSelected,
        noSuchOption,
        noHelpForThisOption,
        catSelectionHelpTitle,
        catSelectionHelp0,
        catSelectionHelp1,
        catSelectionHelp2,
        catSelectionHelp3,
        catSelectionHelp4,
        catSelectionHelp5,
        catSelectionHelp6,
        exitWarning,
        testDataNotCleared,
        nameTooBig,
        anyKeyToContinue,
        veriosnMismatch,
        kernelVersion,
        clientVersion,
        mismatchOption,

        -- for Floppy Exec msg Keys.
        floppyExec,
        menuName,
        insertDiskLabeled,
        diskForWSDiag,
        diskForScannerDiag,
        diskForCartTapeDiag,
        isDiskReady,
        fileNotFound,
        hardwareError,
        notReady,
        noSuchDrive,
        invalidFormat,
        needsScavenging,
        loading,
        noDiagPkg,
        checkDisk,
        instanceExists,
        tryAgain,
        insertDisk,
        cantErase,
        finishLoading,
        quit,
        aborted,
        unloading,
        finished,
        cantFindConfig

        };

    InitializeCMDiagMsgKeys: PROC;

    END... of CMDiagMsgKeysDove.mesa

LOG

Created on 7-Mar-85 by KL
Added items for Floppy Exec on 17-Dec-85 15:25:34 by MXT
Changed for use heap for messages on 9-Feb-87 19:21:18 by KXW
Changed to include scanner disgnostics on  5-Mar-87 10:15:31 by KXW
Added runScannerTests test entry on 12-Mar-87 13:52:36 by JMA
Added entries for Cartridge Tape test entry on 26-Oct-87 15:41:49 by JMA
```

```
-- File: CMDiagMsgKeysImplDove.mesa
-- Lasted edited on 29-Jan-88 15:46:17 by STC
--
-- Copyright (C) 1985, 1986, 1987, 1988 by Xerox Corporation. All rights reserved.
--
-- This file implements the English version of CMDiagMsgKeysDove. The messages
-- are used by various kernel modules.
--
-- To implement non-English messages, perform the following:
--
--    1) Replace "English" with <LanguageOfConcern>.
--    2) Replace English messages with message in the language of concern.
--
DIRECTORY
  CMDiagMsgKeysDove      USING   [DiagMessages, EnglishMessages],
  Heap                   USING   [Create],
  String                 USING   [CopyToNewString];


CMDiagMsgKeysImplDove: PROGRAM
  IMPORTS Heap, String
  EXPORTS CMDiagMsgKeysDove =

BEGIN
  OPEN S: String;

  cmMsgKeys: PUBLIC LONG DESCRIPTOR FOR CMDiagMsgKeysDove.EnglishMessages;

  z: UNCOUNTED ZONE;

  InitializeCMDiagMsgKeys: PUBLIC PROC = {

  -- cmMsgKeys points to the array of English messages used by the Control Module
  -- initialization

  z ← Heap.Create[initial: 10];

  cmMsgKeys ← DESCRIPTOR[z.NEW[CMDiagMsgKeysDove.EnglishMessages], 104];

  -- Title prompts:
  cmMsgKeys[copyRight] ←
   S.CopyToNewString["Copyright (C) Xerox Corporation 1985, 1986, 1987, 1988. All rights reserved."L, z];
  cmMsgKeys[OfflineDiagTitle] ←
   S.CopyToNewString["Offline Diagnostics Version"L, z];
  cmMsgKeys[running] ←
   S.CopyToNewString["Running:"L, z];

  -- Logon and password prompts:
  cmMsgKeys[passWordPrompt] ←
   S.CopyToNewString["Please enter password: "L, z];
  cmMsgKeys[inCorrectPassword] ←
   S.CopyToNewString["Incorrect password"L, z];
  -- Passwords are defined here
  cmMsgKeys[administratorPWD] ←
   S.CopyToNewString["rgmsn"L, z];
  cmMsgKeys[techRepPWD] ←
   S.CopyToNewString["rexifsn"L, z];    -- For services engineers.
  cmMsgKeys[manufacturePWD] ←
   S.CopyToNewString["foo"L, z];
  cmMsgKeys[diagProgrammerPWD] ←
   S.CopyToNewString["bar"L, z];

  cmMsgKeys[userSelectionPrompt] ←
   S.CopyToNewString["What class of user do you belong to?"L, z];
  cmMsgKeys[normalUser] ←
   S.CopyToNewString["1 - Normal User"L, z];
  cmMsgKeys[administrator] ←
   S.CopyToNewString["2 - System Administrator"L, z];
  cmMsgKeys[techRep] ←
   S.CopyToNewString["3 - Technical Support"L, z];

  -- Help info for log in.
  cmMsgKeys[loginHelpTitle] ←
   S.CopyToNewString["Log In Help"L, z];
  cmMsgKeys[normalUserExplanation] ←
   S.CopyToNewString["Normal users can only run harmless tests that give qualitative indications. No login is needed"L, z];
  cmMsgKeys[otherUserClassExplanation] ←
   S.CopyToNewString["Other users can run tests that may damage the system. They must login with a valid password"L, z];

  --  Menu for selecting subsystems
  cmMsgKeys[runEthernet] ←
   S.CopyToNewString["Ethernet Tests"L, z];
  cmMsgKeys[runFloppy] ←
   S.CopyToNewString["Floppy Disk Tests"L, z];
  cmMsgKeys[runFloppyUtility] ←
   S.CopyToNewString["Floppy Disk Utility"L, z];
  cmMsgKeys[runHardDisc] ←
   S.CopyToNewString["Rigid Disk Tests"L, z];
  cmMsgKeys[runRS232] ←
   S.CopyToNewString["RS232C Tests"L, z];
  cmMsgKeys[runKbdDsplMouse] ←
   S.CopyToNewString["Keyboard/Display/Mouse/Beeper Tests"L, z];
  cmMsgKeys[runPrinter] ←
   S.CopyToNewString["Printer Tests"L, z];
  cmMsgKeys[runTapeDrive] ←
   S.CopyToNewString["Cartridge Tape Tests"L, z];
  cmMsgKeys[runFSBPU] ←
```

```
    S.CopyToNewString["Formatter, Scavenger and Bad Page Utility"L, z];
  cmMsgKeys[runLaserDisc] ←
    S.CopyToNewString["Laser DisK Tests"L, z];
  cmMsgKeys[runManufacture] ←
    S.CopyToNewString["Manufacturing Tests"L, z];
  cmMsgKeys[runMisc] ←
    S.CopyToNewString["Miscellaneous Tests"L, z];
  cmMsgKeys[runSecureDeviceTests] ←
    S.CopyToNewString["Secure Information Device Tests"L, z];
  cmMsgKeys[sysConfigUtility] ←
    S.CopyToNewString["System Configuration Utility"L, z];
  cmMsgKeys[runScannerTests] ←
    S.CopyToNewString["Scanner Diagnostics"L, z];
  cmMsgKeys[runCartridgeTapeTests] ←
    S.CopyToNewString["Cartridge Tape Tests"L, z];
  cmMsgKeys[selfTests] ←
    S.CopyToNewString["Control Module Self-Tests"L, z];
  cmMsgKeys[noClients] ←
    S.CopyToNewString["Boot file is empty"L, z];

  -- Interactive prompts.
  cmMsgKeys[availableSelections] ←
    S.CopyToNewString["Available Selections"L, z];
  cmMsgKeys[enterSelectionPrompt] ←
    S.CopyToNewString["Please enter selection:"L, z];
  cmMsgKeys[helpForSelection] ←
    S.CopyToNewString["Please enter, inclusively, a number between 1 and "L, z];
  cmMsgKeys[enterYesNoPrompt] ←
    S.CopyToNewString["Please enter Y(es) or N(o):"L, z];
  cmMsgKeys[enterNamePrompt] ←
    S.CopyToNewString["Please enter a name:"L, z];
  cmMsgKeys[PassedMark] ←
    S.CopyToNewString["P"L, z]; -- Test ran successfully
  cmMsgKeys[FailedMark] ←
    S.CopyToNewString["F"L, z]; -- Test failed. At least one error exist.
  cmMsgKeys[blankIt] ←
    S.CopyToNewString["  "L, z];
  cmMsgKeys[ambiguousMark] ←
    S.CopyToNewString["?"L, z]; -- Test ran is not sure whether the test passed of failed
  cmMsgKeys[lastSelection] ←
    S.CopyToNewString["Last selection from this menu was:"L, z];
  cmMsgKeys[noTestAllowed] ←
    S.CopyToNewString["No selections assigned. Please enter any key to continue:"L, z];
  cmMsgKeys[goToHigherMenu] ←
    S.CopyToNewString["Go To Previous Menu"L, z]; -- Exit selection.
  cmMsgKeys[exitCurrentMenu] ←
    S.CopyToNewString["Exits this menu and returns to parent menu"L, z];
  cmMsgKeys[noTests] ←
    S.CopyToNewString["No Test To Run"L, z];
  cmMsgKeys[moreText] ←
    S.CopyToNewString["More to go. Display the rest? [Y/N]:"L, z];
  cmMsgKeys[yes] ←
    S.CopyToNewString["y"L, z]; -- This is yes.
  cmMsgKeys[Yes] ←
    S.CopyToNewString["Y"L, z]; -- This is Yes.
  cmMsgKeys[no] ←
    S.CopyToNewString["n"L, z]; -- This is no.
  cmMsgKeys[No] ←
    S.CopyToNewString["N"L, z]; -- This is No.
  cmMsgKeys[legalDigits] ←
    S.CopyToNewString["Legal decimal digits: 0 to 9, Legal hex digits: A/a to F/f, Directional inputs: +, -"L, z];
  cmMsgKeys[moreData] ←
    S.CopyToNewString["More data... Any key will clear current data and continue"L, z];
  cmMsgKeys[dash] ←
    S.CopyToNewString["-"L, z];
  cmMsgKeys[decimalDigitsOnly] ←
    S.CopyToNewString["Decimal digits only"L, z];
  cmMsgKeys[illegalInput] ←
    S.CopyToNewString["Illegal input"L, z];
  cmMsgKeys[isOutOfRange] ←
    S.CopyToNewString["is out of range."L, z];
  cmMsgKeys[enterHexNumber] ←
    S.CopyToNewString["Please enter a hexadecimal number"L, z];
  cmMsgKeys[enterDecimalNumber] ←
    S.CopyToNewString["Please enter a decimal number"L, z];
  cmMsgKeys[testSelected] ←
    S.CopyToNewString["Selection:"L, z];
  cmMsgKeys[noSuchOption] ←
    S.CopyToNewString["No such option"L, z];
  cmMsgKeys[noHelpForThisOption] ←
    S.CopyToNewString["No help for this option"L, z];

  cmMsgKeys[catSelectionHelpTitle] ←
    S.CopyToNewString["Useful information"L, z];
  cmMsgKeys[catSelectionHelp0] ←
    S.CopyToNewString["The blinking cursor points to the applicable prompt. Do as prompted"L, z];
  cmMsgKeys[catSelectionHelp1] ←
    S.CopyToNewString["SPACE and CR are input terminators. BACKSPACE erases the last input character"L, z];
  cmMsgKeys[catSelectionHelp2] ←
    S.CopyToNewString["A question mark alone (?) gives help to the entire menu"L, z];
  cmMsgKeys[catSelectionHelp3] ←
    S.CopyToNewString["<n> followed by ? gives help to the item identified by n"L, z];
  cmMsgKeys[catSelectionHelp4] ←
    S.CopyToNewString["UNDO inverts the screen"L, z];
  cmMsgKeys[catSelectionHelp5] ←
    S.CopyToNewString["STOP aborts the current test, if allowed; or exits the current menu"L, z];
  cmMsgKeys[catSelectionHelp6] ←
```

```
    S.CopyToNewString["The current selection is displayed at the top of the screen"L, z];
  cmMsgKeys[exitWarning] ←
    S.CopyToNewString["Test data will be deleted upon exit. Is this OK?"L, z];
  cmMsgKeys[testDataNotCleared] ←
    S.CopyToNewString["Test data not deleted"L, z];
  cmMsgKeys[nameTooBig] ←
    S.CopyToNewString["Input is too long"L, z];
  cmMsgKeys[anyKeyToContinue] ←
    S.CopyToNewString["Enter any key to continue:"L, z];
  cmMsgKeys[veriosnMismatch] ←
    S.CopyToNewString["Version mismatch!!!"L, z];
  cmMsgKeys[kernelVersion] ←
    S.CopyToNewString["First floppy version:"L, z];
  cmMsgKeys[clientVersion] ←
    S.CopyToNewString["Second floppy version:"L, z];
  cmMsgKeys[mismatchOption] ←
    S.CopyToNewString["Should I ignore version mismatch and continue? [Y/N]"L, z];

  -- For Floppy Exec
  cmMsgKeys[floppyExec] ←
    S.CopyToNewString["Floppy Executive"L, z];
  cmMsgKeys[menuName] ←
    S.CopyToNewString["Load Offline Diagnostics Package"L, z];
  cmMsgKeys[insertDiskLabeled] ←
    S.CopyToNewString["Insert Floppy Disk Labeled:"L, z];
  cmMsgKeys[diskForWSDiag] ←
    S.CopyToNewString["6085 Offline Diagnostics Disk for Workstation Diagnostics."L, z];
  cmMsgKeys[diskForScannerDiag] ←
    S.CopyToNewString["6085 Offline Diagnostics Disk for Pro Imager Diagnostics."L, z];
  cmMsgKeys[diskForCartTapeDiag] ←
    S.CopyToNewString["6085 Offline Diagnostics Disk for VP Cartridge Tape Diagnostics."L, z];
  cmMsgKeys[isDiskReady] ←
    S.CopyToNewString["Is the Required  Disk now loaded?"L, z];
  cmMsgKeys[fileNotFound] ←
    S.CopyToNewString["File not found on floppy"L, z];
  cmMsgKeys[hardwareError] ←
    S.CopyToNewString["Floppy disk drive hardware problem"L, z];
  cmMsgKeys[notReady] ←
    S.CopyToNewString["Floppy not in drive or drive not Ready"L, z];
  cmMsgKeys[noSuchDrive] ←
    S.CopyToNewString["No such drive"L, z];
  cmMsgKeys[invalidFormat] ←
    S.CopyToNewString["Floppy needs formatting"L, z];
  cmMsgKeys[needsScavenging] ←
    S.CopyToNewString["Floppy needs scavenging"L, z];
  cmMsgKeys[loading] ←
    S.CopyToNewString["Loading diagnostics from the second floppy disk..."L, z];
  cmMsgKeys[noDiagPkg] ←
    S.CopyToNewString["No diagnostic file was found on the floppy disk"L, z];
  cmMsgKeys[checkDisk] ←
    S.CopyToNewString["Please check the floppy disk"L, z];
  cmMsgKeys[instanceExists] ←
    S.CopyToNewString["Offline Diagnostics Package already exists"L, z];
  cmMsgKeys[tryAgain] ←
    S.CopyToNewString["Let's try this one more time..."L, z];
  cmMsgKeys[insertDisk] ←
    S.CopyToNewString["Insert auxiliary diagnostic disk, press RETURN to continue."L, z];
  cmMsgKeys[cantErase] ←
    S.CopyToNewString["Can't erase the loaded file"L, z];
  cmMsgKeys[finishLoading] ←
    S.CopyToNewString["Diagnostics loaded successfully."L, z];
  cmMsgKeys[quit] ←
    S.CopyToNewString["The load state is invalid. Please reboot and try again."L, z];
  cmMsgKeys[aborted] ←
    S.CopyToNewString["Okay, I've aborted... Reboot is needed."L, z];
  cmMsgKeys[unloading] ←
    S.CopyToNewString["Unloading loaded programs..."L, z];
  cmMsgKeys[finished] ←
    S.CopyToNewString["Finished"L, z];
  cmMsgKeys[cantFindConfig] ←
    S.CopyToNewString["Can't find config link"L, z];

};


  END... of CMDiagMsgKeysImplDove.mesa


LOG

Created on 19-Mar-85 by KL
Added Floppy exec items on 17-Dec-85 15:25:57 by MXT
Changed wording of runTapeDrive on 24-Jul-86 16:24:56 by RK
Used Heap than Global Frame for string, on 12-Feb-87 20:32:58 by KXW
Fixed AR10296 on 27-Feb-87 13:37:07 by KXW
Added some items to include scanner disgnostics on 5-Mar-87 14:07:18 by KXW
Added runScannerTests entry for scanner diagn on 12-Mar-87 13:57:05 by JMA
Fixed the number of message array elements on 18-Mar-87 11:45:41 by KXW
Changed isDiskReady on 28-Apr-87 11:37:33 by KXW
Changed Manuf and Program passwords on 27-Aug-87 16:08:24 by STC
Remove # from diskForWSDiag and diskForScannerDiag on  8-Sep-87 14:24:20 by STC
change scanner to Pro Imager on  17-Sep-87 11:06:58 by STC
Added entries for cartridge tape diagnostics on 26-Oct-87 15:42:36 by JMA
Added 1988 in copyRight, add cartridge tape on 21-Jan-88  9:28:15 by STC
```

```
-- File: FormWindow.mesa - last edit:
-- Breisacher.ES        11-Dec-84 10:04:02
-- JPhillips.es         14-May-84 15:57:21
-- SAJohnson.ES         15-May-84 11:12:24
-- JGS                   6-Jun-84 13:56:05

-- Copyright (C) 1984 by Xerox Corporation

DIRECTORY
  BlackKeys USING [Keyboard],
  Environment USING [BitAddress],
  MenuData USING [MenuHandle],
  SimpleTextDisplay USING [Flushness, StreakSuccession],
  Window USING [Box, Dims, Handle],
  XLReal USING [Number, zero],
  XString USING [Reader, ReaderBody];

FormWindow: DEFINITIONS =
  BEGIN

  -- Types

  -- Item Characteristics

  Bitmap: TYPE = RECORD [
    height, width: CARDINAL,
    bitsPerLine: CARDINAL,
    bits: Environment.BitAddress];

  BooleanItemLabelType: TYPE = {string, bitmap};

  BooleanItemLabel: TYPE = RECORD [
    var: SELECT type: BooleanItemLabelType FROM
      string => [string: XString.ReaderBody],
      bitmap => [bitmap: Bitmap],
      ENDCASE];

  ChangeReason: TYPE = {user, client, restore};

  ChoiceItemType: TYPE = {string, bitmap, wrapIndicator};

  ChoiceIndex: TYPE = CARDINAL [0..37777B];

  ChoiceItem: TYPE = RECORD [
    var: SELECT type: ChoiceItemType FROM
      string => [choiceNumber: ChoiceIndex, string: XString.ReaderBody],
      bitmap => [choiceNumber: ChoiceIndex, bitmap: Bitmap],
      wrapIndicator => NULL,
      ENDCASE];

  ChoiceItems: TYPE = LONG DESCRIPTOR FOR ARRAY ChoiceIndex OF ChoiceItem;

  ItemKey: TYPE = CARDINAL;

  ItemType: TYPE = MACHINE DEPENDENT{
    choice(0), multiplechoice, decimal, integer, boolean, text, command, tagonly,
    window, last(15)};

  Line: TYPE [2];

  TabType: TYPE = {fixed, vary};

  TabStops: TYPE = RECORD [
    variant: SELECT type: TabType FROM
      fixed => [interval: CARDINAL],
      vary => [list: LONG DESCRIPTOR FOR ARRAY OF CARDINAL]
    ENDCASE];

  TextHintAction: TYPE = {replace, append, nil};
    -- replace means the hint replaces the current string in the item.
    -- insert means insert at the current type-in point (useful for AdobeQuery-like fields).
    -- nil means don't replace the string at all.

  Visibility: TYPE = {visible, invisible, invisibleGhost};
    -- visible = visible and functional
    -- invisible = invisible and takes up NO space
    -- invisibleGhost = invisible item BUT takes up space

  -- Procedure Types

  BooleanChangeProc: TYPE = PROCEDURE [
    window: Window.Handle, item: ItemKey, calledBecauseOf: ChangeReason,
    newValue: BOOLEAN];

  ChoiceChangeProc: TYPE = PROCEDURE [
    window: Window.Handle, item: ItemKey, calledBecauseOf: ChangeReason,
    oldValue, newValue: ChoiceIndex];

  ChoiceHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [
      hints: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex,
      freeHints: FreeChoiceHintsProc];

  CommandProc: TYPE = PROCEDURE [window: Window.Handle,
    item: ItemKey, clientData: LONG POINTER];

  FreeChoiceHintsProc: TYPE = PROCEDURE [
```

```
    window: Window.Handle, item: ItemKey,
    hints: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex];

  FreeTextHintsProc: TYPE = PROCEDURE [
    window: Window.Handle, item: ItemKey,
    hints: LONG DESCRIPTOR FOR ARRAY OF XString.ReaderBody];

  GlobalChangeProc: TYPE = PROCEDURE [
    window: Window.Handle, item: ItemKey,
    calledBecauseOf: ChangeReason, clientData: LONG POINTER];

  LayoutProc: TYPE = PROCEDURE [
    window: Window.Handle, clientData: LONG POINTER];

  MakeItemsProc: TYPE = PROCEDURE [
    window: Window.Handle, clientData: LONG POINTER];

  MultipleChoiceChangeProc: TYPE = PROCEDURE [
    window: Window.Handle, item: ItemKey, calledBecauseOf: ChangeReason,
    oldValue: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex,
    newValue: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex];

  NextIntoProc: TYPE = PROCEDURE [window: Window.Handle, item: ItemKey];

  NextOutOfProc: TYPE = PROCEDURE [window: Window.Handle, item: ItemKey];

  TextHintsProc: TYPE = PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [
      hints: LONG DESCRIPTOR FOR ARRAY OF XString.ReaderBody,
      freeHints: FreeTextHintsProc,
      hintAction: TextHintAction ← replace];


  -- Constants and data objects

  nextTabStop: CARDINAL = LAST[CARDINAL];

  nullItemKey: ItemKey = LAST[CARDINAL];

  -- Signals and errors

  Error: ERROR [code: ErrorCode];

  ErrorCode: TYPE = MACHINE DEPENDENT {notAFormWindow(0), wrongItemType,
    invalidChoiceNumber, noSuchLine,
    alreadyAFormWindow, invalidItemKey,
    itemNotOnLine, duplicateItemKey,
    incompatibleLayout, alreadyLaidOut, last(15)};

  LayoutError: SIGNAL [code: LayoutErrorCode];

  LayoutErrorCode: TYPE = {onTopOfAnotherItem, notEnufTabsDefined};


  -- Procedures

  -- Create and Destroy, etc.

  Create: PROCEDURE [
    window: Window.Handle,
    makeItemsProc: MakeItemsProc,
    layoutProc: LayoutProc ← NIL, -- uses DefaultLayout
    windowChangeProc: GlobalChangeProc ← NIL,
    minDimsChangeProc: MinDimsChangeProc ← NIL,
    zone: UNCOUNTED ZONE ← NIL,
    clientData: LONG POINTER ← NIL];

  DefaultLayout: LayoutProc;

  Destroy: PROCEDURE [window: Window.Handle];

  GetClientData: PROCEDURE [window: Window.Handle] RETURNS [clientData: LONG POINTER];

  GetGlobalChangeProc: PROC [window: Window.Handle]
    RETURNS [proc: FormWindow.GlobalChangeProc];

  GetZone: PROCEDURE [window: Window.Handle] RETURNS [zone: UNCOUNTED ZONE];

  IsIt: PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN];

  MinDimsChangeProc: TYPE = PROCEDURE [window: Window.Handle,
    old, new: Window.Dims];

  NeededDims: PROCEDURE [window: Window.Handle]
    RETURNS [Window.Dims];
    -- Returns the minimum dimensions required for a window to
    --  hold all the currently visible items in the form.

  NumberOfItems: PROCEDURE [window: Window.Handle] RETURNS [CARDINAL];
    -- Returns the number of items currently in the form.
    -- Includes invisible items.
    -- Useful for clients that create more text items as the user NEXTs out of the last one.

  Repaint: PROCEDURE [window: Window.Handle];
    -- Forces a repaint of the form.

  SetGlobalChangeProc: PROCEDURE [window: Window.Handle,
```

```
    proc: GlobalChangeProc] RETURNS [old: GlobalChangeProc];

  -- Create and destroy form items, etc.

  MakeBooleanItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    changeProc: BooleanChangeProc ← NIL,
    label: BooleanItemLabel,
    initBoolean: BOOLEAN ← TRUE];

  MakeChoiceItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    values: ChoiceItems,
    initChoice: ChoiceIndex,
    fullyDisplayed: BOOLEAN ← TRUE,
    verticallyDisplayed: BOOLEAN ← FALSE,
    hintsProc: ChoiceHintsProc ← NIL,
    changeProc: ChoiceChangeProc ← NIL,
    outlineOrHighlight: OutlineOrHighlight ← highlight];

  OutlineOrHighlight: TYPE = {outline, highlight};

  MakeCommandItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    commandProc: CommandProc,
    commandName: XString.Reader,
    clientData: LONG POINTER ← NIL];

  MakeDecimalItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    signed: BOOLEAN ← FALSE,
    width: CARDINAL,
    initDecimal: XLReal.Number ← XLReal.zero,
    wrapUnderTag: BOOLEAN ← FALSE,
    hintsProc: TextHintsProc ← NIL,
    nextOutOfProc: NextOutOfProc ← NIL,
    displayTemplate: XString.Reader ← NIL,
    SPECIALKeyboard: BlackKeys.Keyboard ← NIL];

  MakeIntegerItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    signed: BOOLEAN ← FALSE,
    width: CARDINAL,
    initInteger: LONG INTEGER ← 0,
    wrapUnderTag: BOOLEAN ← FALSE,
    hintsProc: TextHintsProc ← NIL,
    nextOutOfProc: NextOutOfProc ← NIL,
    SPECIALKeyboard: BlackKeys.Keyboard ← NIL];

  MakeMenuItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    menu: MenuData.MenuHandle];

  MakeMultipleChoiceItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    values: ChoiceItems,
```

```
    initChoice: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex,
    fullyDisplayed: BOOLEAN ← TRUE,
    verticallyDisplayed: BOOLEAN ← FALSE,
    hintsProc: ChoiceHintsProc ← NIL,
    changeProc: MultipleChoiceChangeProc ← NIL,
    outlineOrHighlight: OutlineOrHighlight ← highlight];

  MakeTagOnlyItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader,
    visibility: Visibility ← visible];

  MakeTextItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    suffix: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    readOnly: BOOLEAN ← FALSE,
    width: CARDINAL,
    initString: XString.Reader ← NIL,
    wrapUnderTag: BOOLEAN ← FALSE,
    passwordFeedback: BOOLEAN ← FALSE,
    hintsProc: TextHintsProc ← NIL,
    nextOutOfProc: NextOutOfProc ← NIL,
    SPECIALKeyboard: BlackKeys.Keyboard ← NIL];

  MakeWindowItem: PROCEDURE [
    window: Window.Handle,
    myKey: ItemKey,
    tag: XString.Reader ← NIL,
    visibility: Visibility ← visible,
    boxed: BOOLEAN ← TRUE,
    size: Window.Dims,
    nextIntoProc: NextIntoProc ← NIL] RETURNS [clientWindow: Window.Handle];

  DestroyItem: PROCEDURE [window: Window.Handle, item: ItemKey,
    repaint: BOOLEAN ← TRUE];

  DestroyItems: PROCEDURE [
    window: Window.Handle, item: LONG DESCRIPTOR FOR ARRAY OF ItemKey,
    repaint: BOOLEAN ← TRUE];

  -- Getting and setting current values

  GetBooleanItemValue: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [value: BOOLEAN];

  GetChoiceItemValue: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [value: ChoiceIndex];

  GetDecimalItemValue: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [value: XLReal.Number];

  GetIntegerItemValue: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [value: LONG INTEGER];

  GetMultipleChoiceItemValue: PROCEDURE [window: Window.Handle,
    item: ItemKey, zone: UNCOUNTED ZONE]
    RETURNS [value: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex];

  GetTextItemValue: PROCEDURE [
    window: Window.Handle, item: ItemKey, zone: UNCOUNTED ZONE]
    RETURNS [value: XString.ReaderBody];

  LookAtTextItemValue: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [value: XString.ReaderBody];

  DoneLookingAtTextItemValue: PROCEDURE [window: Window.Handle, item: ItemKey];

  GetWindowItemValue: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [value: Window.Handle];

  GetTag: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [tag: XString.ReaderBody];

  -- Setting values

  SetBooleanItemValue: PROCEDURE [
    window: Window.Handle, item: ItemKey, newValue: BOOLEAN,
    repaint: BOOLEAN ← TRUE];

  SetChoiceItemValue: PROCEDURE [
    window: Window.Handle, item: ItemKey, newValue: ChoiceIndex,
    repaint: BOOLEAN ← TRUE];

  SetDecimalItemValue: PROCEDURE [
    window: Window.Handle, item: ItemKey, newValue: XLReal.Number,
    repaint: BOOLEAN ← TRUE];

  SetIntegerItemValue: PROCEDURE [
    window: Window.Handle, item: ItemKey, newValue: LONG INTEGER,
    repaint: BOOLEAN ← TRUE];

  SetMultipleChoiceItemValue: PROCEDURE [
```

```
    window: Window.Handle, item: ItemKey,
    newValues: LONG DESCRIPTOR FOR ARRAY OF ChoiceIndex,
    repaint: BOOLEAN ← TRUE];

  SetTextItemValue: PROCEDURE [
    window: Window.Handle, item: ItemKey, newValue: XString.Reader,
    repaint: BOOLEAN ← TRUE];

  -- Changing display of items

  GetVisibility: PROCEDURE [window: Window.Handle, item: ItemKey]
   RETURNS [visibility: Visibility];

  SetVisibility: PROCEDURE [window: Window.Handle, item: ItemKey, visibility: Visibility, repaint: BOOLEAN ← TRUE];

  -- Layout operations

  SetTabStops: PROCEDURE [window: Window.Handle, tabStops: TabStops];

  GetTabStops: PROC [window: Window.Handle] RETURNS [tabStops: TabStops];

  noTabStop: CARDINAL = CARDINAL.LAST-1;

  defaultTabStops: TabStops = [fixed[interval: 100]];

  LineUpBoxes: PROCEDURE [window: Window.Handle,
      items: LONG DESCRIPTOR FOR ARRAY OF ItemKey ← NIL];

  AppendLine: PROCEDURE [
    window: Window.Handle,
    spaceAboveLine: CARDINAL ← 0]
    RETURNS [line: Line];

  AppendItem: PROCEDURE [
    window: Window.Handle,
    item: ItemKey,
    line: Line,
    preMargin: CARDINAL ← 0,
    tabStop: CARDINAL ← nextTabStop,
    repaint: BOOLEAN ← TRUE];

  InsertLine: PROCEDURE [
    window: Window.Handle,
    before: Line,
    spaceAboveLine: CARDINAL ← 0]
    RETURNS [line: Line];

  InsertItem: PROCEDURE [
    window: Window.Handle,
    item: ItemKey,
    line: Line,
    beforeItem: ItemKey,
    preMargin: CARDINAL ← 0,
    tabStop: CARDINAL ← nextTabStop,
    repaint: BOOLEAN ← TRUE];

  RemoveItemFromLine: PROCEDURE [
    window: Window.Handle,
    item: ItemKey,
    line: Line,
    repaint: BOOLEAN ← TRUE];

  SetItemWidth: PROCEDURE [window: Window.Handle, item: ItemKey,
    width: CARDINAL];

  LayoutInfoFromItem: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [line: Line, margin: CARDINAL, tabStop: CARDINAL, box: Window.Box];

  -- Fixed layout
  SetItemBox: PROCEDURE [window: Window.Handle, item: ItemKey, box: Window.Box];
    -- This is disjoint from the Append/Insert layout procedures. The client may call one or the other, not both!

  -- Miscellaneous item operations

  GetReadOnly: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [readOnly: BOOLEAN];

  GetNextOutOfProc: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [NextOutOfProc];

  GetNextAvailableKey: PROCEDURE [window: Window.Handle]
    RETURNS [key: ItemKey];

  HasAnyBeenChanged: PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN];

  HasBeenChanged: PROCEDURE [window: Window.Handle, item: ItemKey]
    RETURNS [yes: BOOLEAN];

  ResetChanged: PROCEDURE [window: Window.Handle, item: ItemKey];

  ResetAllChanged: PROCEDURE [window: Window.Handle];

  Restore: PROCEDURE [window: Window.Handle];

  Save: PROCEDURE [window: Window.Handle];

  SetChanged: PROCEDURE [window: Window.Handle, item: ItemKey];
```

```
SetAllChanged: PROCEDURE [window: Window.Handle];

SetInputFocus: PROCEDURE [
  window: Window.Handle,
  item: ItemKey,
  beforeChar: CARDINAL ← CARDINAL.LAST];

SetNextOutOfProc: PROCEDURE [window: Window.Handle, item: ItemKey, nextOutOfProc: NextOutOfProc]
  RETURNS [old: NextOutOfProc];

SetReadOnly: PROCEDURE [window: Window.Handle, item: ItemKey, readOnly: BOOLEAN]
  RETURNS [old: BOOLEAN];

SetSelection: PROCEDURE [
  window: Window.Handle,
  item: ItemKey,
  firstChar: CARDINAL ← 0,
  lastChar: CARDINAL ← CARDINAL.LAST];

SetWindowItemSize: PROCEDURE [
  window: Window.Handle, windowItemKey: ItemKey, newSize: Window.Dims];

TakeNEXTKey: PROCEDURE [window: Window.Handle, item: ItemKey];


-- Multinational stuff, i.e. which way text flows.

StreakSuccession: TYPE = SimpleTextDisplay.StreakSuccession;
  -- For individual text and number fields.
  -- Default is fromFirstChar.

Flushness: TYPE = SimpleTextDisplay.Flushness;
  -- For individual text and number fields.
  -- Default is fromFirstChar for text items, flushRight for number items.

GetStreakSuccession: PROCEDURE [window: Window.Handle, item: ItemKey] RETURNS [old: StreakSuccession];

SetStreakSuccession: PROCEDURE [window: Window.Handle, item: ItemKey, new: StreakSuccession] RETURNS [old: StreakSuccession];

GetFlushness: PROCEDURE [window: Window.Handle, item: ItemKey] RETURNS [old: Flushness];

SetFlushness: PROCEDURE [window: Window.Handle, item: ItemKey, new: Flushness] RETURNS [old: Flushness];


END.
```

```
-- File: FormWindowImpl.mesa - last edit
-- JPhillips.es         13-Jul-87  8:54:55
-- guzik.ES             26-Mar-87 11:19:10
-- Dianysian.es         16-Jan-87 10:47:50
-- Mita.es              30-Sep-86 15:31:53
-- Breisacher.es         1-Apr-86 16:11:07
-- SAJohnson.ES         28-Sep-84 14:02:03

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
 --Attention USING [Post],
 Atom USING [ATOM, MakeAtom, null],
 BWSMessages USING [GetMessageHandle],
 BWSMessagesX6 USING [kSetToNeutral],
 Context USING [Create, Data, Destroy, Find, Release, Type],
 BodyWindowParent USING [Create, CreateBody],
 Cursor USING [Set],
 FormWindow,
 FormWindowExtra5,
 FormWindowExtra6,
 FormWindowExtra7,
 FormWindowOps USING [contextFW, ChangeSize, DeallocateSavedItems,
  DeleteItemFromLayout, DeleteItemFromList,
  EnumerateItemsFromContext, EnumerateLaidOutItemsFromContext,
  EnumerateSavedItemsFromContext, FWContext, FWContextObject,
  GetFWContext, InternalTabStops, IsFixed, Item, ItemFromItemKey,
  ItemFromContextAndItemKey, ItemMode, ItemProcsRecord, ItemTypeProcedures, LineInt,
  SaveAnItem, SetChangedBit, TabStops, WindowDestroyedProc],
 Heap USING [Create, Delete],
 MenuData,
 NeverFreeZONE USING [Create],
 PopupMenu,
 Selection USING [Clear, Convert, Free, Value],
 SpecialPropertySheet USING [ExecuteMenuItem],
 StarWindowShell USING [ShellFromChild],
 SubwindowFriends USING [AttachScrollbarsProc, SetSWProcs],
 SubwindowManager USING [TransitionProc],
 TIP USING [ClearInputFocusOnMatch, GetInputFocus, NotifyProc, Results, ResultObject,
  SetTableAndNotifyProc],
 TIPStar USING [NormalTable, SetMode],
 Window USING [Box, Dims, EnumerateInvalidBoxes, GetBox, GetParent, Handle, IntersectBoxes, InvalidateBox, IsDescendantOfRoot,
 IsPlaceInBox, nullBox, Place, SetDisplayProc, SlideAndSize, TrimBoxStickouts, ValidateTree],
 XMessage USING [Get, Handle],
 XString USING [CharacterLength, nullReaderBody, Piece, ReaderBody];

FormWindowImpl: PROGRAM
IMPORTS Atom, BodyWindowParent, BWSMessages, Context, Cursor, FormWindow, FormWindowExtra5, FormWindowOps, Heap, MenuData, NeverFreeZONE,
PopupMenu, Selection, SpecialPropertySheet, StarWindowShell, SubwindowFriends, TIP, TIPStar, Window, XMessage, XString
EXPORTS FormWindow, FormWindowExtra5, FormWindowExtra6, FormWindowExtra7, FormWindowOps =
BEGIN OPEN FormWindow;

--TYPES

-- Globals

 z: UNCOUNTED ZONE ← NeverFreeZONE.Create[initial:2, increment: 1];  --permenant zone, used for itemProcs
 --NOTE: if this zone gets used for anything else, careful attention should be paid to the size of the new nodes.  This is currently
 right on the border line of 2 pages (allowing for the 50+ pages of overhead).  Any growth may necessitate increasing the initial size
 to 3 pages.

  --itemProcs is used to keep track of each item type's procedures.  The client
  --registers his procedures by calling HereAreProcedures
 itemProcs: PUBLIC LONG POINTER TO FormWindowOps.ItemProcsRecord;

 props, stop, menu, nextDown, pointDown, adjustUp, pointUp, pointMotion, adjustMotion,
 moveModeMotion, copyModeMotion, moveModeDown, copyModeDown, moveModeUp, copyModeUp, enter, exit, moveModeEnter, copyModeEnter,
 moveModeExit, copyModeExit: Atom.ATOM ← Atom.null;
 enterResult, newR: TIP.Results ← NIL;

-- Signals and errors

Error: PUBLIC ERROR [code: FormWindow.ErrorCode] = CODE;

LayoutError: PUBLIC SIGNAL [code: FormWindow.LayoutErrorCode] = CODE;

-- Public Procedures

-- Create and Destroy, etc.

Create: PUBLIC PROCEDURE [
  window: Window.Handle,
  makeItemsProc: MakeItemsProc,
  layoutProc: LayoutProc ← FormWindow.DefaultLayout,
  windowChangeProc: GlobalChangeProc ← NIL,
  minDimsChangeProc: MinDimsChangeProc ← NIL,
  zone: UNCOUNTED ZONE ← NIL,
  clientData: LONG POINTER ← NIL] =

BEGIN
  myContext: FormWindowOps.FWContext;
  zoneIsForms: BOOLEAN ← FALSE;

  IF window = NIL THEN RETURN;
  IF zone = NIL THEN {zone ← Heap.Create[initial: 2]; --create its own zone
                      zoneIsForms ← TRUE};
  IF Context.Find[FormWindowOps.contextFW, window] # NIL
```

```
      THEN Error[alreadyAFormWindow];
    --create and initialize context, making the window a formwindow
    myContext ← zone.NEW[FormWindowOps.FWContextObject];
    myContext.tabStops ← NIL;
    myContext.minDimsChangeProc ← minDimsChangeProc;
    myContext.windowChangeProc ← windowChangeProc;
    myContext.zone ← zone;
    myContext.zoneIsForms ← zoneIsForms; --so can tell if should destroy the zone
    myContext.clientData ← clientData;
    Context.Create[FormWindowOps.contextFW, myContext, DestroyContext, window];

    -- the individual make procs fill the items and layout in the context

    IF makeItemsProc # NIL THEN makeItemsProc[window, clientData];
    myContext.inLayoutProc ← TRUE;
    IF layoutProc # NIL THEN layoutProc[window, clientData]
    ELSE FormWindow.DefaultLayout[window, clientData];
    myContext.inLayoutProc ← FALSE;
    IF myContext.layout # NIL THEN MainMeasureProc[window, myContext];
    --set the display and notification procs
    [] ← Window.SetDisplayProc[window, FWDisplayProc];
    IF minDimsChangeProc # NIL THEN
      minDimsChangeProc[window, window.GetBox[].dims, FormWindow.NeededDims[window!
        FormWindow.Error => IF code = incompatibleLayout THEN CONTINUE]];
    TIP.SetTableAndNotifyProc[window, TIPStar.NormalTable[], NotifyProc];
  END; --Create

  Destroy: PUBLIC PROCEDURE [window: Window.Handle] =
    BEGIN
    context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
    zone: UNCOUNTED ZONE ← context.zone;

    -- notify window and TIP of the change
    [] ← Window.SetDisplayProc[window, NIL];
    TIP.SetTableAndNotifyProc[window, NIL, NIL];
    Context.Destroy[FormWindowOps.contextFW, window];
    END; --Destroy

  GetZone: PUBLIC PROCEDURE [
    window: Window.Handle]
    RETURNS [zone: UNCOUNTED ZONE] =
  BEGIN
     context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
     RETURN[context.zone];
  END; --GetZone

  IsIt: PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [yes: BOOLEAN] =
  BEGIN
     context: Context.Data ← Context.Find[FormWindowOps.contextFW, window];
     RETURN[IF context = NIL THEN FALSE ELSE TRUE];
  END; --IsIt

  NumberOfItems: PUBLIC PROCEDURE [
    window: Window.Handle]
   RETURNS [CARDINAL ← 0] =
  BEGIN
    c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
    IF c.keysToHandles = NIL THEN RETURN[0];
    RETURN[c.keysToHandles.curSize-1];
  END; --NumberOfItems

  GetNextAvailableKey: PUBLIC PROCEDURE [
    window: Window.Handle]
   RETURNS [key: FormWindow.ItemKey] =
  BEGIN
    -- really returning the FIRST available key
    c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
    IF c.keysToHandles=NIL OR c.keysToHandles[0]=NIL THEN RETURN[0];
    key ← c.keysToHandles[0].key;
    FOR i: FormWindow.ItemKey IN [1..c.keysToHandles.size) DO
      IF c.keysToHandles[i]=NIL THEN RETURN[i];
      key ← MAX[c.keysToHandles[i].key, key];
    ENDLOOP;
    RETURN[key+1]
  END;  --GetNextAvailableKey

  Repaint: PUBLIC PROCEDURE [window: Window.Handle] =
  BEGIN
    IF window.IsDescendantOfRoot[] THEN Window.ValidateTree [window.GetParent[]];
  END; --Repaint

  DestroyItem: PUBLIC PROCEDURE [
     window: Window.Handle,
     item: ItemKey,
     repaint: BOOLEAN ← TRUE] =
  BEGIN
    ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
    c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];
    itemHandle: FormWindowOps.Item ← FormWindowOps.ItemFromContextAndItemKey[c, item];
    IF itemHandle = NIL THEN RETURN;
    --take it out of the layout
    FormWindowOps.DeleteItemFromLayout[window, c, itemHandle, repaint];
    --take it out of the item list
    FormWindowOps.DeleteItemFromList[c, itemHandle];
    --call the proc
    IF itemProcs[itemHandle.type].destroy # NIL THEN
```

```
      itemProcs[itemHandle.type].destroy[itemHandle, c.zone];
  IF itemHandle = c.inputFocus THEN c.inputFocus ← NIL;
  --get rid of it in keysToHandles and decrement curSize
  c.keysToHandles[item] ← NIL;
  c.keysToHandles.curSize ← c.keysToHandles.curSize - 1;
  Context.Release[FormWindowOps.contextFW, window];
END; --DestroyItem

DestroyItems: PUBLIC PROCEDURE [
  window:Window.Handle,
  item: LONG DESCRIPTOR FOR ARRAY OF ItemKey,
  repaint: BOOLEAN ← TRUE] =
BEGIN
  ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
  c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];
  itemHandle: FormWindowOps.Item;
  FOR ix: CARDINAL IN [0..LENGTH[item]) DO
  { itemHandle ← FormWindowOps.ItemFromContextAndItemKey[c, item[ix]];
    --take it out of the layout
    FormWindowOps.DeleteItemFromLayout[window, c, itemHandle, FALSE];
    --take it out of the item list
    FormWindowOps.DeleteItemFromList[c, itemHandle];
    --call the proc
    IF itemProcs[itemHandle.type].destroy # NIL THEN
        itemProcs[itemHandle.type].destroy[itemHandle, c.zone];
    --get rid of it in keysToHandles and decrement curSize
    c.keysToHandles[item[ix]] ← NIL;
    c.keysToHandles.curSize ← c.keysToHandles.curSize - 1 };
   ENDLOOP;

  -- call measure to reset the place of other items in the window
  IF c.layout # NIL THEN MainMeasureProc[window, c];
  << items being invalidated individually in DeleteItemFromLayout >>
  IF window.IsDescendantOfRoot[] THEN Window.ValidateTree [window.GetParent[]];
  Context.Release[FormWindowOps.contextFW, window];
END; --DestroyItems

GetTag: PUBLIC PROCEDURE [
  window: Window.Handle,
  item: ItemKey]
  RETURNS [tag: XString.ReaderBody ← XString.nullReaderBody] =
BEGIN
 context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
 itemHandle: FormWindowOps.Item ← FormWindowOps.ItemFromContextAndItemKey[context, item];
 IF itemHandle # NIL THEN RETURN[itemHandle.tag];
END; --GetTag

-- Changing display of items

GetVisibility: PUBLIC PROCEDURE [
  window: Window.Handle,
  item: FormWindow.ItemKey]
 RETURNS [visibility: FormWindow.Visibility] =
BEGIN
  i: FormWindowOps.Item = FormWindowOps.ItemFromItemKey[window, item];
  IF i # NIL THEN RETURN[visibility ← i.visibility]
END; --GetVisibility

SetVisibility: PUBLIC PROCEDURE [
  window: Window.Handle,
  item: ItemKey,
  visibility: FormWindow.Visibility,
  repaint: BOOLEAN ← TRUE] =
BEGIN
  ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
  c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];
  i: FormWindowOps.Item = FormWindowOps.ItemFromContextAndItemKey[c, item];
  IF i = NIL THEN RETURN;
BEGIN
  windowDims: Window.Dims ← Window.GetBox[window].dims;
  oldHeight: INTEGER ← i.box.dims.h;
  boxToInvlidate: Window.Box;
  i.visibility ← visibility;
  IF c.layout # NIL AND i.line # NIL THEN
    { -- change the data structures and call the client
    MeasureLine[window, i.line, c];
    -- $$$ KLUDGE AR 00000: The reapaint passed in ChangeSize is a kludge!
    --  In the case of window items, MeasureLine may insert the item into the
    -- tree which will casue a validation before FormWindowOps.ChangeSize
    --  will have a chance to do a shift.  By passing repaint: FALSE,
    -- we will do a Window.Invalidate instead of shift.
    FormWindowOps.ChangeSize[window: window, item: i,
      oldHt: oldHeight, newHt: i.box.dims.h,
      repaint: IF i.type = window THEN FALSE ELSE repaint]};

  IF itemProcs[i.type].changeVisibility # NIL THEN
    itemProcs[i.type].changeVisibility[window, c, i, visibility];

  --invalidate everything on that 'line'
  --Use windowDims.w instead of the new width, because the extra area would have
  --been invalidated in the SlideAndSize
  boxToInvlidate.place ← [0,
    i.boxWithTags.place.y - (IF i.line # NIL THEN i.line.spaceAboveLine ELSE 0)];
  boxToInvlidate.dims.w ← windowDims.w;
  boxToInvlidate.dims.h ← IF i.line # NIL THEN
      MAX[i.line.height, i.boxWithTags.dims.h] + i.line.spaceAboveLine
    ELSE i.boxWithTags.dims.h;
```

```
    Window.InvalidateBox[window, boxToInvlidate];
    IF window.IsDescendantOfRoot[] AND repaint AND ~c.inLayoutProc THEN Window.ValidateTree[window.GetParent[]];
    Context.Release[FormWindowOps.contextFW, window];
  END
  END;  -- SetVisibility

  GetBoxed: PUBLIC PROCEDURE [
    window: Window.Handle,
    item: FormWindow.ItemKey]
   RETURNS [boxed: BOOLEAN] =
   BEGIN
    i: FormWindowOps.Item = FormWindowOps.ItemFromItemKey[window, item];
    IF i # NIL THEN RETURN[boxed ← i.boxed]
  END; --GetBoxed


  <<The problem with the SetBoxed procedure:

          Items are not used to growing side ways!

    So it is not really "finished" and has not been exported anywhere.
  >>
  SetBoxed: <<PUBLIC>> PROCEDURE [
    window: Window.Handle,
    item: ItemKey,
    boxed: BOOLEAN,
    repaint: BOOLEAN ← TRUE] =
  BEGIN
    ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
    c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];
    i: FormWindowOps.Item = FormWindowOps.ItemFromContextAndItemKey[c, item];
    IF i = NIL THEN RETURN;
  BEGIN
    windowDims: Window.Dims ← Window.GetBox[window].dims;
    IF c.layout # NIL AND i.line # NIL THEN
    { -- figure out which item to start remeasuring with
      itemToMeasureFrom: FormWindowOps.Item ← ItemToStartMeasure[i];
      -- change the data structures
      i.boxed ← boxed;
      MainMeasureProc[window, c, itemToMeasureFrom] }
    ELSE i.boxed ← boxed;

    IF c.minDimsChangeProc # NIL THEN
        {neededDims: Window.Dims ← FormWindow.NeededDims[window];
        IF neededDims # windowDims THEN c.minDimsChangeProc[window, windowDims, [neededDims.w, neededDims.h]]};
    --invalidate everything from that 'line' on
    Window.InvalidateBox[
      window: window,
      box: [place: [0, i.boxWithTags.place.y],
            dims: [w: windowDims.w, h: 30000]] ];
    IF window.IsDescendantOfRoot[] AND repaint THEN Window.ValidateTree[window.GetParent[]];
    Context.Release[FormWindowOps.contextFW, window];
  END
  END;  -- SetBoxed

  SetReadOnly: PUBLIC PROCEDURE [
    window: Window.Handle,
    item: FormWindow.ItemKey,
    readOnly: BOOLEAN]
    RETURNS [old: BOOLEAN] =
  BEGIN
    ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
    c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];
    i: FormWindowOps.Item = FormWindowOps.ItemFromContextAndItemKey[c, item];
    IF itemProcs[i.type].changeReadOnly # NIL THEN
      itemProcs[i.type].changeReadOnly[window, c, i, readOnly];
    old ← i.readOnly;
    i.readOnly ← readOnly;
    Context.Release[FormWindowOps.contextFW, window];
  END; --SetReadOnly

  -- Miscellaneous item operations
  HasAnyBeenChanged: PUBLIC PROCEDURE [window: Window.Handle]
   RETURNS [yes: BOOLEAN] =
  BEGIN
   context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
   RETURN[context.changedAny];
  END; --HasAnyBeenChanged

  HasBeenChanged: PUBLIC PROCEDURE [
    window: Window.Handle,
    item: ItemKey]
    RETURNS [yes: BOOLEAN ← FALSE] =
  BEGIN
   context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
   itemHandle: FormWindowOps.Item ← FormWindowOps.ItemFromContextAndItemKey[context, item];
   IF itemHandle # NIL THEN RETURN[itemHandle.changed];
  END; --HasBeenChanged

  ResetChanged: PUBLIC PROCEDURE [window: Window.Handle, item: ItemKey] =
    {FormWindowOps.SetChangedBit[window, item, FALSE]};

  ResetAllChanged: PUBLIC PROCEDURE [window: Window.Handle] =
  BEGIN
    ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
    context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];
```

```
    ItemProc: PROCEDURE [item: FormWindowOps.Item] RETURNS [stop: BOOLEAN ← FALSE] =
     {item.changed ← FALSE};

    FormWindowOps.EnumerateItemsFromContext[context, ItemProc];
    -- now set the global changed bit
    context.changedAny ← FALSE;
    Context.Release[FormWindowOps.contextFW, window];
  END; --ResetAllChanged

  <<Saving and Restoring Items.  The client specifies when he wants to
    save and restore items.  A snapshot of the itemKey and value of each
    item (except for Window items) is taken on a Save.  A bit is then
    set specifying that a save was done.  A Restore can be done if a
    Save was previously executed.  Restore sets all of the items using
    the internal set procedures.>>

  Restore: PUBLIC PROCEDURE [window: Window.Handle] =
  BEGIN
    ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
   c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];

   CallSet: PROC [
     key: FormWindow.ItemKey,
     value: LONG POINTER ← NIL,
     length: CARDINAL ← 0,
     isNeutral: BOOLEAN ← FALSE]
      RETURNS [stop: BOOLEAN ← FALSE] =
   BEGIN
    BEGIN ENABLE FormWindow.Error => IF code = invalidItemKey THEN CONTINUE;
    item: FormWindowOps.Item ← FormWindowOps.ItemFromContextAndItemKey[c, key];
    newMode: FormWindowOps.ItemMode ← IF isNeutral THEN neutral ELSE normal;
    oldMode: FormWindowOps.ItemMode ← IF item.neutral THEN neutral ELSE normal;
    IF itemProcs[item.type].internalSetValue # NIL
     THEN itemProcs[item.type].internalSetValue[window, item, value];
    IF isNeutral OR (newMode # oldMode) THEN { -- we need to change modes
      IF itemProcs[item.type].changeMode # NIL THEN
        itemProcs[item.type].changeMode[
         window, c, item, oldMode, newMode, TRUE]};
    END; --enable
    END; --CallSet

   IF ~c.hasBeenSaved THEN RETURN; --nothing previously saved
   <<should an error be raised instead of just returning?>>

   <<Restore those guys!!>>
   FormWindowOps.EnumerateSavedItemsFromContext[c, CallSet];

   --call the global change proc
   IF c.windowChangeProc # NIL THEN
     c.windowChangeProc[window, FormWindow.nullItemKey, restore, c.clientData];

   --remeasure,invalidate and repaint
   IF c.layout # NIL THEN MainMeasureProc[window, c];
   Window.InvalidateBox [window, [ [0,0], [30000, 30000] ] ];
   IF window.IsDescendantOfRoot[] THEN Window.ValidateTree [window.GetParent[]];
   Context.Release[FormWindowOps.contextFW, window];
  END; --Restore

  Save: PUBLIC PROCEDURE [window: Window.Handle] =
  BEGIN
   c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
   destroy: PROC [window: Window.Handle, v: LONG POINTER];

   CallSaveInternal: PROC [item: FormWindowOps.Item]
        RETURNS [stop: BOOLEAN ← FALSE] =
    BEGIN
     value: LONG POINTER ← NIL;
     length: CARDINAL ← 0;
     IF itemProcs[item.type].internalGetValue # NIL THEN
        {[value, length, destroy] ← itemProcs[item.type].internalGetValue[window, item];
         IF length # 0 THEN
           FormWindowOps.SaveAnItem[c, item, value, length];
         IF destroy # NIL THEN destroy[window, value];
         -- if length = 0, don't bother saving it--};
    END; --CallInternal

   IF c.hasBeenSaved THEN FormWindowOps.DeallocateSavedItems[c];
   FormWindowOps.EnumerateLaidOutItemsFromContext[c, CallSaveInternal];
   c.hasBeenSaved ← TRUE;
  END; --Save

  SetChanged: PUBLIC PROCEDURE [
    window: Window.Handle,
    item: ItemKey] =
  {FormWindowOps.SetChangedBit[window, item]};

  SetAllChanged: PUBLIC PROCEDURE [window: Window.Handle] =
  BEGIN
    ENABLE UNWIND => Context.Release[FormWindowOps.contextFW, window];
   context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window, TRUE];

   ItemProc: PROCEDURE [item: FormWindowOps.Item] RETURNS [stop: BOOLEAN ← FALSE] =
    {item.changed ← TRUE};

   FormWindowOps.EnumerateItemsFromContext[context, ItemProc];
   -- now set the global changed bit
   context.changedAny ← TRUE;
```

```
 Context.Release[FormWindowOps.contextFW, window];
END; --SetAllChanged

HereAreProcedures: PUBLIC PROC [
    type: FormWindow.ItemType,
    procs: FormWindowOps.ItemTypeProcedures] =
BEGIN
  itemProcs[type] ← procs;
END; --HereAreProcedures

TakeNEXTKey: PUBLIC PROCEDURE [
  window: Window.Handle,
  item: FormWindow.ItemKey] =
BEGIN
 c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
 i: FormWindowOps.Item ← FormWindowOps.ItemFromItemKey [window: window, itemKey: item];
 --find the next key to send it to
 []←ProcessNextKey[window, c, i];
END; --TakeNEXTKey

-- private procedures

DestroyContext: PROCEDURE [
  myContext: Context.Data,
  window: Window.Handle] =
BEGIN
 fwContext: FormWindowOps.FWContext ← LOOPHOLE[myContext];
 zone: UNCOUNTED ZONE ← fwContext.zone;
 IF fwContext.notifyDestroy # NIL THEN fwContext.notifyDestroy[fwContext, window];
 --destroy each item
 FOR item: FormWindowOps.Item ← fwContext.items, item.next
   UNTIL item = NIL DO
   --ask the item whether it wants to destroy anything
   IF itemProcs[item.type].destroy # NIL
   THEN itemProcs[item.type].destroy[item, zone];
 ENDLOOP;
 FOR type: FormWindow.ItemType IN [choice..last) DO
   IF itemProcs[type].windowDestroyed # NIL
   THEN itemProcs[type].windowDestroyed[fwContext, window];
 ENDLOOP;
 --need to destroy the layout stuff whenever we figure it out
 IF ~fwContext.zoneIsForms THEN
   {FreeLayout[fwContext];
   zone.FREE[@fwContext.keysToHandles];
   IF fwContext.hasBeenSaved THEN FormWindowOps.DeallocateSavedItems[fwContext];
   zone.FREE[@fwContext.tabStops];
   zone.FREE[@fwContext]}
 ELSE Heap.Delete[zone];
END; --DestroyContext

<<This proc should really reside in FormWindowLayoutImpl>>
FreeLayout: PROC [fwContext: FormWindowOps.FWContext] =
{IF fwContext.layout # NIL THEN
   {currentLine: FormWindowOps.LineInt ← fwContext.layout;
    nextLine: FormWindowOps.LineInt;
    WHILE currentLine #  NIL DO
      nextLine ← currentLine.nextLine;
      fwContext.zone.FREE[@currentLine];
      currentLine ← nextLine;
      ENDLOOP;
    fwContext.layout ← NIL};
 };


ItemToStartMeasure: PUBLIC PROCEDURE [i: FormWindowOps.Item,
  beforeItem: BOOLEAN ← FALSE ]
 RETURNS [itemToMeasureFrom: FormWindowOps.Item ← NIL] =
BEGIN
   -- figure out which item to start remeasuring with
    IF i.visibility = invisible OR beforeItem THEN
    --need to establish a non-bogus place to start
    { FOR prev: FormWindowOps.Item ← i.prevItemOnLine, prev.prevItemOnLine
       UNTIL prev = NIL  -- beginning of the line -- DO
          IF prev.visibility # invisible THEN
          { itemToMeasureFrom ← prev; RETURN };
      ENDLOOP;
      FOR l: FormWindowOps.LineInt ← i.line.prevLine, l.prevLine
       UNTIL (l = NIL) DO
        IF l.items = NIL THEN LOOP;
        IF l.items.visibility # invisible THEN
        { itemToMeasureFrom ← l.items; RETURN }
      ENDLOOP }
    ELSE itemToMeasureFrom ← i;  -- remeasure from this item
END;  --ItemToStartMeasure

MainMeasureProc: PUBLIC PROCEDURE [
  window: Window.Handle,
  c: FormWindowOps.FWContext,
  beginItem: FormWindowOps.Item ← NIL] =
BEGIN
<< This measureProc will measure from the beginning of the line of the item passed in. This will handle getting a starting box.place that
has the right values (including tabs and lineHeight) established for the line.  The item.boxWithTags.place Some later version might want
to do this some other way than redoing from the beginning of the line - though I'm not convinced of the worth of such a savings.

This proc is never called for fixed layout.  Fixed layout item place and dims are set at the time of the calls to SetItemBox by
individual calls to the itemMeasureProcs.
>>
```

```
  -- set place where place.x is set at the beginning of the new line and
  -- place.y at the bottom of the previous line (or top of the form)
  place: Window.Place ←
    IF beginItem = NIL THEN [0,0]  -- measuring from the beginning of the items list
    ELSE [0, beginItem.boxWithTags.place.y - beginItem.line.spaceAboveLine];

  IF c.inLayoutProc OR c.layout = NIL --fixed layout-- THEN RETURN;

  FOR l: FormWindowOps.LineInt ← (IF beginItem # NIL THEN beginItem.line ELSE c.layout), l.nextLine
    UNTIL l = NIL DO
  {lineHeight: CARDINAL ← 0;
   place.y ← place.y + l.spaceAboveLine;
   FOR item: FormWindowOps.Item ← l.items, item.nextItemOnLine
    UNTIL item = NIL DO
     --set the item.box.place
     IF item.visibility = invisible THEN LOOP;
     place.x ← Findx[c.tabStops, item, place.x];
     item.boxWithTags.place ← place;
     IF itemProcs[item.type].measure # NIL THEN
       itemProcs[item.type].measure[window, c, item];
     -- set l.height here by finding tallest item
     lineHeight ← MAX[lineHeight, item.boxWithTags.dims.h];
     place.x ← place.x + item.boxWithTags.dims.w;
   ENDLOOP;
   l.height ← lineHeight;
   place.x ← 0;
   IF l.height = 0 <<no visible items on line>>
   THEN place.y ← place.y - l.spaceAboveLine
   ELSE place.y ← place.y + l.height};
 ENDLOOP;
END; --MainMeasureProc

<<This procedure is for moving items on one line horizontally.  Its called when
  an item changes visibility.  It is meant to call before calling FormWindowOps.ChangeSize.  This porc does NOT set the line.height, that
  will be done in ChangeSize.
>>
MeasureLine:  PROCEDURE[
  window: Window.Handle,
  line: FormWindowOps.LineInt,
  c: FormWindowOps.FWContext] =
  BEGIN
  place: Window.Place ← [0,0];
  <<Find place.y for the items on the line>>
  IF c.inLayoutProc OR c.layout = NIL --fixed layout-- THEN RETURN;
  FOR l: FormWindowOps.LineInt ← c.layout, l.nextLine
    UNTIL l = line DO
    place.y ← place.y + l.height;
    IF l.height # 0 THEN place.y ← place.y + l.spaceAboveLine;
    ENDLOOP;
  place.y ← place.y + line.spaceAboveLine;
  FOR item: FormWindowOps.Item ← line.items, item.nextItemOnLine
   UNTIL item = NIL DO
    IF item.visibility = invisible THEN LOOP;
    place.x ← Findx[c.tabStops, item, place.x];
    item.boxWithTags.place  ← place;
    IF itemProcs[item.type].measure # NIL THEN
      itemProcs[item.type].measure[window, c, item];
    place.x ← place.x + item.boxWithTags.dims.w;
  ENDLOOP;
  END;

<<This proc cuts stringLabel to desiredWidth size.  It uses simple linear
 extrapolation>>
MeasureLable: PUBLIC PROCEDURE [
  stringLabel: XString.ReaderBody,
  desiredWidth, actualWidth: CARDINAL]
  RETURNS [measuredLabel: XString.ReaderBody] =
{desiredCharWidth: CARDINAL ←
   (XString.CharacterLength[@stringLabel]*desiredWidth)/actualWidth;
 [measuredLabel, ] ← XString.Piece[@stringLabel, 0, desiredCharWidth];
};--MeasureLable

Findx: PROCEDURE[
  tabStops: FormWindowOps.TabStops,
  item: FormWindowOps.Item,
  placex: INTEGER]
 RETURNS [INTEGER] =
BEGIN
  IF tabStops # NIL AND item.tabStop # FormWindow.noTabStop THEN
   SELECT item.tabStop FROM
    = FormWindow.nextTabStop => --next available tab
      WITH tab: tabStops  SELECT FROM
        fixed => IF tab.interval # 0 AND placex # 0 THEN
          placex ← ((placex/tab.interval)*tab.interval)+tab.interval;
        vary  => IF tab.size # 0 THEN
        { foundTab: BOOLEAN ← FALSE;
          i: CARDINAL;
          cardinalX: CARDINAL;
          IF placex >= 0 THEN cardinalX ← placex ELSE ERROR;
          FOR i IN [0..tab.size) DO
            IF tab[i] < cardinalX THEN LOOP
            ELSE {foundTab ← TRUE; EXIT};
          ENDLOOP;
          IF foundTab THEN placex ← tab[i]
          ELSE LayoutError[notEnufTabsDefined] };
      ENDCASE;
```

```
    # FormWindow.nextTabStop => --specified tab
      WITH tab: tabStops  SELECT FROM
        fixed => placex ← item.tabStop*tab.interval;
        vary  => placex ← tab[item.tabStop];
      ENDCASE;
  ENDCASE;
  RETURN[placex + item.preMargin];
END;  -- Findx

ShiftItemPlaces: PUBLIC PROC [item: FormWindowOps.Item, dif: INTEGER ] =
BEGIN
 oldPlace: Window.Place;
 IF item = NIL THEN RETURN;
 FOR line: FormWindowOps.LineInt ← item.line, line.nextLine UNTIL line = NIL DO
   FOR i: FormWindowOps.Item ← line.items, i.nextItemOnLine
    UNTIL i = NIL DO
     i.boxWithTags.place.y ← i.boxWithTags.place.y + dif;
     oldPlace ← i.box.place;
     i.box.place.y ← i.box.place.y + dif;
     IF itemProcs[i.type].shift # NIL THEN
      itemProcs[i.type].shift[i, oldPlace, i.box.place];
   ENDLOOP
 ENDLOOP
END;  --ShiftItemPlaces

NeutralPopupProc: PUBLIC FormWindowExtra5.PopupProc =
BEGIN
  z: UNCOUNTED ZONE ← FormWindow.GetZone [window];
  msgHandle: XMessage.Handle ← BWSMessages.GetMessageHandle [];
  setToNeutral: XString.ReaderBody ← XMessage.Get [msgHandle, BWSMessagesX6.kSetToNeutral];
  menuItem: MenuData.ItemHandle;
  menuArray: ARRAY [0..1) OF MenuData.ItemHandle;
  fwItem: FormWindowOps.Item ← FormWindowOps.ItemFromItemKey [window, item];

  IF fwItem.readOnly OR (fwItem.type = command OR fwItem.type = tagonly) THEN
    {menu ← NIL; freeProc ← NIL; RETURN};

  menuItem ← MenuData.CreateItem [
    zone: z,
    name: @setToNeutral,
    proc: MyPopupProc,
    itemData: item];
  menuArray ← [menuItem];
  menu ← MenuData.CreateMenu [z, menuItem, DESCRIPTOR[menuArray], TRUE];
  freeProc ← NeutralFreeProc;
  MenuData.DestroyItem [z, menuItem];
END;

NeutralFreeProc: FormWindowExtra5.MenuFreeProc =
BEGIN
  z: UNCOUNTED ZONE ← FormWindow.GetZone [window];
  MenuData.DestroyMenu [z, menuHandle]
END;

MyPopupProc: MenuData.MenuProc =
BEGIN ENABLE FormWindowExtra5.ItemError => CONTINUE;
  FormWindowExtra5.SetItemNeutralness [
    window: window,
    item: CARDINAL[itemData],
    neutral: TRUE,
    repaint: TRUE]
END;

FWDisplayProc: PROCEDURE [window: Window.Handle] =
BEGIN
   c: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
   dims: Window.Dims ← Window.GetBox[window].dims;
   upperLeft: Window.Place ← [INTEGER.LAST, INTEGER.LAST];
   lowerRight: Window.Place ← [0, 0];
   boxToPaint: Window.Box ← Window.nullBox;

   EachInvalidBox: PROCEDURE [w: Window.Handle, box: Window.Box] = {
          upperLeft.x ← MIN [upperLeft.x, box.place.x];
          upperLeft.y ← MIN [upperLeft.y, box.place.y];
          lowerRight.x ← MAX [lowerRight.x, box.place.x + box.dims.w];
          lowerRight.y ← MAX [lowerRight.y, box.place.y + box.dims.h];
          };

   Window.EnumerateInvalidBoxes [window, EachInvalidBox];
   boxToPaint ← [upperLeft, [lowerRight.x-upperLeft.x, lowerRight.y-upperLeft.y]];
   boxToPaint ← Window.TrimBoxStickouts [window, boxToPaint];

   IF c.layout = NIL --fixed layout-- THEN
      FOR i: FormWindowOps.Item ← c.items, i.next UNTIL i = NIL DO
        IF (i.visibility # visible) OR (i.boxWithTags = Window.nullBox) OR
          Window.IntersectBoxes[i.boxWithTags, boxToPaint].dims = [0,0] THEN LOOP;       IF itemProcs[i.type].display # NIL
        THEN itemProcs[i.type].display[i, window];
      ENDLOOP
    ELSE
      FOR l: FormWindowOps.LineInt ← c.layout, l.nextLine
        UNTIL l = NIL DO
        FOR item: FormWindowOps.Item ← l.items, item.nextItemOnLine
          UNTIL item = NIL DO
          IF itemProcs[item.type].display # NIL AND item.visibility = visible AND
            Window.IntersectBoxes[item.boxWithTags, boxToPaint].dims # [0,0]  THEN
            itemProcs[item.type].display[item, window];
        ENDLOOP;
```

```
      ENDLOOP;
END; --FWDisplayProc

NotifyProc: TIP.NotifyProc =
  BEGIN
  noPlace: Window.Place = [-1,-1];
  context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
  place: Window.Place ← noPlace;
  item, itemIncludingTag: FormWindowOps.Item ← NIL;
  oldChanged: BOOLEAN ← FALSE;
  windowDestroyed: BOOLEAN ← FALSE;
  FOR input: TIP.Results ← results, input.next UNTIL ¬nput = NIL DO
    WITH z: input SELECT FROM
      coords => place ← z.place;
      atom =>
        BEGIN
        IF place = noPlace THEN item ← context.inputFocus
        ELSE [item, itemIncludingTag] ← ResolveToItem[place, context];
        IF item # NIL THEN
          IF item.readOnly THEN
            SELECT item.type FROM
            -- text, decimal and integer are selectable but not editable when readOnly
              text, decimal,integer => NULL;
            -- other items not selectable OR editable when readOnly
            ENDCASE => SELECT z.a FROM
              pointDown =>
              -- let readOnly choice items through on the pointDown
              -- in case the popup is being pointed to
              -- choice will NOT change but the user can see the list of options
              -- especially useful to user if the choice is notFullyDisplayed
                IF item.type=choice AND itemProcs[choice].tipResults # NIL THEN {
                  []←itemProcs[choice].tipResults[item, window, results];
                  RETURN }
                ELSE RETURN;
              ENDCASE => RETURN;
        SELECT z.a FROM
          -- hitting the props key when the formWindow (inside a psheet ONLY)
          -- has the input focus will cause the psheet to close
          -- a NoOp if the fw is not in a shell or the shell is not a psheet
          props => {
           ChildHasFocus: PROC[window, inputFocus: Window.Handle]
            RETURNS[BOOLEAN ← FALSE] = {
             FOR w:Window.Handle ← inputFocus, w.GetParent[]
              UNTIL w=NIL DO { IF w = window THEN RETURN[TRUE] }
             ENDLOOP };
           IF context.inputFocus # NIL
             OR ChildHasFocus[window, TIP.GetInputFocus[]] THEN {
            SpecialPropertySheet.ExecuteMenuItem[
              StarWindowShell.ShellFromChild[window], done];
            RETURN }};
          stop => [] ← TIPStar.SetMode[normal];
          nextDown => windowDestroyed ← ProcessNextKey[window, context, item];
          enter, copyModeEnter, moveModeEnter => RETURN;
          exit, copyModeExit, moveModeExit => {
            IF context.trackingItem # NIL THEN {  -- we exited the window directly from an item
              SendAnExitAtom[context.trackingItem, place, context.zone, window, z.a];
              context.trackingItem ← NIL};
            RETURN};
          pointMotion, adjustMotion,
          moveModeMotion, copyModeMotion =>
            IF item # context.trackingItem THEN
              BEGIN
              -- We've transitioned across items, so send an Exit to the old one and add an Enter to the new one's results.
              IF context.trackingItem # NIL THEN
                SendAnExitAtom[context.trackingItem, place, context.zone, window, z.a];
              IF item # NIL THEN
                results ← PrefixAnEnterAtom[results, place, z.a]
                -- we will attach an enter atom only if we are actually entering an item.
              END;
          menu => {
            IF itemIncludingTag = NIL OR itemIncludingTag.readOnly THEN RETURN;
            -- action is over white area or item is read only.

            IF context.popupProc # NIL THEN {
              menuHandle: MenuData.MenuHandle;
              freeProc: FormWindowExtra5.MenuFreeProc;
              [menuHandle, freeProc] ←
                context.popupProc [window, itemIncludingTag.key];
              oldChanged ← itemIncludingTag.changed;
              itemIncludingTag.changed ← FALSE;
              IF menuHandle # NIL THEN {
                PopupMenu.Popup [menuHandle, window, FALSE];
                freeProc [window, menuHandle]};
              IF itemIncludingTag.changed THEN {
                context.changedAny ← TRUE;
                IF context.windowChangeProc # NIL THEN
                   context.windowChangeProc[window,
                   itemIncludingTag.key, user, FormWindow.GetClientData[window]]};
              itemIncludingTag.changed ← oldChanged OR itemIncludingTag.changed;
              RETURN};
            LOOP};
          ENDCASE;
        IF windowDestroyed THEN RETURN;
        context.trackingItem ← item;
        IF item = NIL THEN --action over white area.
          {SELECT z.a FROM
            moveModeMotion, copyModeMotion, moveModeDown, copyModeDown =>
```

```
            Cursor.Set[questionMark];
          moveModeUp, copyModeUp => [] ← TIPStar.SetMode[normal];
          ENDCASE;
        RETURN};
      IF itemProcs[item.type].tipResults # NIL THEN {
        tookFocus: BOOLEAN ← FALSE;
        hadFocus: FormWindowOps.Item ← context.inputFocus;
        GoneAway: FormWindowOps.WindowDestroyedProc = {
          <<[fwContext: FWContext, window: Window.Handle]>>
          IF fwContext = context THEN windowDestroyed ← TRUE };
        SetGoneAway: PROC = INLINE { context.notifyDestroy ← GoneAway };
        ClearGoneAway: PROC = INLINE { context notifyDestroy ← NIL };
        SetGoneAway[];
        oldChanged ← item.changed;
        item.changed ← FALSE;
        IF itemProcs[item.type].tipResults[item, window, results] THEN
          tookFocus ← TRUE;
        IF windowDestroyed THEN RETURN;
        ClearGoneAway[];
        IF tookFocus THEN {
          context.inputFocus ← item;
          IF hadFocus # context.inputFocus
            AND context.tookFocusProc # NIL THEN
              context.tookFocusProc[window, context.inputFocus.key];
          };
        }
      ELSE SELECT z.a FROM
            moveModeUp, copyModeUp => [] ← TIPStar.SetMode[normal];
            moveModeDown, copyModeDown => Cursor.Set[questionMark];
            ENDCASE;
      IF item.changed THEN
         {context.changedAny ← TRUE;
          IF context.windowChangeProc # NIL THEN
              context.windowChangeProc[window, item.key, user, FormWindow.GetClientData[window]]};
      item.changed ← oldChanged OR item.changed;
      SELECT z.a FROM
        adjustUp, pointUp,
        copyModeUp, moveModeUp => context.trackingItem ← NIL;
        ENDCASE;--we're tracking items only between mousebutton down and up
      EXIT;
      END; -- it is an atom
    string => {
      item ← context.inputFocus;
      IF item = NIL THEN RETURN;
      oldChanged ← item.changed;
      item.changed ← FALSE;
      IF itemProcs[item.type].tipResults # NIL THEN
        IF itemProcs[item.type].tipResults[item, window, results] THEN
          context.inputFocus ← item;
      IF item.changed THEN
        {context.changedAny ← TRUE;
         IF context.windowChangeProc # NIL THEN
           context.windowChangeProc[window, item.key, user, FormWindow.GetClientData[window]]};
      item.changed ← oldChanged OR item.changed;
      EXIT; };
    ENDCASE;
  ENDLOOP;
END;

ProcessNextKey: PROC [
  window: Window.Handle,
  context: FormWindowOps.FWContext,
  item: FormWindowOps.Item]
 RETURNS [windowDestroyed: BOOLEAN ← FALSE]  =
  BEGIN
    <<CheckNext is used for finding the next item to send the next key to>>
   CheckNext: PROC [nextItem: FormWindowOps.Item]
      RETURNS [stop: BOOLEAN ← FALSE] =
   BEGIN
    -- skip the first one (this is the current input focus)
    IF item = nextItem THEN RETURN;
    -- don't process if nextItem is invisible or readOnly
    IF nextItem # NIL THEN IF nextItem.visibility # visible
      OR nextItem.readOnly THEN RETURN;

    -- if the client called SetInputFocus, then we want to stop here and
    -- not call the next takeNextProc.
    IF context.clientSetInputFocus THEN RETURN [stop: TRUE];

    IF itemProcs[nextItem.type].takeNextProc # NIL
      THEN stop ← itemProcs[nextItem.type].takeNextProc[window, nextItem];
    IF stop THEN context.inputFocus ← nextItem;

   END; --CheckNext

   --START here
   IF item = NIL THEN RETURN;

   <<If the current item has a nextOutOfProc, call it>>
   IF itemProcs[item.type].nextOutOfProc # NIL THEN
   { -- in case the window goes away - like nexting out of password field in
     -- logon sheet - be prepared to bail out.
     windowGone: BOOLEAN ← FALSE;
     GoneAway: FormWindowOps.WindowDestroyedProc =
      << [fwContext: FWContext, window: Window.Handle] >>
      { IF fwContext = context THEN windowGone ← TRUE };
     SetGoneAway: PROC = INLINE { context.notifyDestroy ← GoneAway };
```

```
      ClearGoneAway: PROC = INLINE { context.notifyDestroy ← NIL };
      SetGoneAway[];
      -- we initialize the clientSetInputFocus boolean here so that
      -- it can be checked inside CheckNext. This lets us know whether the
      -- client wants to set the IF himself during his NextOutOfProc, rather
      -- then letting us do it.
      context.clientSetInputFocus ← FALSE;
      itemProcs[item.type].nextOutOfProc[window, item];
      IF windowGone THEN RETURN[TRUE] ELSE ClearGoneAway[] };

    <<find the next item with a non-nil takeNextProc.
     Call it.  If it returns yes then EXIT, else find the next
     item with a non-nil proc...>>

     FormWindowOps.EnumerateLaidOutItemsFromContext[context, CheckNext, item.key];
    END; --ProcessNextKey


  PrefixAnEnterAtom: PROC [results: TIP.Results, place: Window.Place, mode: Atom.ATOM] RETURNS   [newResults: TIP.Results] =
    BEGIN
    <<enterResult and newR are globals which are reused every time this proc is
    called.  In the Init proc we have already set newR.next ← enterResult.>>
    WITH y: enterResult SELECT FROM
           atom => y.a ←  SELECT mode FROM
                                    moveModeMotion,
                                    moveModeEnter => moveModeEnter,
                                    copyModeMotion,
                                    copyModeEnter => copyModeEnter,
                               ENDCASE => enter;
           ENDCASE;
    enterResult.next ← results;
    newR.body ← coords[place: place];
    RETURN [newR];
    END;

  ResolveToItem: PROC [
    place: Window.Place,
    context: FormWindowOps.FWContext]
    RETURNS [item, itemIncludingTag: FormWindowOps.Item ← NIL] =
    BEGIN

    CheckItem: PROC [i: FormWindowOps.Item] RETURNS [stop: BOOLEAN ← FALSE] = {
      IF Window.IsPlaceInBox[place, i.box] THEN {
        IF i.visibility = visible THEN {
           item ← itemIncludingTag ← i;
           RETURN [stop: TRUE]}}
      ELSE IF Window.IsPlaceInBox[place, i.boxWithTags] THEN {
        IF i.visibility = visible THEN {
          itemIncludingTag ← i;
          RETURN [stop: TRUE]}};
      RETURN [stop: FALSE]; };

    IF FormWindowOps.IsFixed[context]
     THEN FormWindowOps.EnumerateItemsFromContext[context, CheckItem]
    ELSE FormWindowOps.EnumerateLaidOutItemsFromContext[context, CheckItem];
    END;

  SendAnExitAtom: PROC [
    item: FormWindowOps.Item,
    place: Window.Place,
    z: UNCOUNTED ZONE,
    window: Window.Handle,
    mode: Atom.ATOM] =
    BEGIN
    exitResult: TIP.Results ← z.NEW[TIP.ResultObject ← [
      body: atom[a: SELECT mode FROM
                      moveModeMotion,
                      moveModeExit => moveModeExit,
                      copyModeMotion,
                      copyModeExit => copyModeExit,
                      ENDCASE => exit],
      next: NIL]];
    results: TIP.Results ← z.NEW[TIP.ResultObject ← [
      body: coords[place: place],
      next: exitResult]];
    IF itemProcs[item.type].tipResults # NIL THEN
      [] ← itemProcs[item.type].tipResults[item, window, results];
    z.FREE[@results];
    z.FREE[@exitResult];
    END;

    SetTookFocusProc: PUBLIC PROCEDURE [window: Window.Handle,
      proc: FormWindowExtra6.TookFocusProc] = {
      context: FormWindowOps.FWContext ← FormWindowOps.GetFWContext[window];
      context.tookFocusProc ← proc;
      };

    TransitionProc: SubwindowManager.TransitionProc = {
      IF window=NIL THEN RETURN;
      SELECT state FROM
        sleeping, dead => {
          selectedWindow: Selection.Value ← Selection.Convert[target: window];
          TIP.ClearInputFocusOnMatch[window];
          IF selectedWindow.value = window THEN Selection.Clear[];
          Selection.Free[@selectedWindow] };
        ENDCASE };
```

```
  DefaultMinDimsChangeProc:  PUBLIC FormWindow.MinDimsChangeProc = {
    Window.SlideAndSize[window, [Window.GetBox[window].place, new]];
    <<Window.ValidateTree[Window.GetParent[window]];>>
   };

  SetUpFormSW: SubwindowFriends.AttachScrollbarsProc = {
    BodyWindowParent.Create[parent: subwindow, verticalScrollbar: vertScrollbar,
      horizontalScrollbar: horizScrollbar, zone: zone];
    sw ← BodyWindowParent.CreateBody[subwindow, Window.nullBox, NIL, NIL];
   };

  InitAtoms: PROC = {
    props ← Atom.MakeAtom["PropsDown"L];
    stop ← Atom.MakeAtom["Stop"L];
    menu ← Atom.MakeAtom["Menu"L];
    nextDown ← Atom.MakeAtom["NextDown"L];
    pointDown ← Atom.MakeAtom["PointDown"L];
    adjustUp ← Atom.MakeAtom["AdjustUp"L];
    pointUp ← Atom.MakeAtom["PointUp"L];
    pointMotion ← Atom.MakeAtom["PointMotion"L];
    adjustMotion ← Atom.MakeAtom["AdjustMotion"L];
    moveModeMotion ← Atom.MakeAtom["MoveModeMotion"L];
    copyModeMotion ← Atom.MakeAtom["CopyModeMotion"L];
    moveModeDown ← Atom.MakeAtom["MoveModeDown"L];
    copyModeDown ← Atom.MakeAtom["CopyModeDown"L];
    moveModeUp ← Atom.MakeAtom["MoveModeUp"L];
    copyModeUp ← Atom.MakeAtom["CopyModeUp"L];
    enter ← Atom.MakeAtom["Enter"L];
    exit ← Atom.MakeAtom["Exit"L];
    moveModeEnter ← Atom.MakeAtom["MoveModeEnter"L];
    copyModeEnter ← Atom.MakeAtom["CopyModeEnter"L];
    moveModeExit ← Atom.MakeAtom["MoveModeExit"L];
    copyModeExit ← Atom.MakeAtom["CopyModeExit"L];
  };

  Init: PROC ={
    InitAtoms[];
    --must increase this allocation if any new items are added.
    itemProcs ←
      z.NEW[FormWindowOps.ItemProcsRecord[FormWindow.ItemType.last.ORD]];
    FOR i: FormWindow.ItemType IN
      [FIRST[FormWindow.ItemType]..LAST[FormWindow.ItemType]) DO
      itemProcs[i] ← [NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL,NIL];
      ENDLOOP;
    --prepare enter TIP.Results
    enterResult ← z.NEW[atom TIP.ResultObject ← [
      next: NIL,
      body: atom[a: enter]]];
    newR ← z.NEW[TIP.ResultObject ← [
      next: enterResult,
      body: coords[place: [0,0]]]];
    -- initialize the subwindow type "form"
    SubwindowFriends.SetSWProcs[
      type:form,
      attachScrollbarsProc: SetUpFormSW,
      transitionProc:TransitionProc];
  };


  --Main program
  Init[];

END.
 February 17, 1984 - SAJohnson.es - Call global change proc when something changes  his isn't quite correct yet.)
 February 21, 1984 - SAJohnson.es  - Changed references to systemFontHeight to the , ocedure SystemFontHeight. Moved in the first item on
 a line 5 bits.
 March 8, 1984 - SAJohnson.es - Fixed problem in DisplayTag when no tag exists and this is after the first time around for the display.
 March 13, 1984 - SAJohnson.es - In Create, if layoutproc is NIL then use default layout proc.
 March 19, 1984 - SAJohnson.es - Added DisplaySuffix.
 March 26, 1984 - SAJohnson.es - Added Next key processing.
 March 29, 1984 - SAJohnson.es - keysToHandles additions.
 4-Apr-84 17:37:27 - SAJohnson.es - Implemented Save and Restore.
 May 2, 1984 - JPhillips.es - Implemented TabStops.
 May 8, 1984 - SAJohnson.es - In Save, call destroy proc which client passed back from InternalGet in order to allow client to deallocate
 storage. (Currently used by text and number items.)
 May 10, 1984 - SAJohnson.es - Implemented DestroyItem Procs.
 May 15, 1984 - SAJohnson.es - 2.0 defs changes
 May 31, 1984 - SAJohnson.es - CauseRepaint in SetVisibility. Implemented TakeNextKey.
 June 1, 1984 - SAJohnson.es - ProcessNextKey - clear the input focus and selection.
 June 1, 1984 - JPhillips.es - Implemented NumberOfItems.
 June 4, 1984 - JPhillips.es - handle repaint boolean and implement Repaint.
 June 5, 1984 - JPhillips.es - move SetInputFocus and SetSelection to TextAndNumberItemsImpl.  move Display procs to
 FormWindowDisplayImpl.
 June 11, 84 - JPhillips.es - updated to 2.0b defs.  Remove StringIntoWindow Public Proc.
 July 12, 84 - JPhillips.es - implement readOnliness
 July 16, 84 - JPhillips.es - add changeVisibility
 July 17, 84 - JPhillips.es - change Save/Restore algorithm to Save/Restore/Restore.
 July 18, 84 - JPhillips.es - add Invalidate when ~repaint in SetVisibility.
 July 19, 84 - JPhillips.es - don't send next key notification to an invisible item!
 July 23, 84 - JPhillips.es - fixed bug where tabStop not defaulted to nextTab when using fixed tabs.
 July 25, 1984 - SAJohnson.es - Added fixed layout to DisplayProc.
 July 26, 1984 - SAJohnson.es - In Restore, call globalChangeProc.
 July 27, 1984 - SAJohnson.es - Check for visibility in ResolveToItem.
 Aug 20, 1984 - SAJohnson.es - In SetVisibility, make sure only the portion of the window that needs repainting is repainted.
 Aug 27, 1984 - SAJohnson.es - In Create, save clientData in context.
 Aug 30, 1984 - SAJohnson.es - Invalidate more in SetVisibility.
```

Sept 25, 1984 - JPhillips.es - BlinkDisplay on pointDown only.
Sept 26, 1984 - SAJohnson.es - Changed all calls to Validate to calls to ValidateTree so all children are repainted.
Sept 27, 1984 - JPhillips.es - In FindX - get rid of all the (IF placex↑ = 0 THEN placex↑ = 5) instances that forced a small margin at the left edge of the FW.
Sept 28, 1984 - SAJohnson.es - In SetVisibility, don't do so much invalidating.
Nov 20, 1984 - JPhillips.es - Add MainMeasureProc and new FWDisplayProc. IN DestroyItem, DestroyItems - add call to MainMeasureProc, Invalidate and ValidateTree. Add call to MainMeasureProc in SetVisibility, Restore.
Dec 5, 1984 - JPhillips.es - 4.0 defs changes.
Dec 14, 1984 - JPhillips.es - return value for ProcessNextKey so that logon can close the window when nexting out of password field.
Dec 17, 1984 - JPhillips.es - do not call takeNext if item is readOnly.
Jan 25 85 - JPhillips - tweek in MainMeasure for line height w/ no visible items
Jan 28, 85 - JPhillips - subtract out the spaceAboveLine if all items on line invisible. (AR #12772)
12-Feb-85 -Dianysian -context.trackingItem is non NIL only while either mouse buttons is down.
16-Feb-85 - JPhillips - check for windowDestroyed by client during itemChangeProc.
27-Mar-85 - Dianysian - Allocating the itemProcs from the heap.
9-May-85 - Breisacher - Changed ProcessNextKey call to EnumerateLaidOutItemsFromContext back to start at item.key rather than context.inputFocus.key. ARs 14693 14797.
7-Jun-85 - Dianysian - enterResult and newR are global, and are reused.
3-Jul-85 - Dianysian - added step => TIPStar.SetMode[normal];
9-Aug-85 - Dianysian - copy move to background taken care of.
9-Aug-85 - Dianysian - PrefixAnEnterAtom adds different flavors of enter
17-Sep-85 - Dianysian - call FreeLayout, free the context.
1-Apr-86 - Breisacher - SetTookFocusProc stuff - AR 6406.
21-MAY-86 - jphillips - Put UNWINDs in where the context is locked. Add Get/SetBoxed.
5-Jun-86 - Dianysian - SetVisibility on an item not in the layout no more crashes
10-Jun-86 - Dianysian - Added i.line # NIL to SetBoxed.
18-Jun-86 - Dianysian - GlobalChangeProc gets called only in real changes(AR6476)
20-Jun-86 - jphillips - Create permanent zone with largeNodeThreshold big enough for the 423 word node that is allocated for itemProcs (total = 442 words in 3 nodes). (AR8026)
23 June 86 - jphillips - well gee whiz...itemProcs initialization = last.ORD all this time. Changing the heap initialization caused an address fault that should have happened long ago. Woops. Leave initial size as is and loop through itemProcs [FIRST .. LAST) in init and destroy.
30-Sep-86 - Mita - Change to NeverFreeZONE. Still use Heap for normal FormWindow.
7-Oct-86 - jphillips - initialize subwindow type "form".
21-Oct-86 - jphillips - add a DefaultMinDimsChangeProc and TransitionProc. Call the minDimsChangeProc at the end of Create.
27-Oct-86 - jp - sleeping and dead should be treated the same in TransitionProc.
17-Nov-86 - guzik - Changed NotifyProc to accomodate a menu atom (chording). Also changed ResolveToItem to accomodate choriding in the item tag.
21-Nov-86 - jphillips - get rid of blankSpace added to neededDims. Do SetUpFormSW.
24-Nov-86 - guzik - Added call to changeMode proc in Restore to accomodate neutral props. Also changed parameters to CallSet, EnumerateSavedItemsFromContext, and SaveAnItem.
12-Dec-86 - Dianysian - NeutralPopupProc is a Nop if readOly, command item or tagonly item. Also, in Create swaped Window.SetDisplayProc & minDimsChangeProc.
16-Dec-86 - jphillips - Don't do the ValidateTree in the DefaultMinDimsChangeProc. Let the guy that called it do it if and when appropriate. ValidateTree should be done on the form window's parent anywhere that we might have caused a SlideAndSize to be done on the form window.
18-Dec-86 - Dianysian - Fixed MeasureLine to ignore empty lines.
29-Dec-86 - Dianysian - Fixed calling windowChangeProc in menu arm of NotifyProc.
1-Jan-87 - jp - check for window=NIL in TransitionProc. (I could have sworn this got fixed before.)
16-Jan-87 - Dianyisan - Put special case for windowitems in SetVisibility
3-Mar-87 - jphillips - lots of little bug fixes:
AR 10664 Special case readOnly choice items in NotifyProc.
AR 10313 GetNextAvailableKey should return first available key. Also the loop in there should use size NOT curSize.
AR 10809 NumberOfItems should return curSize not curSize-1.
AR 10314 DestroyItem(s) should decrement curSize.
AR 3802 Close psheet if props hit when the formWindow has the inputFocus. (If the formWindow is not inside a psheet it will be a noop.)
4-Mar-87 14:39:00 - guzik - RE: AR 8999 - Changed ItemFromContextAndItemKey[c, ix] to ItemFromContextAndItemKey[c, item[ix]] and c.keysToHandles[ix] to c.keysToHandles[item[ix]] in DestroyItems 4-Mar-87 15:09:11 - guzik - Removed Selection.Clear and ClearInputFocusOnMatch in CheckNext just before calling takeNextProc. It should be the responsibility of the individual TakeNextProcs to determine the disposition of the selection and input focus for the FormWindow. AR10813
5-Mar-87 11:07:43 - guzik - We check to see if the client called SetInputFocus from within his/her NextOutOfProc. If so, then we don't reset it during ProcessNextKey. AR 10373.
5-Mar-87 - jphillips - recapture a lost edit from 1 Jan (really 5 Jan): Check for window in tree before doing ValidateTree in Repaint.
6-Mar-87 - jphillips - put the same check into DestroyItems, SetVisibility, SetBoxed. Get rid of CauseRepaint because it is only being called from one place now. Put the appropriate code into Restore in place of the call to CauseRepaint.
11-Mar-87 - jphillips - for the moment back out the fix to NumberOfItems cuz it causes Remote Printing to crash. (See 3-Mar-87 AR 10809)
26-Mar-87 11:13:20 - guzik - Initialized oldChanged to FALSE in NotifyProc. AR 11422
12-May-87 - JPhillips - Our part of AR 11470. Close on "props" if a child of the form window has the inputFocus.
1-Jul-87 - JPhillips - AR 13080. check to see if the FormWindow has the inputFocus (but not in an item).
13-Jul-87 - JPhillips - Removed the erroneous EXIT from the ChildHasFocus loop.

```
-- File: FormWindowMessageParseImpl.mesa - last edit:
-- guzik:OSBU South:Xerox 25-Nov-86  9:47:44
-- Dianysian.es        19-Jun-85  8:45:31
-- Breisacher.ES        4-Dec-84 18:52:38

-- Copyright (C) 1985, 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  FormWindow,
  FormWindowMessageParse,
  XCharSet0,
  XChar,
  XString;

FormWindowMessageParseImpl: PROGRAM
  IMPORTS XChar, XCharSet0, XString
  EXPORTS FormWindowMessageParse = BEGIN

  -- TYPEs

  ChoiceItemSeq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF FormWindow.ChoiceItem];

  ChoiceItemSeqHandle: TYPE = LONG POINTER TO ChoiceItemSeq;

  -- Global data and constants

  itemSeparator: XString.Character ← XCharSet0.Make [commercialAt];
  numberSeparator: XString.Character ← XCharSet0.Make [colon];
  wrapIndicator: XString.Character ← XCharSet0.Make [verticalBar];

  -- Public procedures

  ParseChoiceItemMessage: PUBLIC PROC [
    choiceItemMessage: XString.Reader,
    zone: UNCOUNTED ZONE]
    RETURNS [choiceItems: FormWindow.ChoiceItems] = {
    nItems: CARDINAL ← NumberOfItems [choiceItemMessage];
    choiceItemsSeq: ChoiceItemSeqHandle ← zone.NEW [ChoiceItemSeq [nItems]];
    message: XString.ReaderBody ← choiceItemMessage↑;
    FOR i: CARDINAL IN [0..nItems) DO
      choiceItemsSeq [i] ← GetNextItem [@message, zone];
      ENDLOOP;
    RETURN [DESCRIPTOR [choiceItemsSeq, nItems]];
    };

  FreeChoiceItems: PUBLIC PROC [
    choiceItems: FormWindow.ChoiceItems,
    zone: UNCOUNTED ZONE] = {
    lp: ChoiceItemSeqHandle ← LOOPHOLE [BASE[choiceItems]];
    zone.FREE [@lp];
    };

  -- Private procedures

  GetNextItem: PROC [message: XString.Reader, zone: UNCOUNTED ZONE] RETURNS [choiceItem: FormWindow.ChoiceItem] = {

    LookingFor: TYPE = {number, stringOrWrap};

    lookingFor: LookingFor ← stringOrWrap;
    front: XString.ReaderBody;
    choiceString: XString.ReaderBody;
    choiceNumber: FormWindow.ChoiceIndex ← LAST[FormWindow.ChoiceIndex];
    breakChar: XString.Character;

    breakTable: XString.BreakTableObject ← [
      otherSets: not,
      set: 0]; -- default codes to ALL[not]
    breakTable.codes [XChar.Code [itemSeparator]] ← stop;
    breakTable.codes [XChar.Code [numberSeparator]] ← stop;
    breakTable.codes [XChar.Code [wrapIndicator]] ← stop;

    UNTIL XString.Empty [message] DO
      [front: front, breakChar: breakChar] ← XString.Scan [
        r: message,
        break: @breakTable,
        option: ignore];

      SELECT breakChar FROM
        numberSeparator => {
          IF lookingFor # stringOrWrap THEN ERROR;
          lookingFor ← number;
          choiceString ← front; };
        itemSeparator,
        XChar.not => IF lookingFor = number THEN {
          choiceNumber ← CARDINAL[XString.ReaderToNumber [@front<< !
            XString.Overflow,
            XString.InvalidNumber => -- Raise an error here>>]];
          lookingFor ← stringOrWrap;
          RETURN [[string[choiceNumber, choiceString]]]; };
        wrapIndicator => {
          IF lookingFor # stringOrWrap THEN ERROR;
          RETURN [[wrapIndicator[]]]; };
        ENDCASE => ERROR;
      ENDLOOP;

    };
```

```
  NumberOfItems: PROC [message: XString.Reader] RETURNS [nItems: CARDINAL] = {
    -- Number of items = number of "@" and "|" characters.
    rb: XString.ReaderBody ← message↑;

    breakTable: XString.BreakTableObject ← [
      otherSets: not,
      set: 0]; -- default codes to ALL[not]
    breakTable.codes [XChar.Code [itemSeparator]] ← stop;
    breakTable.codes [XChar.Code [wrapIndicator]] ← stop;

    nItems ← 0;
    UNTIL XString.Empty [@rb] DO
      [] ← XString.Scan [
        r: @rb,
        break: @breakTable,
        option: ignore];
      nItems ← nItems + 1;
      ENDLOOP;
    };

  -- Mainline code

  END...

log:

 4-Dec-84 - Breisacher.es - change for 4.0 defs change to XString.Scan.
 25-Nov-86  - guzik - changed GetNextItem so that ReaderToNumber does not produce warnings.
```

```
--NSPrint.mesa - (Last modified by AOF on  3-Jun-83  9:26:51)
--Mesa interface to Printing protocol.

--Copyright (C) Xerox Corporation 1982. All rights reserved.

DIRECTORY
  Courier USING [ErrorCode],
  NSDataStream USING [Source],
  NSString USING [String],
  System USING [NetworkAddress, UniversalID];

NSPrint: DEFINITIONS =
  BEGIN

  --TYPES
  Time: TYPE = LONG CARDINAL;
  String: TYPE = NSString.String;

  RequestID: TYPE = System.UniversalID;
  SystemElement: TYPE = System.NetworkAddress;

  Media: TYPE = LONG DESCRIPTOR FOR ARRAY OF Medium;
  Medium: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): MediumType FROM
      paper => [paper(1): Paper],
      ENDCASE];
  MediumType: TYPE = MACHINE DEPENDENT {paper(0)};
  MediumIndex: TYPE = CARDINAL[0..1);
  Paper: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): PaperType FROM
      unknown => [],  --illegal argument, possible result
      knownSize => [knownSize(1): PaperSize],
      otherSize => [otherSize(1): PaperDimensions],
      ENDCASE];
  PaperType: TYPE = MACHINE DEPENDENT {unknown(0), knownSize, otherSize(2)};
  PaperIndex: TYPE = CARDINAL[0..3);
  PaperSize: TYPE = MACHINE DEPENDENT {
    dontUse(0) -- the protocol defines this enumeration as starting at 1! --,
    usLetter, usLegal, a0, a1, a2, a3, a4, a5, a6, a7, a8, a9,
    isoB0, isoB1, isoB2, isoB3, isoB4, isoB5, isoB6, isoB7, isoB8, isoB9, isoB10,
    jisB0, jisB1, jisB2, jisB3, jisB4, jisB5, jisB6, jisB7, jisB8, jisB9,
    jisB10(34)};
  PaperDimensions: TYPE = MACHINE DEPENDENT RECORD [
    length(0), width(1): CARDINAL];  --units are millimeters

  PrintAttributes: TYPE = LONG DESCRIPTOR FOR ARRAY OF PrintAttribute;
  PrintAttribute: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): PrintAttributeType FROM
      printObjectName => [printObjectName(1): String ← [NIL, 0, 0]],
      printObjectCreateDate => [printObjectCreateDate(1): Time ← 0],
      senderName => [senderName(1): String ← [NIL, 0, 0]],
      ENDCASE];
  PrintAttributeType: TYPE = MACHINE DEPENDENT {
    printObjectName(0), printObjectCreateDate, senderName(2)};
  PrintAttributesIndex: TYPE = CARDINAL[0..3);

  PrintOptions: TYPE = LONG DESCRIPTOR FOR ARRAY OF PrintOption;
  PrintOption: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): PrintOptionType FROM
      printObjectSize => [printObjectSize(1): LONG CARDINAL ← 0],
      recipientName => [recipientName(1): String ← [NIL, 0, 0]],
      message => [message(1): String ← [NIL, 0, 0]],
      copyCount => [copyCount(1): CARDINAL ← 1],
      pagesToPrint => [pagesToPrint(1): PagesToPrint ← [1, LAST[CARDINAL]]],
      mediumHint => [mediumHint(1): Medium ← [paper[[knownSize[usLetter]]]]],
      priorityHint => [priorityHint(1): PriorityHint ← normal],
      releaseKey => [releaseKey(1): CARDINAL ← LAST[CARDINAL]],
      staple => [staple(1): BOOLEAN ← FALSE],
      twoSided => [twoSided(1): BOOLEAN ← FALSE],
      ENDCASE];
  PrintOptionType: TYPE = MACHINE DEPENDENT {
    printObjectSize(0), recipientName, message, copyCount, pagesToPrint,
    mediumHint, priorityHint, releaseKey, staple, twoSided(9)};
  PrintOptionsIndex: TYPE = CARDINAL[0..10);
  PagesToPrint: TYPE = MACHINE DEPENDENT RECORD [
    beginningPageNumber(0), endingPageNumber(1): CARDINAL];
```

```
  PriorityHint: TYPE = MACHINE DEPENDENT {low(0), normal, high(2)};

  PrinterProperties: TYPE = LONG DESCRIPTOR FOR ARRAY OF PrinterProperty;
  PrinterProperty: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): PrinterPropertyType FROM
      media => [media(1): Media],
      staple => [staple(1): BOOLEAN],
      twoSided => [twoSided(1): BOOLEAN],
      ENDCASE];
  PrinterPropertyType: TYPE = MACHINE DEPENDENT {
    media(0), staple, twoSided(2)};
  PrinterPropertiesIndex: TYPE = CARDINAL[0..3);

  PrinterStatus: TYPE = LONG DESCRIPTOR FOR ARRAY OF PrinterStatusComponent;
  PrinterStatusComponent: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): PrinterStatusType FROM
      spooler => [spooler(1): Spooler],
      formatter => [formatter(1): Formatter],
      printer => [printer(1): Printer],
      media => [media(1): Media],
      ENDCASE];
  PrinterStatusType: TYPE = MACHINE DEPENDENT {
    spooler(0), formatter, printer, media(3)};
  PrinterStatusIndex: TYPE = CARDINAL[0..4);
  Spooler: TYPE = MACHINE DEPENDENT {available(0), busy, disabled, full(3)};
  Formatter: TYPE = MACHINE DEPENDENT {available(0), busy, disabled(2)};
  Printer: TYPE = MACHINE DEPENDENT {
    available(0), busy, disabled, needsAttention, needsKeyOperator(4)};

  RequestStatus: TYPE = LONG DESCRIPTOR FOR ARRAY OF RequestStatusComponent;
  RequestStatusComponent: TYPE = MACHINE DEPENDENT RECORD [
    var(0): SELECT type(0): RequestStatusType FROM
      status => [status(1): Status],
      statusMessage => [statusMessage(1): String],
      ENDCASE];
  RequestStatusType: TYPE = MACHINE DEPENDENT {status(0), statusMessage(1)};
  RequestStatusIndex: TYPE = CARDINAL[0..2);
  Status: TYPE = MACHINE DEPENDENT {
    pending(0), inProgress, completed, completedWithWarnings, unknown, rejected,
    aborted, canceled, held(8)};

  ConnectionProblem: TYPE = MACHINE DEPENDENT {
    noRoute(0), noResponse, transmissionHardware, transportTimeout,
    tooManyLocalConnections, tooManyRemoteConnections,
    missingCourier, missingProgram, missingProcedure, protocolMismatch,
    parameterInconsistency, invalidMessage, returnTimedOut(12)
    --otherCallProblem(LAST[CARDINAL])--};

  ErrorType: TYPE = MACHINE DEPENDENT {
    busy(0), insufficientSpoolSpace, invalidPrintParameters, masterTooLarge,
    mediumUnavailable, serviceUnavailable, spoolingDisabled, spoolingQueueFull,
    systemError, tooManyClients, undefinedError, connectionError, transferError(12), courier};

  TransferProblem: TYPE = MACHINE DEPENDENT {
    aborted(0), formatIncorrect(2), noRendezvous, wrongDirection(4)};

  UndefinedProblem: TYPE = CARDINAL;

  --ERRORS
  Error: ERROR [why: ErrorRecord];
  ErrorRecord: TYPE = RECORD [
    SELECT errorType: ErrorType FROM
      busy, insufficientSpoolSpace, invalidPrintParameters, masterTooLarge,
      mediumUnavailable, serviceUnavailable, spoolingDisabled, spoolingQueueFull,
      systemError, tooManyClients => [],
      undefinedError => [undefined: UndefinedProblem],
      transferError => [transfer: TransferProblem],
      connectionError => [connection: ConnectionProblem],
      courier => [courier: Courier.ErrorCode],
      ENDCASE];

  --PROCEDURE MODELS
  Print: PROCEDURE [
    master: NSDataStream.Source,
    printAttributes: PrintAttributes,
    printOptions: PrintOptions,
```

```
    systemElement: SystemElement]
    RETURNS [printRequestID: RequestID];

  GetPrinterProperties: PROCEDURE [systemElement: SystemElement]
    RETURNS [properties: PrinterProperties];

  GetPrinterStatus: PROCEDURE [systemElement: SystemElement]
    RETURNS [status: PrinterStatus];

  GetPrintRequestStatus: PROCEDURE [
    printRequestID: RequestID, systemElement: SystemElement]
    RETURNS [status: RequestStatus];

  FreeString: PROCEDURE [string: LONG POINTER TO String];
  FreeMedia: PROCEDURE [media: LONG POINTER TO Media];
  FreePrinterProperties: PROCEDURE [printerProperties: LONG POINTER TO PrinterProperties];
  FreePrinterStatus: PROCEDURE [printerStatus: LONG POINTER TO PrinterStatus];
  FreeRequestStatus: PROCEDURE [requestStatus: LONG POINTER TO RequestStatus];


  END.

LOG
 7-Sep-82 14:35:56 - AOF - Created file.
 8-Sep-82 13:43:32 - AOF - Cleanup parameter names, error codes, etc.
 8-Sep-82 23:27:27 - Alfvin - Brought in line with protocol spec, NSFile.
13-Sep-82  9:41:42 - AOF - Fix bugs.
28-Oct-82 13:55:07 - Beeley - Fixed PaperSize range, adding "dontUse(0)"; deleted Formatter[full].
 3-Jun-83  9:26:15 - AOF - Correct defaults for PrintOptions.pagesToPrint and String.
```

```
-- PressNSPrintImpl.mesa - edited by:
--  Johnsson     26-May-83  8:38:17
--  Poskanzer    11-Apr-85 19:19:22

-- Copyright (C) 1983, 1985 by Xerox Corporation.  All rights reserved.

DIRECTORY
  AddressTranslation USING [Error, PrintError, StringToNetworkAddress],
  Courier USING [ErrorCode],
  Heap USING [systemZone],
  Interpress82Maker USING [],
  MFile USING [CopyFileHandle, Handle],
  MSegment USING [Address, Create, Delete, Handle],
  MStream USING [Copy],
  NSDataStream USING [Aborted, SinkStream],
  NSString USING [AppendToMesaString],
  PressPrint USING [Handle, Object, TroubleCode],
  Runtime USING [IsBound],
  StatusWindow USING [Register, zone],
  Stream USING [Delete, Handle, SetPosition],
  String USING [
    AppendChar, AppendCharAndGrow, AppendString, AppendStringAndGrow,
    CopyToNewString, Length, StringBoundsFault],
  System USING [GreenwichMeanTime],
  NSPrint USING [
    Error, ErrorRecord, FreePrinterStatus, FreeRequestStatus, GetPrintRequestStatus, GetPrinterStatus,
Print,
    PrintAttribute, PrinterStatus, PrintOption, RequestID, RequestStatus, String,
    SystemElement];

PressNSPrintImpl: PROGRAM
  IMPORTS
    AddressTranslation, Heap, MFile, MSegment, MStream, NSDataStream,
    NSPrint, NSString, Runtime, StatusWindow, Stream, String
  EXPORTS Interpress82Maker =
  BEGIN

  object: PressPrint.Object ← [
    Trouble: Trouble,
    GetStatus: GetStatus,
    IsPressFile: IsPressFile,
    SendPressStream: SendPressStream,
    Delete: Delete];

  SPO: TYPE = LONG POINTER TO StatusProcObject;
  StatusProcObject: TYPE = RECORD [
    reqID: NSPrint.RequestID, se: NSPrint.SystemElement];


  CreatePrinter: PUBLIC PROCEDURE RETURNS [h: PressPrint.Handle] =
    BEGIN
    RETURN[@object];
    END;

  Delete: PROCEDURE [PressPrint.Handle] = {};

  Trouble: SIGNAL [code: PressPrint.TroubleCode, message: LONG STRING ← NIL] = CODE;

  IsPressFile: PROCEDURE [fh: MFile.Handle] RETURNS [isPressFile: BOOLEAN] =
    BEGIN
    header: STRING = "Interpress/Xerox"L;
    seg: MSegment.Handle;
    p: LONG POINTER TO PACKED ARRAY OF CHARACTER;
    seg ← MSegment.Create[MFile.CopyFileHandle[fh, [], readOnly], [], 0, 1];
    p ← MSegment.Address[seg];
    FOR i: CARDINAL IN [0..header.length) DO
      IF header[i] # p[i] THEN {isPressFile ← FALSE; EXIT};
      REPEAT FINISHED => isPressFile ← TRUE;
      ENDLOOP;
    MSegment.Delete[seg];
    RETURN
    END;

  MakeNSString: PROCEDURE [s: LONG STRING] RETURNS [NSPrint.String] = {
    RETURN[IF s = NIL THEN [NIL, 0, 0]
    ELSE [LOOPHOLE[@s.text], s.length, s.maxlength]]};
```

```
    SendPressStream: PROCEDURE [
      stream: Stream.Handle, bytes: LONG CARDINAL, host: LONG STRING,
      copies: CARDINAL ← 1, sides: [0..2] ← 0,
      fileName: LONG STRING ← NIL, userName: LONG STRING ← NIL,
      date: System.GreenwichMeanTime] =
      BEGIN

      z: UNCOUNTED ZONE = Heap.systemZone;
      spo: SPO ← z.NEW[StatusProcObject];

      BEGIN ENABLE UNWIND => z.FREE[@spo];
      SendStream: PROCEDURE [sink: NSDataStream.SinkStream] = {
        [] ← MStream.Copy[from: stream, to: sink, bytes: LAST[LONG CARDINAL] !
          NSDataStream.Aborted => CONTINUE;
          UNWIND => Stream.Delete[sink ! NSDataStream.Aborted => CONTINUE]];
        Stream.Delete[sink ! NSDataStream.Aborted => CONTINUE]};
      hostAddr: NSPrint.SystemElement = GetAddress[host];
      attributes: ARRAY [0..3) OF NSPrint.PrintAttribute ← [
        [printObjectName[MakeNSString[fileName]]],
        [printObjectCreateDate[date]],
        [senderName[MakeNSString[userName]]]];
      options: ARRAY [0..4) OF NSPrint.PrintOption ← [
        [printObjectSize[bytes]],
        [recipientName[MakeNSString[userName]]],
        [copyCount[copies]],
        [twoSided[sides = 2]]];
      spo.reqID ← NSPrint.Print[
        [proc[SendStream]], DESCRIPTOR[attributes], DESCRIPTOR[options], hostAddr !
          NSPrint.Error => SELECT why.errorType FROM
            busy, tooManyClients => {SIGNAL Trouble[busy];
                                     Stream.SetPosition[stream,0];
                                     RETRY};
            spoolingDisabled => {SIGNAL Trouble[busy, "spooling disabled"L];
                                        Stream.SetPosition[stream,0];
                                        RETRY};
            ENDCASE => ReportNSPrintError[why]];

      spo.se ← hostAddr;
      IF Runtime.IsBound[LOOPHOLE[StatusWindow.Register]] THEN {
        name: LONG STRING ← z.NEW[
          StringBody[String.Length[fileName] + String.Length[host] + 4]];
        IF fileName # NIL THEN String.AppendString[name, fileName];
        String.AppendString[name, " on "L];
        IF host # NIL THEN String.AppendString[name, host];
        StatusWindow.Register[name, StatusProc, spo];
        z.FREE[@name]}
      ELSE z.FREE[@spo];

      END;
      END;

    StatusProc: PROCEDURE [h: SPO, reject: BOOLEAN]
      RETURNS [done: BOOLEAN ← FALSE, message: LONG STRING ← NIL] = {
      IF Runtime.IsBound[LOOPHOLE[StatusWindow.Register]] THEN {
         swZone: UNCOUNTED ZONE = StatusWindow.zone;
         status: NSPrint.RequestStatus;
         IF reject THEN {x: SPO ← h; Heap.systemZone.FREE[@x]; RETURN[TRUE, NIL]};
         message ← swZone.NEW[StringBody[60]];
         status ← NSPrint.GetPrintRequestStatus[h.reqID, h.se ! NSPrint.Error => {
            AppendNSPrintError[@message, why, swZone]; GOTO Error}];
            FOR i: CARDINAL IN [0.. LENGTH[status]) UNTIL done DO
               ENABLE String.StringBoundsFault => EXIT;
               IF i # 0 THEN String.AppendChar[message, ' ];
               WITH s: status[i] SELECT FROM
                  status => {
                     String.AppendString[
                        message, SELECT s.status FROM
                     pending => "pending"L, inProgress => "in progress"L,
                     completed => "completed"L, held => "held"L,
                     completedWithWarnings => "completed with warnings"L,
                     unknown => "unknown"L, rejected => "rejected"L,
                     aborted => "aborted"L, canceled => "canceled"L, ENDCASE => "?"L];
                               done ← (s.status # pending) AND (s.status # inProgress)
                              AND (s.status # held)};
                statusMessage => NSString.AppendToMesaString[message,
```

```
                                s.statusMessage];
            ENDCASE;
          ENDLOOP;
        IF done THEN {x: SPO ← h; Heap.systemZone.FREE[@x]};
        NSPrint.FreeRequestStatus[@status];};
      EXITS Error => NULL};

  ReportNSPrintError: PROCEDURE [e: NSPrint.ErrorRecord] =
    BEGIN
    z: UNCOUNTED ZONE = Heap.systemZone;
    s: LONG STRING ← z.NEW[StringBody[100]];
    AppendNSPrintError[@s, e, z];
    ERROR Trouble[other, s ! UNWIND => z.FREE[@s]];
    END;

  AppendNSPrintError: PROCEDURE [s: LONG POINTER TO LONG STRING, e: NSPrint.ErrorRecord, z: UNCOUNTED
ZONE] =
    BEGIN
    String.AppendStringAndGrow[s, "NSPrint.Error["L, z];
    WITH e SELECT FROM
      busy => String.AppendStringAndGrow[s, "busy"L, z];
      insufficientSpoolSpace => String.AppendStringAndGrow[s, "insufficientSpoolSpace"L, z];
      invalidPrintParameters => String.AppendStringAndGrow[s, "invalidPrintParameters"L, z];
      masterTooLarge => String.AppendStringAndGrow[s, "masterTooLarge"L, z];
      mediumUnavailable => String.AppendStringAndGrow[s, "mediumUnavailable"L, z];
      serviceUnavailable => String.AppendStringAndGrow[s, "serviceUnavailable"L, z];
      spoolingDisabled => String.AppendStringAndGrow[s, "spoolingDisabled"L, z];
      spoolingQueueFull => String.AppendStringAndGrow[s, "spoolingQueueFull"L, z];
      systemError => String.AppendStringAndGrow[s, "systemError"L, z];
      tooManyClients => String.AppendStringAndGrow[s, "tooManyClients"L, z];
      undefinedError => String.AppendStringAndGrow[s, "undefinedError"L, z];
      transferError => String.AppendStringAndGrow[s, "transferError"L, z];
      connectionError => String.AppendStringAndGrow[s, "connectionError"L, z];
      courier => AppendCourierError[s, courier, z];
      ENDCASE;
    String.AppendCharAndGrow[s, '], z];
    END;

  AppendCourierError: PROCEDURE [s: LONG POINTER TO LONG STRING, e: Courier.ErrorCode, z: UNCOUNTED
ZONE] =
    BEGIN
    eString: STRING = SELECT e FROM
      transmissionMediumHardwareProblem => "transmissionMediumHardwareProblem"L,
      transmissionMediumUnavailable => "transmissionMediumUnavailable"L,
      transmissionMediumNotReady => "transmissionMediumNotReady"L,
      noAnswerOrBusy => "noAnswerOrBusy"L,
      noRouteToSystemElement => "noRouteToSystemElement"L,
      transportTimeout => "transportTimeout"L,
      remoteSystemElementNotResponding => "remoteSystemElementNotResponding"L,
      noCourierAtRemoteSite => "noCourierAtRemoteSite"L,
      tooManyConnections => "tooManyConnections"L,
      invalidMessage => "invalidMessage"L,
      noSuchProcedureNumber => "noSuchProcedureNumber"L,
      returnTimedOut => "returnTimedOut"L,
      callerAborted => "callerAborted"L,
      unknownErrorInRemoteProcedure => "unknownErrorInRemoteProcedure"L,
      streamNotYours => "streamNotYours"L,
      truncatedTransfer => "truncatedTransfer"L,
      parameterInconsistency => "parameterInconsistency"L,
      invalidArguments => "invalidArguments"L,
      noSuchProgramNumber => "noSuchProgramNumber"L,
      protocolMismatch => "protocolMismatch"L,
      duplicateProgramExport => "duplicateProgramExport"L,
      noSuchProgramExport => "noSuchProgramExport"L,
      invalidHandle => "invalidHandle"L,
      noError => "noError"L,
      ENDCASE => "?"L;
    String.AppendStringAndGrow[s, "Courier.Error["L, z];
    String.AppendStringAndGrow[s, eString, z];
    String.AppendCharAndGrow[s, '], z];
    END;

  GetStatus: PROCEDURE [host: LONG STRING]
    RETURNS [available: BOOLEAN ← TRUE, msg: LONG STRING] =
    BEGIN
    hostAddr: NSPrint.SystemElement = GetAddress[host];
```

```
    status: NSPrint.PrinterStatus;
    z: UNCOUNTED ZONE = Heap.systemZone;
    msg ← String.CopyToNewString[host, z, 100];
    String.AppendChar[msg, ':];
    status ← NSPrint.GetPrinterStatus[hostAddr !
      NSPrint.Error => WITH why SELECT FROM
        busy, tooManyClients => {String.AppendString[msg, " busy\n"L]; GOTO available};
        ENDCASE => {AppendNSPrintError[@msg, why, z]; GOTO notAvailable}];
    FOR i: CARDINAL IN [0..LENGTH[status]) DO
      WITH s: status[i] SELECT FROM
        spooler => {
          String.AppendStringAndGrow[@msg, " Spooler "L, z];
          SELECT s.spooler FROM
            available => String.AppendStringAndGrow[@msg, "available;"L, z];
            busy => String.AppendStringAndGrow[@msg, "busy;"L, z];
            disabled => String.AppendStringAndGrow[@msg, "disabled;"L, z];
            full => String.AppendStringAndGrow[@msg, "full;"L, z];
            ENDCASE};
        formatter => {
          String.AppendStringAndGrow[@msg, " Formatter "L, z];
          SELECT s.formatter FROM
            available => String.AppendStringAndGrow[@msg, "available;"L, z];
            busy => String.AppendStringAndGrow[@msg, "busy;"L, z];
            disabled => String.AppendStringAndGrow[@msg, "disabled;"L, z];
            ENDCASE};
        printer => {
          String.AppendStringAndGrow[@msg, " Printer "L, z];
          SELECT s.printer FROM
            available => String.AppendStringAndGrow[@msg, "available;"L, z];
            busy => String.AppendStringAndGrow[@msg, "busy;"L, z];
            disabled => String.AppendStringAndGrow[@msg, "disabled;"L, z];
            needsAttention => String.AppendStringAndGrow[@msg, "needs attention;"L, z];
            needsKeyOperator => String.AppendStringAndGrow[@msg, "needs key operator;"L, z];
            ENDCASE};
      ENDCASE;
      ENDLOOP;
    NSPrint.FreePrinterStatus[@status];
    String.AppendCharAndGrow[@msg, '\n, z];
    EXITS
      notAvailable => available ← FALSE;
      available => NULL;
    END;

  GetAddress: PROCEDURE [host: LONG STRING]
    RETURNS [addr: NSPrint.SystemElement] =
    BEGIN
    addr ← AddressTranslation.StringToNetworkAddress[host!
      AddressTranslation.Error => {
          msg : STRING = [100];
          appendProc: PROC[s: LONG STRING, clientData: LONG POINTER ← NIL] ={
             String.AppendString[msg,s ! String.StringBoundsFault => RESUME[NIL]];};
             AddressTranslation.PrintError[error: errorRecord, proc: appendProc];
     << Error => {
        msg: STRING = [100];
        String.AppendString[msg, host !
          String.StringBoundsFault => RESUME[NIL]];
        String.AppendString[msg, " - name not found"L !
          String.StringBoundsFault => RESUME[NIL]];}>>
        ERROR Trouble[other, msg]}].addr;
    END;

  END.

11-Apr-85 19:20:43 - Poskanzer.SV - Added the fix for the "drop the interpress master on the floor is
the printer's load is too high" bug.
```

```
-- PrintControl.mesa - edited by:
--  Loretta              3-Sep-81 14:39:53
--  Rhonda              11-Sep-81 15:27:30
--  Johnsson             6-Sep-83 15:12:31
--  Daniels             24-Jan-84 18:44:10
--  Gainey               6-Jul-84 17:18:52
--  Hamilton.ES         25-Nov-84 21:16:05
--  Poskanzer           12-Apr-85  9:38:48

-- Copyright (C) 1981, 1983, 1984  by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii USING [CR, NUL],
  Environment USING [Block, bytesPerPage, charsPerWord],
  Exec USING [
    AddCommand, CheckForAbort, ExecProc, FreeTokenString, GetToken, Handle, OutputProc, PutChar],
  FileName USING [AllocVFN, Error, FreeVFN, VirtualFilename],
  FileTransfer,
  Format USING [LongDecimal, StringProc],
  Heap USING [Create, Delete, systemZone],
  Inline USING [LongMult],
  Interpress82Maker USING [CreatePresser, CreatePrinter],
  Interpress82MakerExtra USING [SetCharacterCode],
  MFile USING [
    Acquire, AcquireTemp, Error, GetCreateDate, Handle, Release, SetAccess],
  MStream USING [Copy, Create, Error, GetLength, ReadWrite, SetLength, WriteOnly],
  OldPressMaker USING [CreatePresser, CreatePrinter],
  Press USING [Encoding, FontSlope, FontWeight, Handle, Mica, micasPerInch],
  PressPrint USING [Handle],
  PressStream USING [
    Create, GetPageNumber, Margins, Parameters, ParametersHandle, SetPageNumber,
    SetParameters],
  PressStreamExtra USING [Parameters, ParametersHandle, SetExtraParameters],
  PressUtilities USING [NoFontsDotWidths],
  PrintOpsExtra USING [SetupExtraHardCopyOptions],
  Process USING [Pause, SecondsToTicks],
  Runtime USING [IsBound],
  Selection USING [Convert, Source],
  Stream USING [
    CompletionCode, Delete, GetBlock, GetPosition, Handle, PutBlock,
    PutChar, SetPosition],
  String USING [
    AppendChar, AppendDecimal, AppendString, CopyToNewString, Empty,
    EqualString, EquivalentStrings],
  Time USING [Append, AppendCurrent, Current, Packed, Unpack];

PrintControl: MONITOR
  IMPORTS
    Exec, FileName, FileTransfer, Format, Heap, Inline, Interpress82Maker,
    Interpress82MakerExtra, MFile, MStream, OldPressMaker, PressUtilities,
    PressStream, PressStreamExtra, PrintOpsExtra, Process, Runtime, Selection,
    Stream, String, Time =
  BEGIN

  Mica: TYPE = Press.Mica;

  z: UNCOUNTED ZONE;

  execH: Exec.Handle ← NIL;

  PutString: Format.StringProc ← NIL;

  printerName: LONG STRING ← NIL;

  myPress: Press.Handle ← NIL;
  myPrint: PressPrint.Handle ← NIL;

  NUL: CHARACTER = Ascii.NUL;
  Inch: Mica = Press.micasPerInch;
  charsPerWord: CARDINAL = Environment.charsPerWord;
  bytesPerPage: CARDINAL = Environment.bytesPerPage;
  pagesPerChunk: CARDINAL = 200; -- number of pages to trigger printing

  debugging: BOOLEAN ← FALSE;

  pressStream: Stream.Handle ← NIL;
```

```
    pressFileName: LONG STRING ← NIL;

    transmitting: BOOLEAN;
    copies: CARDINAL;
    sides: CARDINAL;

    bufferStream: Stream.Handle;
    outputStream: Stream.Handle;
    tempStream: Stream.Handle ← NIL;
    fh: MFile.Handle;

    normalFont: CARDINAL = 0;

    Abort: SIGNAL = CODE;
    NoteError: SIGNAL [message: STRING] = CODE;

    fontChanged, fontSpecified: BOOLEAN;

    Defaults: TYPE = RECORD [font: LONG STRING, columns: CARDINAL];

    lDefault: Defaults ← [NIL, 2];
    pDefault: Defaults ← [NIL, 1];

    bufferFile, outputFile: LONG STRING ← NIL;
    haveStatus: BOOLEAN;

    cParameters, dParameters: PressStream.Parameters;
    cEParameters, dEParameters: PressStreamExtra.Parameters;
→   userName, font: LONG STRING;                    phoneNumber: LONG STRING ← NIL;

    fontWeight: Press.FontWeight ← medium;
    fontSlope: Press.FontSlope ← regular;

    busy: BOOLEAN ← FALSE;

    PutDecimal: PROCEDURE [d: CARDINAL] = {
      s: STRING = [10];
      String.AppendDecimal[s, d];
      PutString[s]};
    PutLine: PROCEDURE [s: LONG STRING] = {PutString[s]; Exec.PutChar[execH, Ascii.CR]};
    PutChar: PROCEDURE [c: CHARACTER] = {Exec.PutChar[execH, c]};
    PutCR: PROCEDURE = {Exec.PutChar[execH, Ascii.CR]};

    SetDebugging: PROCEDURE [d: BOOLEAN] =
      BEGIN
      debugging ← d;
      PutString["Debugging "L];
      PutLine[IF debugging THEN "on"L ELSE "off"L];
      RETURN
      END;

    SetDefaultFont: PROCEDURE [f: LONG STRING, p: PressStream.ParametersHandle] =
      BEGIN
      IF font # NIL AND String.EquivalentStrings[f, font] THEN RETURN;
      font ← String.CopyToNewString[f, z];
      fontChanged ← TRUE;
      RETURN
      END;

    SetHost: PROCEDURE [h: LONG STRING, p: PressStream.ParametersHandle] =
      BEGIN
      outputFile ← NIL;
      transmitting ← TRUE;
      IF printerName # NIL AND ~String.EquivalentStrings[h, printerName] THEN
        FinishFile[];
      printerName ← String.CopyToNewString[h, z];
      haveStatus ← FALSE;
      RETURN
      END;                                  SetRemote:

→   SetOutputFile: PROCEDURE [f: LONG STRING, p: PressStream.ParametersHandle] =
      BEGIN OPEN String;
      FinishFile[];
      printerName ← NIL;
      FOR i: CARDINAL IN [0..f.length) DO
```

```
      IF f[i] = '. THEN {IF i = f.length - 1 THEN AppendString[f, "press"L]; EXIT}
      REPEAT FINISHED => {
        AppendChar[f, '.];
        IF myPress.encoding = Interpress82 THEN AppendString[f, "inter"L];
        AppendString[f, "press"L]};
      ENDLOOP;
    outputFile ← String.CopyToNewString[f, z];
    transmitting ← FALSE;
    PutString["Output to "L];
    PutLine[f];
    RETURN
    END;

  SetLandscape: PROCEDURE [c: CARDINAL, p: PressStream.ParametersHandle,
    pE: PressStreamExtra.ParametersHandle] =
    BEGIN
    p.mode ← landscape;
    IF ~fontSpecified THEN SetDefaultFont[lDefault.font, p];
    p.columns ← c;
    p.margins ← pE.lMargins;
    RETURN
    END;

  SetPortrait: PROCEDURE [c: CARDINAL, p: PressStream.ParametersHandle,
    pE: PressStreamExtra.ParametersHandle] =
    BEGIN
    p.mode ← portrait;
    IF ~fontSpecified THEN SetDefaultFont[pDefault.font, p];
    p.columns ← c;
    p.margins ← pE.pMargins;
    END;

  SetTabWidth: PROCEDURE [c: CARDINAL, p: PressStream.ParametersHandle] =
    BEGIN p.tab ← c; END;

  SetCopies: PROCEDURE [c: CARDINAL, p: PressStream.ParametersHandle] =
    BEGIN copies ← c; END;

  SetSides: PROCEDURE [s: CARDINAL, p: PressStream.ParametersHandle] =
    BEGIN sides ← s; END;

  SetCharacterCode: PROCEDURE [
    n: CARDINAL, pE: PressStreamExtra.ParametersHandle] =
    BEGIN
    pE.xeroxCharacterCode ← n;
    IF Runtime.IsBound[LOOPHOLE[Interpress82MakerExtra.SetCharacterCode]] THEN
      Interpress82MakerExtra.SetCharacterCode[myPress, n];
    -- Maybe there should be a SetCharacterCode for OldPrint2 too?
    END;

  SetMargin: PROCEDURE [b: Buffer, p: PressStream.ParametersHandle] =
    BEGIN
    d: CHARACTER = GetMarginDirection[b];
    micas: CARDINAL = GetNumber[b, 1905];
    SELECT d FROM
      't => p.margins[top] ← micas;
      'b => p.margins[bottom] ← micas;
      'l => p.margins[left] ← micas;
      'r => p.margins[right] ← micas;
      ENDCASE => p.margins ← ALL[micas];
    END;

  AbortFile: PROCEDURE [s: STRING] =
    BEGIN
    PutLine[s];
    pressStream.Delete[];
    pressStream ← NIL;
    END;

  CheckStatus: PROCEDURE =
    BEGIN
    avail: BOOLEAN;
    msg: LONG STRING;
    IF haveStatus THEN RETURN;
    IF printerName = NIL OR printerName.length = 0 THEN {
      PutLine["No printer host specified"L]; ERROR Abort};
```

```
    [avail, msg] ← myPrint.GetStatus[printerName ! myPrint.Trouble => {
      IF ~String.Empty[message] THEN PutString[message];
      ERROR Abort}];
    IF ~String.Empty[msg] THEN PutString[msg];
    Heap.systemZone.FREE[@msg];
    haveStatus ← TRUE;
    IF ~avail THEN ERROR Abort;
    END;

    IsInterpressMaster: PROCEDURE [blockPointer :
     LONG POINTER TO PACKED ARRAY[0..0) OF CHARACTER]    RETURNS [BOOLEAN] =
    BEGIN
    headerString: STRING = "Interpress/Xerox"L;
    FOR i: CARDINAL IN [0..headerString.length) DO
      IF headerString[i] # blockPointer↑[i] THEN RETURN[FALSE]; ENDLOOP;
    RETURN[TRUE]
    END;

    PutPressType: PROCEDURE [type: Press.Encoding] =
    BEGIN
    PutString[IF type = Interpress82 THEN "Interpress"L ELSE "Press"L];
    END;

  PutAlready: PROCEDURE [type: Press.Encoding] =
    BEGIN
    PutString[" already in "L]; PutPressType[type]; PutString[" format... "L];
    END;

  SendPressFile: PROCEDURE [
    fileName: LONG STRING, vfn: FileName.VirtualFilename]
    RETURNS [isPressFile: BOOLEAN ← FALSE] =
    BEGIN
    IF String.Empty[vfn.host] THEN {
      fh: MFile.Handle = MFile.Acquire[fileName, anchor, []];
      isPressFile ← myPrint.IsPressFile[fh];
      IF isPressFile THEN {
        PutString[fileName];
        PutAlready[myPress.encoding];
        IF transmitting THEN {
          s: Stream.Handle;
          MFile.SetAccess[fh, readOnly];
          s ← MStream.Create[fh, []];
          SendStream[s, MStream.GetLength[s], fileName, MFile.GetCreateDate[fh] !
            UNWIND => Stream.Delete[s]];
          Stream.Delete[s]}
        ELSE PutString["skipped"L];
        RETURN}
      ELSE MFile.Release[fh]};
    END;

  PressThisFile: PROCEDURE [fileName: LONG STRING] =
    BEGIN
    conn: FileTransfer.Connection ← NIL;
    SetCredentials: PROCEDURE [name, password: LONG STRING] = {
      FileTransfer.SetPrimaryCredentials[conn, name, password]};
    isPressFile: BOOLEAN ← FALSE;
    vfn: FileName.VirtualFilename ← NIL;
    IF transmitting THEN CheckStatus[];
    IF String.EqualString[fileName, "$$$"L] THEN {
      PressCurrentSelection[]; RETURN};
    conn ← FileTransfer.Create[];
    FileTransfer.SetProcs[conn, NIL, Messages];
      BEGIN ENABLE {
        UNWIND => {FileName.FreeVFN[vfn]; FileTransfer.Destroy[conn]};
        FileName.Error => {
          PutString[fileName]; PutLine[" is an invalid file name."L];
          ERROR Abort};
        MFile.Error, FileTransfer.Error => {
          PutString["Can't read "L]; PutLine[fileName]; CONTINUE}};
      vfn ← FileName.AllocVFN[fileName];
      IF ~SendPressFile[fileName, vfn] THEN
        BEGIN
        IF fontChanged THEN InstallFont[font];
          BEGIN ENABLE FileTransfer.Error => SELECT code FROM
            retry => RETRY;
            skipOperation => CONTINUE;
```

```
            notFound => REJECT; -- caught above
                    ENDCASE;
          PrintStreams[
            FileTransfer.ReadStream[conn, vfn], String.Empty[vfn.host]];
          END;
        END;
      END;
    FileName.FreeVFN[vfn];
    FileTransfer.Destroy[conn];
    END;

  PrintStreams: PROCEDURE [stream: Stream.Handle, localFile: BOOLEAN] =
    BEGIN
    header: STRING = [150];
    pages: CARDINAL;
    fileInfo: FileTransfer.FileInfo;
    aborted: BOOLEAN ← FALSE;
    interpressMaster : BOOLEAN;
    tempBufferBytes: CARDINAL = 512;
    tempBuffer: PACKED ARRAY[0..tempBufferBytes] OF CHARACTER;
    tempPointer : LONG POINTER TO PACKED ARRAY[0..0) OF CHARACTER ← LOOPHOLE[LONG[@tempBuffer]];
    tempBlock: Environment.Block ← [LOOPHOLE[LONG[@tempBuffer]],0,];
    why: Stream.CompletionCode;
    tempBlock.stopIndexPlusOne ← tempBufferBytes;

    UNTIL stream = NIL DO
       BEGIN  -- check for non-text files
       fileInfo ← FileTransfer.GetStreamInfo[stream];
       IF fileInfo.type = directory THEN {PutChar['<];
          PutString[fileInfo.directory]; PutChar['>];
          PutString[fileInfo.body];
          NoteError[" directories may not be printed; file skipped"L];
          GOTO skip};
       [why: why, bytesTransferred: tempBlock.stopIndexPlusOne] ←
                stream.GetBlock[tempBlock];
       interpressMaster ← IsInterpressMaster[tempPointer];
       IF (interpressMaster AND (myPress.encoding = Interpress82)) THEN
         {
          FinishFile[];
          IF fileInfo.directory # NIL THEN {
             PutChar['<]; PutString[fileInfo.directory]; PutChar['>]};
          PutString[fileInfo.body];
          PutString[" Fetching "L];
          IF fileInfo.host # NIL THEN {
             fh ← MFile.AcquireTemp[type: text, initialLength: fileInfo.size !
                MFile.Error => {PutString[" no room on volume"L]; GOTO skip}];
             PutString[" ... "L];
             outputStream ← tempStream ← MStream.Create[file: fh, release: []];
             outputStream.PutBlock[tempBlock,why = endOfStream];
             IF why # endOfStream THEN
                [] ← MStream.Copy[from: stream, to: outputStream, bytes:
fileInfo.size];
             outputStream.SetPosition[0]}
          ELSE { outputStream ← stream; PutString[" ... "L]};
          Format.LongDecimal[PutString, fileInfo.size];
          PutString[" bytes"L];
          PutCR[];

          SendStream[outputStream, fileInfo.size, fileInfo.body, fileInfo.create ! Abort => CONTINUE];
          IF outputFile = NIL THEN {
             Stream.Delete[outputStream]; outputStream ← NIL};
            IF bufferStream # NIL THEN Stream.SetPosition[bufferStream, 0];
         }
       ELSE
          {
          IF pressStream # NIL AND outputFile = NIL AND
          (outputStream.GetPosition[]/bytesPerPage > pagesPerChunk) THEN
               FinishFile[];
           PutPressType[myPress.encoding]; PutString["ing "L];
           IF fileInfo.directory # NIL THEN {
               PutChar['<]; PutString[fileInfo.directory]; PutChar['>]};
           PutString[fileInfo.body];
           IF fileInfo.type # text AND ~debugging THEN {
             NoteError["only text files may be printed; file skipped"L];
             GOTO skip};
           ShowParameters[];
```

```
        header.length ← 0;
        String.AppendString[header, fileInfo.body];
        String.AppendString[header, "        "L];
        Time.Append[header, Time.Unpack[fileInfo.create], TRUE];
        cParameters.headerString ← header;

        IF pressStream = NIL THEN StartFile[fileInfo.body]
        ELSE pressStream.PutChar[13C]; -- start on new sheet
        PressStream.SetParameters[pressStream, @cParameters];
        PressStreamExtra.SetExtraParameters[pressStream, @cEParameters];
        PressStream.SetPageNumber[pressStream, 1];
        PutString["... "L];
        pressStream.PutBlock[tempBlock, why = endOfStream];
        IF why # endOfStream THEN
           CopyStream[from: stream, to: pressStream];
        pages ← PressStream.GetPageNumber[pressStream];
        IF sides = 2 AND (pages MOD 2) = 1 THEN myPress.NewPlate[myPress];
        cParameters.headerString ← NIL;
        PutDecimal[pages];
        PutString[" page"L];
        IF pages # 1 THEN PutChar['s];
      };
      PutCR[];
      EXITS skip => PutCR[];
      END; -- check for non text files
      IF tempStream # NIL THEN { tempStream ← NIL};
      IF aborted THEN { IF stream # NIL THEN stream.Delete[];
          EXIT};
      IF localFile THEN {stream.Delete[]; EXIT}
      ELSE stream ← FileTransfer.ReadNextStream[stream];
    ENDLOOP;
    RETURN
    END;

  CopyStream: PROCEDURE [from, to: Stream.Handle] =
    BEGIN
    bufferBytes: CARDINAL = 512;
    buffer: PACKED ARRAY [0..bufferBytes] OF CHARACTER;
    block: Environment.Block ← [LOOPHOLE[LONG[@buffer]], 0, ];
    why: Stream.CompletionCode;
    DO
      block.stopIndexPlusOne ← bufferBytes;
      [why: why, bytesTransferred: block.stopIndexPlusOne] ← from.GetBlock[block];
      to.PutBlock[block, why = endOfStream];
      IF why = endOfStream THEN EXIT;
      IF Exec.CheckForAbort[execH] THEN ERROR Abort;
      ENDLOOP;
    END;

  Messages: FileTransfer.MessageProc =
    BEGIN
    IF level <= warning AND ~debugging THEN RETURN;
    IF s1 # NIL THEN PutString[s1];
    IF s2 # NIL THEN PutString[s2];
    IF s3 # NIL THEN PutString[s3];
    IF s4 # NIL THEN PutString[s4];
    END;

  StartFile: PROCEDURE [name: LONG STRING] =
    BEGIN
    ENABLE MStream.Error => {
      PutString[" Can't write on "L];
      PutLine[IF outputFile = NIL THEN bufferFile ELSE outputFile];
      ERROR Abort};
    IF outputFile = NIL THEN {
      IF bufferStream = NIL THEN
        bufferStream ← MStream.ReadWrite[bufferFile, [], binary];
      outputStream ← bufferStream}
    ELSE outputStream ←
      IF transmitting THEN MStream.ReadWrite[outputFile, [], binary]
      ELSE MStream.WriteOnly[outputFile, [], binary];
    pressStream ← PressStream.Create[myPress, outputStream, name];
    pressFileName ← String.CopyToNewString[name, z];
    END;

  FinishFile: PROCEDURE =
```

```
    BEGIN
    index: LONG CARDINAL;
    IF pressStream = NIL THEN RETURN;
    pressStream.Delete[]; pressStream ← NIL;
    index ← Stream.GetPosition[outputStream];
    MStream.SetLength[outputStream, index];
    IF transmitting THEN {
      Stream.SetPosition[outputStream, 0];
      SendStream[outputStream, index, pressFileName, Time.Current[]]};
    IF outputFile # NIL THEN {
      Stream.Delete[outputStream]; outputStream ← NIL};
    IF bufferStream # NIL THEN Stream.SetPosition[bufferStream, 0];
    END;

  SendStream: PROCEDURE [s: Stream.Handle, bytes: LONG CARDINAL,
    fileName: LONG STRING, createTime: Time.Packed] =
    BEGIN
    aborted: BOOLEAN ← FALSE;
    IF Exec.CheckForAbort[execH] THEN SIGNAL Abort;
    PutString["sending to "L];
    PutString[printerName];
    PutString["... "L];
    myPrint.SendPressStream[s, bytes, printerName, copies,
      sides, fileName, userName, createTime !
        myPrint.Trouble => SELECT code FROM
          busy => {
            Wait[IF ~String.Empty[message] THEN message ELSE "busy"L, 8]; RESUME};
          timeout => {Wait["not responding"L, 4]; RESUME};
          badFile => {
            IF message # NIL THEN PutString[message]
            ELSE PutString["bad file format"L];
            PutString["..."L];
            aborted ← TRUE;
            CONTINUE};
          other => {
            IF ~String.Empty[message] THEN {
              PutString[message]; PutString["..."L]};
            aborted ← TRUE;
            CONTINUE};
          ENDCASE];
    PutLine[IF aborted THEN " Aborted"L ELSE " Done"L];
    END;

  PressCurrentSelection: PROC =
    BEGIN
    source: Selection.Source;
    IF pressStream # NIL AND outputFile = NIL AND (Stream.GetPosition[
      outputStream] / bytesPerPage) > pagesPerChunk THEN FinishFile[];
    IF (source ← Selection.Convert[source]) = NIL THEN {
      PutString["No valid selection"L]; RETURN};
    PutPressType[myPress.encoding]; PutString["ing current selection"L];
    ShowParameters[];
    PutString["... "L];
      BEGIN
      pages: CARDINAL;
      header: STRING = [24];
      Time.AppendCurrent[header];
      IF fontChanged THEN InstallFont[font];
      IF pressStream = NIL THEN StartFile["CurrentSelection"L];
      cParameters.headerString ← header;
      PressStream.SetParameters[pressStream, @cParameters];
      PressStreamExtra.SetExtraParameters[pressStream, @cEParameters];
      PutDecimal[pages←PrintSelection[source]];
      IF sides = 2 AND (pages MOD 2) = 1 THEN myPress.NewPlate[myPress];
      cParameters.headerString ← NIL;
      PutString[" page"L];
      IF pages # 1 THEN PutChar['s];
      END;
    PutCR[];
    RETURN
    END;

  PrintSelection: PROCEDURE [source: Selection.Source]
    RETURNS [lastPage: CARDINAL] =
    BEGIN
    s: STRING ← [100];
```

```
    DO ENABLE UNWIND => source.destroy[source];
      IF Exec.CheckForAbort[execH] THEN ERROR Abort;
      s.length ← 0;
      source.proc[source.data, s];
      IF s.length = 0 THEN {source.destroy[source]; EXIT};
      pressStream.PutBlock[[LOOPHOLE[LONG[@s.text]], 0, s.length]];
      ENDLOOP;
    lastPage ← PressStream.GetPageNumber[pressStream];
    RETURN
    END;

  CheckForExtension: PROCEDURE [name, ext: LONG STRING] =
    BEGIN
    FOR i: CARDINAL DECREASING IN [0..name.length) DO
      IF name[i] = '. THEN RETURN;
      ENDLOOP;
    String.AppendString[name, ext];
    END;

  PointsToMicas: PROCEDURE [p: CARDINAL] RETURNS [Mica] = INLINE {
    RETURN[(Inline.LongMult[p, 635]+9)/18]};

  InstallFont: PROCEDURE [f: LONG STRING, mail: BOOLEAN ← FALSE] =
    BEGIN
    name: STRING ← [40];
    b: BufferItem ← [0, f];
    c: CHARACTER;
    points: CARDINAL;
    FinishFile[];
    DO
      c ← GetChar[@b];
      IF c = NUL THEN EXIT;
      IF c IN ['0..'9] THEN BEGIN Backup[@b]; EXIT END;
      String.AppendChar[name, c];
      ENDLOOP;
    points ← GetNumber[@b, IF cParameters.mode = landscape THEN 6 ELSE 8];
    fontWeight ← medium;
    fontSlope ← regular;
    UNTIL (c ← GetChar[@b]) = NUL DO
      IF c = 'b THEN fontWeight ← bold
      ELSE IF c = 'i THEN fontSlope ← italic;
      ENDLOOP;
    myPress.ClearAliases[myPress];
    myPress.DefineAlias[
      myPress, normalFont, name, PointsToMicas[points], fontWeight, fontSlope];
    fontChanged ← FALSE;
    RETURN
    END;

  Wait: PROCEDURE [why: LONG STRING, howlong: CARDINAL] =
    BEGIN
    start: Time.Packed;
    PutString["\nServer "L];
    PutString[why];
    PutString["... will retry"L];
    start ← Time.Current[];
    UNTIL Time.Current[] - start > howlong DO
      IF Exec.CheckForAbort[execH] THEN SIGNAL Abort;
      Process.Pause[Process.SecondsToTicks[1]];
      ENDLOOP;
    PutString["... "L];
    END;

  ShowParameters: PROCEDURE =
    BEGIN
    PutChar['/];
    PutChar[IF cParameters.mode = landscape THEN 'l ELSE 'p];
    PutDecimal[cParameters.columns];
    IF ~cParameters.headers THEN PutString["~a"L];
    IF ~cParameters.trailers THEN PutString["~z"L];
    IF sides # 0 THEN {PutChar['s]; PutDecimal[sides]};
    IF cEParameters.mapArrows THEN PutString["m"L];
    IF ~cEParameters.indentContinuations THEN PutString["~i"L];
    IF cEParameters.xeroxCharacterCode # 2 THEN {
      PutChar['x];
      PutDecimal[cEParameters.xeroxCharacterCode]};
```

```
    END;

  InitGlobalParameters: PROCEDURE =
    BEGIN
    spruceName, nsName: LONG STRING ← NIL;
    [userName: userName, sprucePrinter: spruceName,
      interpressPrinter: nsName, pFont: pDefault.font, lFont: lDefault.font,
      parameters: dParameters, extraParameters: dEParameters] ←
      PrintOpsExtra.SetupExtraHardCopyOptions[z];
    IF myPress.encoding = Interpress82 THEN {
      printerName ← nsName; z.FREE[@spruceName]}
    ELSE {printerName ← spruceName; z.FREE[@nsName]};
    font ← IF dParameters.mode = landscape THEN lDefault.font ELSE pDefault.font;
    fontSpecified ← FALSE;
    fontChanged ← TRUE;
    copies ← 1;                                    phoneNumber ← NIL;
    sides ← 0;
    bufferFile ← String.CopyToNewString["Print.scratch$"L, z];
    outputFile ← NIL;
    END;

  SetGlobalParameters: PROCEDURE [b: Buffer] =
    BEGIN
    d: PressStream.ParametersHandle = @dParameters;
    dE: PressStreamExtra.ParametersHandle = @dEParameters;
    sense: BOOLEAN ← TRUE;
    sc: CHARACTER;
    UNTIL (sc ← GetChar[b]) = NUL DO
      SELECT sc FROM
        'a => d.headers ← sense;
        'b => SetMargin[b, d];
        'c => SetCopies[GetNumber[b, 1], d];
        'd => SetDebugging[~debugging];
        'i => dE.indentContinuations ← sense;
        'l => SetLandscape[GetNumber[b, 2], d, dE];
        'm => dE.mapArrows ← sense;
        'p => SetPortrait[GetNumber[b, 1], d, dE];
        's => SetSides[GetNumber[b, 0], d];
        't => SetTabWidth[GetNumber[b, 8], d];
        'x => SetCharacterCode[GetNumber[b, 2], dE];
        'z => d.trailers ← sense;
        '-, '~ => {sense ← FALSE; LOOP};
        ENDCASE;
      sense ← TRUE;
      ENDLOOP;
    InitCurrentParameters[]
    END;

  InitCurrentParameters: PROCEDURE =
    BEGIN
    cParameters ← dParameters;
    cEParameters ← dEParameters;
    END;

  BufferItem: TYPE = RECORD [p: CARDINAL, s: LONG STRING];

  Buffer: TYPE = POINTER TO BufferItem;

  GetChar: PROCEDURE [b: Buffer] RETURNS [c: CHARACTER] =
    BEGIN OPEN b;
    c ← Ascii.NUL;
    IF s # NIL AND p < s.length THEN
      BEGIN
      c ← s[p];
      p ← p + 1;
      IF c IN ['A..'Z] THEN c ← LOOPHOLE[LOOPHOLE[c, CARDINAL] + 40B];
      END;
    RETURN
    END;

  Backup: PROCEDURE [b: Buffer] = BEGIN IF b.p # 0 THEN b.p ← b.p - 1; END;

  GetNumber: PROCEDURE [b: Buffer, default: CARDINAL] RETURNS [v: CARDINAL] =
    BEGIN
    c: CHARACTER;
    usedefault: BOOLEAN ← TRUE;
```

```
    v ← 0;
    WHILE (c ← GetChar[b]) IN ['0..'9] DO
      usedefault ← FALSE; v ← v*10 + (c - 60C); ENDLOOP;
    IF c # NUL THEN Backup[b];
    IF usedefault THEN RETURN[default];
    END;

  GetMarginDirection: PROCEDURE [b: Buffer] RETURNS [d: CHARACTER] =
    BEGIN
    d ← GetChar[b];
    IF d # 't AND d # 'b AND d # 'l AND d # 'r THEN {
      IF d # NUL THEN Backup[b];
      d ← 'a};
    END;

  ProcessItem: PROCEDURE [arg, switches: LONG STRING] =
    BEGIN
    b: BufferItem ← [0, switches];
    c: PressStream.ParametersHandle = @cParameters;
    cE: PressStreamExtra.ParametersHandle = @cEParameters;
    sc: CHARACTER;
    sense: BOOLEAN ← TRUE;
    mail: BOOLEAN ← FALSE;
    IF Exec.CheckForAbort[execH] THEN SIGNAL Abort;
    IF arg = NIL THEN {SetGlobalParameters[@b]; RETURN};
    UNTIL arg.length = 0 OR (sc ← GetChar[@b]) = NUL DO
      SELECT sc FROM
        'a => c.headers ← sense;
        'b => SetMargin[@b, c];
        'c => {SetCopies[GetNumber[@b, 1], c]; SetCopies[copies, @dParameters]};
        'd => SetDebugging[~debugging];
        'f => {SetDefaultFont[arg, c]; fontSpecified ← TRUE; arg.length ← 0};
        'h => {SetHost[arg, c]; arg.length ← 0};
        'i => cE.indentContinuations ← sense;
        'l => SetLandscape[GetNumber[@b, 2], c, cE];
        'm => cE.mapArrows ← sense;
        'o => {SetOutputFile[arg, c]; arg.length ← 0};
        'p => SetPortrait[GetNumber[@b, 1], c, cE];
        's => SetSides[GetNumber[@b, 0], c];
        't => SetTabWidth[GetNumber[@b, 8], c];
        'x => SetCharacterCode[GetNumber[@b, 2], cE];
        'z => c.trailers ← sense;
        '-, '~ => {sense ← FALSE; LOOP};
        ENDCASE =>
          BEGIN
          PutString["Unknown switch = "L];
          PutChar[sc];
          PutCR[];
          END;
      sense ← TRUE;
      ENDLOOP;
    IF arg.length = 0 THEN SetGlobalParameters[@b]
    ELSE BEGIN PressThisFile[arg]; InitCurrentParameters[]; END;
    RETURN
    END;

  SetBusy: ENTRY PROCEDURE RETURNS [BOOLEAN] = {
    IF busy THEN RETURN[FALSE]
    ELSE RETURN[busy ← TRUE]};

  ClearBusy: ENTRY PROCEDURE = {busy ← FALSE};

  Command: Exec.ExecProc =
    BEGIN
    arg: LONG STRING ← NIL;
    switches: LONG STRING ← NIL;
    b: BufferItem ← [0, switches];
    Cleanup: PROCEDURE = {
      myPress.Delete[myPress]; myPress ← NIL;
      myPrint.Delete[myPrint]; myPrint ← NIL;
      IF bufferStream # NIL THEN Stream.Delete[bufferStream];
      IF outputStream # NIL AND outputStream # bufferStream THEN
        Stream.Delete[outputStream];
      outputStream ← bufferStream ← NIL;
      font ← bufferFile ← outputFile ← pressFileName ← NIL;
      execH ← NIL;
```

```
      PutString ← NIL;
      arg ← Exec.FreeTokenString[arg];
      switches ← Exec.FreeTokenString[switches];
      Heap.Delete[z]; z ← NIL;
      ClearBusy[]};

    IF ~SetBusy[] THEN {
      outcome ← abort; Exec.OutputProc[h]["Command already running"L]; RETURN};
    execH ← h;
    PutString ← Exec.OutputProc[h];
    haveStatus ← FALSE;
    transmitting ← TRUE;
    pressStream ← NIL;
    pressFileName ← NIL;
    bufferStream ← NIL;
    outputStream ← NIL;
    z ← Heap.Create[initial: 8];
    IF Runtime.IsBound[LOOPHOLE[Interpress82Maker.CreatePrinter]] THEN {
      myPress ← Interpress82Maker.CreatePresser[z];
      myPrint ← Interpress82Maker.CreatePrinter[]}
    ELSE {
      myPress ← OldPressMaker.CreatePresser[z];
      myPrint ← OldPressMaker.CreatePrinter[]};

    BEGIN ENABLE UNWIND => Cleanup[];
      InitGlobalParameters[];
      SetGlobalParameters[@b];
      InitCurrentParameters[];
      DO
        [arg, switches] ← Exec.GetToken[h];
        IF (arg = NIL OR arg.length = 0) AND switches = NIL THEN EXIT;
        ProcessItem[arg, switches !
          Abort => {outcome ← abort; EXIT};
          NoteError => {
            PutString[message]; outcome ← error; RESUME};
          PressUtilities.NoFontsDotWidths => {
            IF outcome = normal THEN {
              PutString["**No ""Fonts.widths"" file**"L]; outcome ← warning};
            RESUME}];
        arg ← Exec.FreeTokenString[arg];
        switches ← Exec.FreeTokenString[switches];
        ENDLOOP;
      IF outcome # abort THEN FinishFile[! Abort => {outcome ← abort; CONTINUE}];
      END;
    Cleanup[];
    RETURN
    END;

  HelpGeneral: PROC [h: Exec.Handle] = {
    Exec.OutputProc[h][
"Print to print the current selection rather than a file.  Local switches affect the printing of that
file only.  Global switches affect all subsequent input files.  The following switches are available:

    /a          Print headings on each page (default true; -a disables).
    /b<micas>   Set all borders.
    /bt<micas>  Set top border.
    /bb<micas>  Set bottom border.
    /bl<micas>  Set left border.
    /br<micas>  Set right border.
    /c<n>       Set number of copies to <n> (default 1).
    <font>/f    Change the font to <font> for the files that follow.
    <host>/h    Direct the output to <host> for the files that follow.
    /i          Specifies indenting of continued lines (default true;
                -i disables).
    /l<n>       Specifies landscape orientation.  <n> is the number of
                columns (default 2).
    /m          Map old arrows ← and ↑ to printable chars.
    <file>/o    Create a press file in <file> (extension defaults to
                .interpress) and disables transmission to the printer.
    /p<n>       Specifies portrait orientation.  <n> is the number of
                columns (default 1).
    /s<n>       Specifies number of sides.  <n> can be 0, 1, or 2, where
                0 means use the printer's default.
    /t<n>       Change the tab stops to every <n> spaces (default 8).
    /x<n>       Specifies Xerox Character Code <n> (default 2).
    /z          Print footings on each page (default true; -z disables)."L];};
```

```
  HelpOldPrint: Exec.ExecProc =
    BEGIN
    Exec.OutputProc[h][
"OldPrint converts text files to Press format files for printing and sends the result to a printer.
The command line format is:

    OldPrint <file>/<localSwitches> /globalSwitches <file>/<localSwitches> ...

Files can be on a remote fileserver.  The special filename $$$ instructs Old"L];
    HelpGeneral [h];
    END;  --HelpOldPrint

  HelpPrint: Exec.ExecProc =
    BEGIN
    Exec.OutputProc[h][
"Print converts text files to Interpress masters for printing and sends the result to a printer.  The
command line format is:

    Print <file>/<localSwitches> /globalSwitches <file>/<localSwitches> ...

Files can be on a remote fileserver.  The special filename $$$ instructs "L];
    HelpGeneral [h];
    END;  --HelpPrint

  Init: PROCEDURE = {
    IF Runtime.IsBound[LOOPHOLE[Interpress82Maker.CreatePrinter]] THEN
      Exec.AddCommand["Print.~"L, Command, HelpPrint]
    ELSE IF Runtime.IsBound[LOOPHOLE[OldPressMaker.CreatePrinter]] THEN
      Exec.AddCommand["OldPrint.~"L, Command, HelpOldPrint];
    };

  -- Main body

  Init[];

  END...

25-Nov-84 20:04:29 - Hamilton.ES - Add Help (Poskanzer.SV).
30-Jan-85 19:27:01 - Poskanzer.SV - Add /i, /x, and /m options.
10-Apr-85 18:50:22 - Poskanzer.SV - Add /← option.
```

```
-- Copyright (C) 1983  by Xerox Corporation. All rights reserved.
-- File: FileContainerShell.mesa - last edit:
-- Breisacher.ES        12-Oct-83  9:15:19
-- Holbrook.ES           5-Aug-83 10:13:17


DIRECTORY
  FileContainerSource USING [ColumnContents, Options],
  ContainerSource USING [Handle, ItemIndex],
  ContainerWindow USING [ColumnHeaders],
  MenuData USING [ArrayHandle],
  NSFile USING [Reference, Scope],
  StarWindowShell USING [Handle],
  Window USING [Handle];

FileContainerShell: DEFINITIONS = BEGIN

  Create: PROCEDURE [
    file: NSFile.Reference,
    columnHeaders: ContainerWindow.ColumnHeaders,
    columnContents: FileContainerSource.ColumnContents,
    regularMenuItems, topPusheeMenuItems: MenuData.ArrayHandle ← NIL,
    scope: NSFile.Scope ← [],
    position: ContainerSource.ItemIndex ← 0,
    options: FileContainerSource.Options ← []]
    RETURNS [
    shell: StarWindowShell.Handle];

  GetContainerWindow: PROCEDURE [shell: StarWindowShell.Handle]
    RETURNS [window: Window.Handle];

  GetContainerSource: PROCEDURE [shell: StarWindowShell.Handle]
    RETURNS [source: ContainerSource.Handle];

  END.
```

```
-- File: FileContainerShellImpl.mesa - last edit:
-- Holbrook            28-Oct-87 14:39:25
-- Breisacher          19-Aug-87  9:50:02
-- Holbrook.ES          9-Sep-86  9:18:13

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, MakeAtom],
  AtomicProfile USING [GetBOOLEAN],
  BWSZone USING [shortLifetime],
  Containee USING [Data, GetCachedName, GetCachedType, GetImplementation,
    ReturnTicket, Ticket],
  ContainerSource USING [ActOn, Handle, ItemIndex, nullItem],
  ContainerSourceExtra USING [Procedures],
  ContainerWindow USING [ColumnHeaders, DeleteAndShowNextPrevious, Destroy, Error, GetSource],
  ContainerWindowExtra3 USING [Access,  fullAccess, readOnlyAccess],
  ContainerWindowExtra4 USING [CreateXX],
  FileContainerShell USING [],
  FileContainerShellExtra,
  FileContainerShellExtra3,
  FileContainerSource USING [ColumnContents, Options],
  Heap USING [systemZone],
  FileContainerSourceExtra3 USING [CreateX],
  MenuData USING [ArrayHandle, CreateItem, CreateMenu, ItemHandle, MenuHandle,
    MenuProc],
  NSFile USING [localSystemElement, nullReference, Reference, Scope, Type],
  Selection USING [Convert, Free, Value],
  SpecialContainer USING [MakeItemVisible],
  SpecialContainer3 USING [AboutToDestroySource],
  StarWindowShell USING [Create, CreateBody, GetBody, GetZone, Handle,
     SetMiddlePusheeCommands, SetRegularCommands, SetPreferredDims, SetTopPusheeCommands, TransitionProc],
  Window USING [Dims, GetBox, Handle],
  XString USING [Character, FromSTRING, InvalidNumber, Overflow, ReaderBody, ReaderToNumber],
  UserTerminal;

FileContainerShellImpl: PROGRAM
  IMPORTS Atom, AtomicProfile, BWSZone, Containee, ContainerSource, ContainerWindow, ContainerWindowExtra4, FileContainerSourceExtra3,
  Heap, MenuData, NSFile, Selection, SpecialContainer, SpecialContainer3, StarWindowShell, UserTerminal, Window, XString
  EXPORTS FileContainerShell, FileContainerShellExtra, FileContainerShellExtra3 = BEGIN OPEN FileContainerShell;

  MenuItemSeq: TYPE = RECORD [SEQUENCE length: CARDINAL OF MenuData.ItemHandle];

  z: UNCOUNTED ZONE ← Heap.systemZone;
  true: BOOLEAN ← TRUE;
  false: BOOLEAN ← FALSE;

  containerDebugging: Atom.ATOM ← Atom.MakeAtom["ContainerDebugging"L];

  debugging: BOOLEAN ← AtomicProfile.GetBOOLEAN[containerDebugging];

  Create: PUBLIC PROCEDURE [
    file: NSFile.Reference,
    columnHeaders: ContainerWindow.ColumnHeaders,
    columnContents: FileContainerSource.ColumnContents,
    regularMenuItems, topPusheeMenuItems: MenuData.ArrayHandle ← NIL,
    scope: NSFile.Scope ← [],
    position: ContainerSource.ItemIndex ← 0,
    options: FileContainerSource.Options ← []]
    RETURNS [shell: StarWindowShell.Handle] = {
    shell ← RealCreate[file, columnHeaders, columnContents, regularMenuItems, topPusheeMenuItems, scope, position, IF options.readOnly
    THEN ContainerWindowExtra3.readOnlyAccess ELSE ContainerWindowExtra3.fullAccess]};

  CreateX: PUBLIC PROCEDURE [
    file: NSFile.Reference,
    columnHeaders: ContainerWindow.ColumnHeaders,
    columnContents: FileContainerSource.ColumnContents,
    regularMenuItems, topPusheeMenuItems: MenuData.ArrayHandle ← NIL,
    scope: NSFile.Scope ← [],
    position: ContainerSource.ItemIndex ← 0,
    access: ContainerWindowExtra3.Access ← ContainerWindowExtra3.fullAccess]
    RETURNS [shell: StarWindowShell.Handle] = {
    shell ← RealCreate[file, columnHeaders, columnContents, regularMenuItems, topPusheeMenuItems, scope, position, access, TRUE]};

  CreateX3, RealCreate: PUBLIC PROCEDURE [  -- 15030 CreateX2 allows no coversheet
    file: NSFile.Reference,
    columnHeaders: ContainerWindow.ColumnHeaders,
    columnContents: FileContainerSource.ColumnContents,
    regularMenuItems, topPusheeMenuItems: MenuData.ArrayHandle ← NIL,
    scope: NSFile.Scope ← [],
    position: ContainerSource.ItemIndex ← 0,
    access: ContainerWindowExtra3.Access ← ContainerWindowExtra3.fullAccess,
    considerShowingCoverSheet: BOOLEAN ← TRUE]
    RETURNS [shell: StarWindowShell.Handle] =
    BEGIN

    body: Window.Handle ← NIL;
    source: ContainerSource.Handle ← NIL;
    sourceX: ContainerSourceExtra.Procedures;
    cwRegularMenuItems, cwTopPusheeMenuItems: MenuData.ArrayHandle;
    mergedMenuItems: LONG POINTER TO MenuItemSeq ← NIL;
    menu: MenuData.MenuHandle;
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    data: Containee.Data ← [file];
    type: NSFile.Type;
    smallPicture: XString.Character;
```

```
    ContainerWidth: PROC [cols: ContainerWindow.ColumnHeaders]
      RETURNS [width: CARDINAL ← 0] = {
        -- add up column widths
        FOR i: CARDINAL IN [0..LENGTH[cols]) DO
          width ← width + cols[i].width;
        ENDLOOP;
        RETURN [width + 30]; -- a little extra
      };

      SetDebugCommands: PROC = {
        menuArray: ARRAY [0..2) OF MenuData.ItemHandle;
        discardString: XString.ReaderBody ←
          XString.FromSTRING ["Discard"L, TRUE];
        miString: XString.ReaderBody ←
          XString.FromSTRING ["MakeItemVisible"L, TRUE];
        menuArray[0] ← MenuData.CreateItem [
          zone: z, name: @discardString, proc: DiscardProc, itemData: body ];
        menuArray[1] ← MenuData.CreateItem [
          zone: z, name: @miString, proc: MakeVisibleProc, itemData: body ];
        menu ← MenuData.CreateMenu [zone: StarWindowShell.GetZone[shell],
          title: NIL,
          array: DESCRIPTOR[menuArray], copyItemsIntoMenusZone: TRUE];
        StarWindowShell.SetMiddlePusheeCommands [sws: shell, commands: menu];
        }; -- SetDebugCommands

      BEGIN
      width: CARDINAL;
      swsDims: Window.Dims;
      IF file = NSFile.nullReference THEN RETURN [ [NIL] ];
      [source, sourceX] ← FileContainerSourceExtra3.CreateX [
        file: file,
        columns: columnContents,
        scope: scope];

      IF source = NIL THEN RETURN [[NIL]];

      [name, ticket] ← Containee.GetCachedName [@data];
      type ← Containee.GetCachedType[@data];
      smallPicture ← Containee.GetImplementation[type].smallPictureProc[@data, type, normal];

      shell ← StarWindowShell.Create [
        name: @name,
        namePicture: smallPicture,
        sleeps: IsLocal [file],
        transitionProc: DestroyProc,
        considerShowingCoverSheet: considerShowingCoverSheet]; -- 15030 coversheet optional

      Containee.ReturnTicket [ticket];

      width ← ContainerWidth[columnHeaders];
      swsDims ← Window.GetBox[shell].dims;
      swsDims.w ← width;
      StarWindowShell.SetPreferredDims[shell, swsDims];

      body ← StarWindowShell.CreateBody [sws: shell, box: [[0,0],[width, 29999]]];
       -- 9676: open container only as wide as columns
      -- StarWindowShell.SetBodyWindowJustFits[shell, TRUE];

      [cwRegularMenuItems, cwTopPusheeMenuItems] ← ContainerWindowExtra4.CreateXX [
        window: body,
        source: source,
        sourceX: sourceX,
        columnHeaders: columnHeaders,
        firstItem: position,
        access: access];

      mergedMenuItems ← MergeMenuArrays [cwRegularMenuItems, regularMenuItems];
      IF mergedMenuItems # NIL THEN
        BEGIN
      menu ← MenuData.CreateMenu [
        zone: StarWindowShell.GetZone[shell],
        title: NIL,
        array: DESCRIPTOR[mergedMenuItems],
        copyItemsIntoMenusZone: TRUE ];
      StarWindowShell.SetRegularCommands [shell, menu];
      z.FREE[@mergedMenuItems];
        END;

      mergedMenuItems ← MergeMenuArrays [cwTopPusheeMenuItems, topPusheeMenuItems];
      menu ← MenuData.CreateMenu [
        zone: StarWindowShell.GetZone[shell],
        title: NIL,
        array: DESCRIPTOR[mergedMenuItems],
        copyItemsIntoMenusZone: FALSE ];
      StarWindowShell.SetTopPusheeCommands [shell, menu];
      z.FREE[@mergedMenuItems];
      IF debugging THEN SetDebugCommands[];

      -- EXITS ErrorExit => shell ← [NIL];
      END;
      RETURN [shell];
      END;

    DiscardProc: MenuData.MenuProc = {
      ContainerWindow.DeleteAndShowNextPrevious
        [window: LOOPHOLE[itemData], item: ContainerSource.nullItem,
```

```
      direction: next];
    };

  MakeVisibleProc: MenuData.MenuProc = {
    itemString: Selection.Value ← Selection.Convert[target: string,
      zone: BWSZone.shortLifetime];
    cw: Window.Handle ← StarWindowShell.GetBody[[window]];
    IF itemString.value # NIL THEN {
      item: CARDINAL ← CARDINAL[XString.ReaderToNumber[itemString.value
        ! XString.Overflow, XString.InvalidNumber =>
          GOTO GetOut]];
       SpecialContainer.MakeItemVisible [window: cw, item: item
         ! ContainerWindow.Error => { UserTerminal.BlinkDisplay[];
           CONTINUE}];
       Selection.Free[@itemString]};
    EXITS
      GetOut => RETURN;
    };

  GetContainerWindow: PUBLIC PROCEDURE [shell: StarWindowShell.Handle]
    RETURNS [window: Window.Handle] = BEGIN
    window ← StarWindowShell.GetBody[shell];
    [] ← ContainerWindow.GetSource[window];
       -- this will raise an error for us if this isn't a container window
    END;

  GetContainerSource: PUBLIC PROCEDURE [shell: StarWindowShell.Handle]
    RETURNS [source: ContainerSource.Handle] = BEGIN
    RETURN [ContainerWindow.GetSource [StarWindowShell.GetBody[shell]]];
    END;

  -- Private procs

  DestroyProc: StarWindowShell.TransitionProc =
  <<[sws: StarWindowShell.Handle, state: StarWindowShell.State]>>
    BEGIN
    IF state=dead THEN {
      cw: Window.Handle ← GetContainerWindow[sws];
      source: ContainerSource.Handle ← GetContainerSource[sws];
      << 13248 We need to make sure that no one can get a hold of a source via EnumerateSources while we're in the process of destroying
      the container window and source.  Race condition can occur: we can destroy the container window at the same time someone's trying
      to unbusy something; they try to update the container window but find nothing but garbage. >>

      -- destroy container window first so source is still available
      SpecialContainer3.AboutToDestroySource[source];
      ContainerWindow.Destroy[cw];
      ContainerSource.ActOn [source, destroy]; };
    RETURN;
    END;

  IsLocal: PROC [ref: NSFile.Reference] RETURNS [yes: BOOLEAN] =
    BEGIN
    RETURN [ref.service.systemElement = NSFile.localSystemElement];
    END;

  MergeMenuArrays: PROC [itemArray1, itemArray2: MenuData.ArrayHandle]
    RETURNS [mergedSeq: LONG POINTER TO MenuItemSeq] = BEGIN
    i: CARDINAL ← 0;
    IF itemArray1 = NIL AND itemArray2 = NIL THEN RETURN[NIL];
    mergedSeq ← z.NEW [MenuItemSeq[itemArray1.LENGTH + itemArray2.LENGTH]];
    FOR j: CARDINAL IN [0..itemArray1.LENGTH) DO
      mergedSeq[i] ← itemArray1[j];
      i ← i + 1;
      ENDLOOP;
    FOR j: CARDINAL IN [0..itemArray2.LENGTH) DO
      mergedSeq[i] ← itemArray2[j];
      i ← i + 1;
      ENDLOOP;
    RETURN[mergedSeq];
    END;

  END.


LOG
31-Aug-84 - Holbrook - AR 10679: GetContainerWindow should raise error if the the shell passed in doesn't have a container window.
25-Oct-84 - Holbrook - Take out SetBodyWindowJustFits; cw now handles this
16-Nov-84 - Holbrook - update to 4.0a
 5-Apr-85 - Holbrook - Fix AR 13882 by using ContainerWindowExtra.Create with readOnly param.
 6-Aug-85 - Holbrook - 15497 containerWindow access.  Added FileContainerShellExtra.CreateX to allow clients to use
 ContainerWindow.Access.
19-Jun-86 - Holbrook - Use FileContainerShellExtra3.CreateXX and ContainerWindowExtra4.CreateXX to allow extra procedures for background
container source
 9-Sep-86 - Holbrook - In TransitionProc, destroy container window before container source (Fixes Guzik crash of 19 Aug)
31-oct-86 - Holbrook - 9676 make container only as wide as columns
19-Aug-87 - LFB - AR 13906 - remove the call to SetContainee.
28-Oct-87 - jph - 15030 FileContainerShellExtra3.CreateX2 with considerShowingCoverSheet
```

```
-- File: FileContainerSource.mesa - last edit:
-- Breisacher.ES        11-Dec-84 14:26:34
-- Holbrook.ES           5-Aug-83 10:11:07

-- Copyright (C) 1984 by Xerox Corporation

DIRECTORY
  Containee USING [DataHandle, Implementation],
  ContainerSource USING [Handle, ItemIndex],
  NSFile USING [Attribute, AttributeType, Attributes, ExtendedAttributeType, Reference, Scope, Selections, Type],
  XString USING [Writer];

FileContainerSource: DEFINITIONS = BEGIN

  -- TYPEs

  AttributeFormatProc: TYPE = PROCEDURE [
    containeeImpl: Containee.Implementation,
    containeeData: Containee.DataHandle,
    attr: NSFile.Attribute,
    displayString: XString.Writer];

  ColumnType: TYPE = {attribute, extendedAttribute, multipleAttributes};

  ColumnContents: TYPE = LONG DESCRIPTOR FOR ARRAY OF ColumnContentsInfo;
    -- The columns will be displayed in the order given by this sequence.

  ColumnContentsInfo: TYPE = RECORD [
    info: SELECT type: ColumnType FROM
      attribute => [
        attr: NSFile.AttributeType,
        formatProc: AttributeFormatProc ← NIL,
        needsDataHandle: BOOLEAN ← FALSE],
      extendedAttribute => [
        extendedAttr: NSFile.ExtendedAttributeType,
        formatProc: AttributeFormatProc ← NIL,
        needsDataHandle: BOOLEAN ← FALSE],
      multipleAttributes => [
        attrs: NSFile.Selections,
        formatProc: MultiAttributeFormatProc ← NIL,
        needsDataHandle: BOOLEAN ← FALSE],
      ENDCASE];

  MultiAttributeFormatProc: TYPE = PROCEDURE [
    containeeImpl: Containee.Implementation,
    containeeData: Containee.DataHandle,
    attrRecord: NSFile.Attributes,
    displayString: XString.Writer];

  Options: TYPE = RECORD [
    readOnly: BOOLEAN ← FALSE];

  -- Procedures

  Create: PROCEDURE [
    file: NSFile.Reference,
    columns: ColumnContents,
    scope: NSFile.Scope ← [],
    options: Options ← []]
    RETURNS [source: ContainerSource.Handle];

  Info: PROCEDURE [
    source: ContainerSource.Handle]
    RETURNS [
    file: NSFile.Reference,
    columns: ColumnContents,
    scope: NSFile.Scope,
    options: Options];

  IsIt: PROCEDURE [source: ContainerSource.Handle] RETURNS [BOOLEAN];

  ChangeScope: PROCEDURE [source: ContainerSource.Handle, newScope: NSFile.Scope];
    -- This will typically be followed by a source.ActOn[relist], then a ContainerWindow.Update.

  GetItemInfo: PROCEDURE [
    source: ContainerSource.Handle,
    itemIndex: ContainerSource.ItemIndex]
    RETURNS [file: NSFile.Reference, type: NSFile.Type];

  -- Common column types

  IconColumn: PROCEDURE RETURNS [attribute ColumnContentsInfo];
  NameColumn: PROCEDURE RETURNS [attribute ColumnContentsInfo];
  SizeColumn: PROCEDURE RETURNS [multipleAttributes ColumnContentsInfo];
  DateColumn: PROCEDURE RETURNS [multipleAttributes ColumnContentsInfo];

  END.
```

```
-- File: FileContainerSourceImplA.mesa - last edit:
-- Holbrook            27-Jan-88 15:58:02
-- Breisacher           7-Dec-87 15:16:50
-- Holbrook.ES         10-Oct-86 12:24:23
-- SAJohnson.ES         8-Jul-85 10:35:27

-- Copyright (C) 1984, 1985, 1986, 1987, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  -- ButtonsAndLights,
  Atom USING [ATOM, MakeAtom],
  BWSMessages USING [GetMessageHandle, kaborted, kdiskPages, kobjects],
  Containee USING [Data, DataHandle, GetImplementation, Implementation, ReturnTicket, Ticket],
  ContaineeExtra USING [GetCachedNameX, GetCachedTypeX],
  ContainerCache USING [AddData, --AllocateCache,-- AppendItem, BeginFill, Clients,  DeleteNItems,
    FillProc, FreeCache, FreeMark, GetNthItem, Handle, IndexFromMark,
    ItemClients, ItemHandle,
    ItemNthString, Mark, MoveMark, Object, ResetCache, SetMark, StatusOfFill],
  ContainerCacheExtra USING [GetLength],
  ContainerCacheExtra2 USING [AllocateCache2],
  ContainerSource USING [ActOnProc, CanYouTakeProc, ChangeInfo,
    ChangeProc, ColumnCountProc, ConvertItemProc, DeleteItemsProc, Error,
    ErrorCode, GetLengthProc, Handle, ItemGenericProc, ItemIndex, nullItem,
    Procedures, ProceduresObject, StringOfItemProc, TakeProc],
  ContainerSourceExtra USING [GetGlobalChangeProcProc, Procedures, ProceduresObject, SetGlobalChangeProcProc],
  ContainerWindow USING [SourceModifyProc],
  Context USING [Create, Type, UniqueType],
  Courier USING [Error, ErrorCode],
  Cursor USING [Set],
  FileContainerSourceOps,
  FileContainerSource USING [AttributeFormatProc, ColumnContents, ColumnContentsInfo, MultiAttributeFormatProc, Options],
  FileContainerSourceExtra,
  FileContainerSourceExtra3,
  Heap USING [Create, FreeNode, MakeNode],
  NSAssignedTypes USING [FileType],
  NSFile USING [ascendingPositionOrdering, Attribute, Attributes, AttributesProc, AttributesRecord, AttributeType, Close, Controls,
  descendingPositionOrdering, Error, ErrorRecord, ExtendedAttributeType, ExtendedSelections, Filter, GetAttributes, Handle, ID, List,
  localSystemElement, MakeReference, noExtendedSelections, nullFilter, nullHandle, nullOrdering, nullReference, nullSession,
  OpenByReference, Ordering, Reference, Scope, Selections, Session, String, Type, Words],
  NSString USING [String],
  Process USING [GetCurrent, priorityBackground, priorityNormal, SetPriority],
  ServicesErrorMessage USING [MsgFromCourierError, MsgFromNSFileError],
  SessionCache USING [GetSession, ReturnSession],
  SpecialContainer,
  SpecialContainer2,
  SpecialContainer3,
  SpecialContainerCache USING [WaitCacheIdle],
  StarFileTypes USING [folder, reference],
  StarWindowShell USING [GetContainee, SetContainee],
  Wastebasket USING [Take],
  Window USING [Handle],
  XCharSet0 USING [Make],
  XFormat USING [Decimal, FormatProc, Handle, Object, Number, NumberFormat, WriterObject],
  XMessage USING [Get, Handle],
  XString USING [AppendChar, AppendReader, Character, ClearWriter,
   FreeWriterBytes, FromNSString, FromSTRING, NewWriterBody, nullReaderBody,
   Reader, ReaderBody, ReaderFromWriter, Writer, WriterBody],
  XTime USING [Append, dateAndTime];

FileContainerSourceImplA: MONITOR LOCKS fs.monitorLock USING fs: FileContainerSourceOps.FS
  IMPORTS Atom, BWSMessages, Containee, ContaineeExtra, ContainerCache, ContainerCacheExtra,  ContainerCacheExtra2,
  SpecialContainerCache, ContainerSource, Context, Courier, Cursor, FileContainerSourceOps, Heap, NSFile, Process, ServicesErrorMessage,
  SessionCache, StarWindowShell, Wastebasket, XCharSet0, XFormat, XMessage, XString, XTime
  EXPORTS FileContainerSource, FileContainerSourceOps, FileContainerSourceExtra, FileContainerSourceExtra3, SpecialContainer,
  SpecialContainer2, SpecialContainer3 =
  BEGIN OPEN FileContainerSourceOps, FileContainerSource;

 --for allocation of returned data in NoticeAppendedFiles
 -- Seq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF ContainerSource.EditInfo];

  -- Data

  z: PUBLIC UNCOUNTED ZONE ← Heap.Create [initial: 1];

  fileSourceProcs: ContainerSource.Procedures ← z.NEW [
    ContainerSource.ProceduresObject ← [
      actOn: ActOnFile,
      canYouTake: CanITake,
      columnCount: FileColumnCount,
      convertItem: ConvertFileItem,
      deleteItems: DeleteFileItems,
      getLength: GetFileLength,
      itemGeneric: FileItemGeneric,
      stringOfItem: StringOfFileItem,
      take: FileTake]];

  fileSourceExtraProcs: ContainerSourceExtra.Procedures ← z.NEW [
    ContainerSourceExtra.ProceduresObject ← [
      canYouTakeX: CanITakeX,
      takeX: FileTakeX,
      isBusy: IsBusy,
      setBusy: SetBusy,
      setMark: FSSetMark,
      freeMark: FSFreeMark,
      indexFromMark: FSIndexFromMark,
      moveOrCreateMark: FSMoveMark,
      setGlobalChangeProc: FSSetGlobal,

FileContainerSourceImplA.mesa        27-Jan-88 15:58:02 PST                                                                    1
```

```
    getGlobalChangeProc: FSGetGlobal]];

versionFormat: XFormat.NumberFormat ← [base: 10, zerofill: FALSE, signed: FALSE, columns: 5];
open, props, takeSelectionBackground, takeSelectionCopyBackground, canYouTakeSelection, takeSelection, takeSelectionCopy, freeMenu:
Atom.ATOM;

allSources: FS ← NIL; -- pointer to all sources
   -- for formatting version numbers in the version column
<< we want to lock the list of all sources during EnumerateSources, but a simple module monitor isn't good enough: we want to allow the
enumeration proc to crreate and destroy sources.  So we'll use Enter and Exit and allocate this dummy FileSourceObject.  >>

allSourcesEnumLock, allSourcesAddLock: FileSourceObject; -- 14807: use two locks

false: BOOLEAN ← FALSE;

changeProcContext: Context.Type ← Context.UniqueType[];

globalSession: NSFile.Session ← NSFile.nullSession; -- 14024 keep one session for all open file container source windows



-- PUBLIC Procedures

AboutToDestroySource: PUBLIC PROC [source: ContainerSource.Handle] = {
  fs: FS = ValidFileSource[source];
  RemoveSource[fs]};

ChangeScope: PUBLIC PROCEDURE [source: ContainerSource.Handle,
  newScope: NSFile.Scope] =
  BEGIN
  fs: FS = ValidFileSource[source];
  Enter[fs,, 200];
  fs.scope ← newScope;
  fs.nonNullScope ← ~IsScopeNull[newScope];
  Exit[fs, 200];
  END;

CheckNeedsDataHandle: PROCEDURE [columns: ColumnContents]
  RETURNS [needs: BOOLEAN ← FALSE] =
  BEGIN
  FOR i: CARDINAL IN [0..LENGTH[columns]) DO
    WITH col: columns[i] SELECT FROM
      multipleAttributes =>
        {IF col.needsDataHandle THEN needs ← TRUE; EXIT};
      attribute =>
        {IF col.needsDataHandle THEN needs ← TRUE; EXIT};
      extendedAttribute =>
        {IF col.needsDataHandle THEN needs ← TRUE; EXIT};
      ENDCASE;
    ENDLOOP;
  END; -- CheckNeedsDataHandle

IsScopeNull: PROCEDURE [s1: NSFile.Scope] RETURNS [null: BOOLEAN] = {
  nulls: NSFile.Scope ← [];
  IF    s1.count = nulls.count
    AND s1.depth = nulls.depth
    AND s1.filter = NSFile.nullFilter
     -- don't need to check ordering or direction; they makes no difference to
     -- number of files enumerated
    THEN RETURN [TRUE]
    ELSE RETURN [FALSE]};

Create: PUBLIC PROCEDURE [
  file: NSFile.Reference,
  columns: ColumnContents,
  scope: NSFile.Scope ← [],
  options: Options ← []]
  RETURNS [source: ContainerSource.Handle] =
  {RETURN [CreateX [file, columns, scope, options].source]};

sourceIndex: CARDINAL ← 0;
CreateX: PUBLIC PROCEDURE [
  file: NSFile.Reference,
  columns: ColumnContents,
  scope: NSFile.Scope ← [],
  options: Options ← []]
  RETURNS [source: ContainerSource.Handle, sourceX: ContainerSourceExtra.Procedures] =
  BEGIN
  fh: NSFile.Handle;
  session: NSFile.Session ← NSFile.nullSession;
  fs: FS ← NIL;
  attributes: NSFile.AttributesRecord ← TRASH;

  IF file = NSFile.nullReference THEN
    ERROR ContainerSource.Error [invalidParameters];
  BEGIN ENABLE
    UNWIND => {
       NSFile.Close[fh, session ! NSFile.Error, Courier.Error => CONTINUE];
       IF session # NSFile.nullSession THEN SessionCache.ReturnSession[session];
       IF fs # NIL THEN z.FREE[@fs]};

  session ← SessionCache.GetSession[file];
  fh ← NSFile.OpenByReference[file, [timeout: 10],session];
  fs ← z.NEW [
    FileSourceObject ← [
      procs: fileSourceProcs,
      columns: CopyColumns [columns],
```

```
        scope: scope,
        nonNullScope: ~IsScopeNull[scope],
        options: options,
        parentReference: file,
        parentHandle: fh,
        session: session,
        selections: MakeSelections [columns],
        globalChangeData: NIL,
        globalChangeProc: NIL,
        needsDataHandle: CheckNeedsDataHandle[columns],
        cache: ContainerCacheExtra2.AllocateCache2[useProcessAbort: FALSE],
        isRemote: file.service.systemElement # NSFile.localSystemElement,
        nextSource: NIL,
        containerWindow: NIL,
        id: sourceIndex]];
    sourceIndex ← sourceIndex+1;
    NSFile.GetAttributes[fh, [[ordering: TRUE, numberOfChildren: TRUE]], @attributes, session];
    fs.ascendingOrDescending ←
     (attributes.ordering = NSFile.ascendingPositionOrdering OR attributes.ordering = NSFile.descendingPositionOrdering);
    AddSource[fs]; -- add source to global list
    fs.length ←
      IF fs.nonNullScope
        THEN 0
        ELSE attributes.numberOfChildren;

    ContainerCache.BeginFill[fs.cache, FillCacheInBackground, fs];
    END;
    RETURN[@fs.procs, fileSourceExtraProcs];
    END;

  EnumerateSources: PUBLIC PROCEDURE [
    enumProc: FileContainerSourceExtra3.SourceEnumProc] = {

    Enter[@allSourcesEnumLock,, 300]; -- 14807
    BEGIN ENABLE UNWIND => Exit[@allSourcesEnumLock, 300];
    cur: FS ← allSources;
    next: FS;
    stop: BOOLEAN ← FALSE;
    WHILE cur # NIL AND ~stop DO
      next ← cur.nextSource;
      stop ← enumProc[LOOPHOLE[cur, ContainerSource.Handle]];
      cur ← next;
      ENDLOOP;
    Exit[@allSourcesEnumLock, 300];
    END; -- ENABLE
    }; -- EnumerateSources

  GetItemInfo: PUBLIC PROCEDURE [
    source: ContainerSource.Handle,
    itemIndex: ContainerSource.ItemIndex]
    RETURNS [file: NSFile.Reference, type: NSFile.Type] = {
    fs: FS = ValidFileSource[source];
    Enter[fs, lockIfNotBetweenCalls, 400];
    BEGIN ENABLE UNWIND => Exit[fs, 400];
    itemData: ItemFileDataHandle ← GetItemDataInternal [fs.cache, itemIndex];
    IF itemData = NIL THEN ERROR ContainerSource.Error[noSuchItem];
    file ←  NSFile.MakeReference [
      fileID: itemData.id,
      service: fs.parentReference.service];
    type ← itemData.type;
    Exit[fs, 400];
    END; -- enable
    };

  Info: PUBLIC PROCEDURE [source: ContainerSource.Handle]
    RETURNS [
    file: NSFile.Reference,
    columns: ColumnContents,
    scope: NSFile.Scope,
    options: Options] =
    BEGIN
    fs: FS = ValidFileSource[source];
    Enter[fs,, 500];
    file ← fs.parentReference;  columns ← DESCRIPTOR[fs.columns];
    scope ← fs.scope; options ← fs.options;
    Exit[fs, 500];
    END;

  InfoX: PUBLIC PROC [source: ContainerSource.Handle]
    RETURNS [sourceX: ContainerSourceExtra.Procedures] =
    BEGIN
    fs: FS = ValidFileSource[source];
    -- no need to enter: these can't be changed
    RETURN [fileSourceExtraProcs];
    END;

  IsIt: PUBLIC PROCEDURE [source: ContainerSource.Handle] RETURNS [BOOLEAN] =
    BEGIN
    IF source = NIL THEN RETURN[FALSE];
    RETURN[source↑ = fileSourceProcs];
    END;

  WaitSourceIdle: PUBLIC PROCEDURE [source: ContainerSource.Handle] = {
    fs: FS = ValidFileSource[source];
    SpecialContainerCache.WaitCacheIdle[fs.cache]};

  -- Common column types and their format procs
```

```
DateColumn: PUBLIC PROCEDURE RETURNS [column: multipleAttributes ColumnContentsInfo] =
  BEGIN
  dateSelections: NSFile.Selections ← [interpreted: [
    type: TRUE, createdOn: TRUE]];
  column ← [multipleAttributes[attrs: dateSelections, formatProc: DateFormatProc]];
  RETURN [column];
  END;

IconColumn: PUBLIC PROCEDURE RETURNS [column: attribute ColumnContentsInfo] =
  BEGIN
  column ← [attribute[attr: type, formatProc: IconFormatProc]];
  RETURN [column];
  END;

NameColumn: PUBLIC PROCEDURE RETURNS [column: attribute ColumnContentsInfo] =
  BEGIN
  column ← [attribute[attr: name, formatProc: NameFormatProc]];
  RETURN [column];
  END;

NameAndVersionColumn: PUBLIC PROCEDURE RETURNS [column: multipleAttributes ColumnContentsInfo] =
  BEGIN
  selections: NSFile.Selections ← [interpreted: [
    name: TRUE, version: TRUE]];
  column ← [multipleAttributes[attrs: selections, formatProc: NameVersionFormatProc]];
  RETURN [column];
  END;


SizeColumn: PUBLIC PROCEDURE RETURNS [column: multipleAttributes ColumnContentsInfo] =
  BEGIN
  sizeSelections: NSFile.Selections ← [interpreted: [
    isDirectory: TRUE, numberOfChildren: TRUE, subtreeSize: TRUE, type: TRUE]];
  column ← [multipleAttributes[attrs: sizeSelections, formatProc: SizeFormatProc]];
  RETURN [column];
  END;

VersionColumn: PUBLIC PROCEDURE RETURNS [column: attribute ColumnContentsInfo] =
  BEGIN
  column ← [attribute[attr: version, formatProc: VersionFormatProc]];
  RETURN [column];
  END;

DateFormatProc: MultiAttributeFormatProc =
  BEGIN
  rb: XString.ReaderBody;
  -- show date for everything except folders (AR 15310) and doc books and mail folders (ARs 15843 15845).
  mailFolder: NSFile.Type = 4417; -- AR 15845
  docBook: NSFile.Type = 4444; -- AR 15843
  IF attrRecord.type = StarFileTypes.folder OR attrRecord.type = mailFolder OR attrRecord.type = docBook THEN {
    rb ← XString.FromSTRING["---"L];
    XString.AppendReader[displayString, @rb]}
  ELSE
    -- old template: <2>-<6>-<4> <8>:<9>:<10>
    XTime.Append[displayString, attrRecord.createdOn, XTime.dateAndTime];
  END;

IconFormatProc: AttributeFormatProc =
  BEGIN
  smallPic: XString.Character;

  WITH a: attr SELECT FROM
    type =>
      smallPic ← containeeImpl.smallPictureProc[containeeData, a.value, normal];
    ENDCASE;

  XString.AppendChar[displayString, smallPic]; -- should catch something here
  END;

NameFormatProc: AttributeFormatProc =
  BEGIN
  name: NSString.String;
  rb: XString.ReaderBody;
  WITH a:attr SELECT FROM
    name => name ← a.value;
    ENDCASE;
  rb ← XString.FromNSString[name];
  XString.AppendReader[displayString, @rb];  -- should catch something here
  END;

NameVersionFormatProc: MultiAttributeFormatProc =
  BEGIN
  rb: XString.ReaderBody;
  writer: XFormat.Object ← XFormat.WriterObject [displayString];
  rb ← XString.FromNSString[attrRecord.name];
  XString.AppendReader[displayString, @rb];
  XString.AppendChar[to: displayString, c: XCharSet0.Make[exclamationPoint]];
  XFormat.Decimal[h: @writer, n: attrRecord.version];
  END;

SizeFormatProc: MultiAttributeFormatProc =
  BEGIN
  rb: XString.ReaderBody;
  writer: XFormat.Object ← XFormat.WriterObject [displayString];
  bwsmh: XMessage.Handle ← BWSMessages.GetMessageHandle[];
  --if it's a folder, then give the number of children; all others give pages
```

```
    IF attrRecord.type = StarFileTypes.folder THEN {
      XFormat.Decimal[h: @writer, n: attrRecord.numberOfChildren];
      rb ← XMessage.Get[bwsmh, BWSMessages.kobjects];
      XString.AppendReader[displayString, @rb]}
    ELSE {
      XFormat.Decimal[h: @writer, n: attrRecord.subtreeSize];
      rb ← XMessage.Get[bwsmh, BWSMessages.kdiskPages];
      XString.AppendReader[displayString, @rb]}
    END;

   padVersion: BOOLEAN ← TRUE;

   VersionFormatProc: AttributeFormatProc =
    BEGIN
    writer: XFormat.Object ← XFormat.WriterObject [displayString];
    WITH a:attr SELECT FROM
      version => XFormat.Number[h: @writer, n: a.value, format: versionFormat];
    ENDCASE;
    END;

  -- ContainerSource Procedures

  ActOnFile: ContainerSource.ActOnProc =
    BEGIN
    fs: FS = ValidFileSource[source];
    SELECT action FROM
      destroy => {
        RemoveSource[fs]; -- 15592 remove the source before we enter the monitor,
        Enter[fs,, 600];
        BEGIN ENABLE UNWIND => Exit[fs, 600];
        ContainerCache.FreeCache [fs.cache];--this will stop any enumeration in progress
        FreeColumns [fs.columns];
        FreeSelections [fs.selections];
        IF fs.parentHandle # NSFile.nullHandle THEN
          NSFile.Close[fs.parentHandle, fs.session ! NSFile.Error, Courier.Error => CONTINUE];
        IF fs.session # NSFile.nullSession THEN SessionCache.ReturnSession[fs.session];
        Exit[fs, 600]; -- not sure this is right
        END; -- enable
        z.FREE[@source]};
      reList => {
        Enter[fs,, 700];
        BEGIN ENABLE UNWIND => Exit[fs, 700];
        fs.length ← CARDINAL.LAST; -- 13756 force recalculation of length
        fs.length ← IF fs.nonNullScope THEN 0 ELSE GetFileLength[LOOPHOLE[fs]].length;
        fs.enumerationFinished ← FALSE;
        ContainerCache.ResetCache[fs.cache];
        ContainerCache.BeginFill[fs.cache, FillCacheInBackground, fs];
        Exit[fs, 700];
        END; -- enable
        };

      sleep, wakeup => NULL;
      ENDCASE => ERROR;
    END;

  FileColumnCount: ContainerSource.ColumnCountProc =
    <<[source: Handle]
    RETURNS [columns: CARDINAL]  >>
    {RETURN[ValidFileSource[source].columns.length]};

  StringOfFileItem: ContainerSource.StringOfItemProc =
  << source: Handle,
    itemIndex: ItemIndex,
    stringIndex: CARDINAL]
    RETURNS [XString.ReaderBody]>>
    BEGIN
    fs: FS = ValidFileSource[source];
    rb: XString.ReaderBody;
    Enter[fs, lockIfNotBetweenCalls, 800];
    BEGIN ENABLE UNWIND => Exit[fs, 800];
    h: ContainerCache.ItemHandle ← ContainerCache.GetNthItem [fs.cache, itemIndex];

    IF h = NIL THEN ERROR ContainerSource.Error[noSuchItem];
    rb ← ContainerCache.ItemNthString[h, stringIndex];
    Exit[fs, 800];
    END; -- enable
    RETURN[rb];
    END;


  -- Internal Procedures

  AddSource: PUBLIC PROC [fs: FS] = {
    Enter[@allSourcesAddLock];  -- 14807 only use an add lock
    fs.nextSource ← allSources;
    allSources ← fs;
    Exit[@allSourcesAddLock]};

  AllocateWriters: PUBLIC PROCEDURE [size: CARDINAL]
    RETURNS [writers: LONG POINTER TO WriterSeq] = {
    writers ← z.NEW[WriterSeq[size]];
    FOR i: CARDINAL IN [0..size) DO
      writers[i] ← XString.NewWriterBody[200, z];
      ENDLOOP;
    };

  FreeWriters: PUBLIC PROCEDURE [writers: LONG POINTER TO WriterSeq] = {
```

```
    FOR i: CARDINAL IN [0..writers.length) DO
      XString.FreeWriterBytes[@writers[i]];
      ENDLOOP;
    z.FREE [@writers];
    };

  AppendMessageToCache: PROC [fs: FS, cache: ContainerCache.Handle, msg: XString.Reader] =
    BEGIN
    strings: ReaderSeqPtr ← z.NEW [ReaderSeq[fs.columns.length]];
    addData: ContainerCache.AddData;

    FOR i: CARDINAL IN [0..fs.columns.length) DO
      strings[i] ← XString.nullReaderBody;
      ENDLOOP;

    << total hack: if there are two or more columns, put message in second column
         (the first column is often an icon column, and is too short for a
         message >>
    << comment out lock code: we can get hung up if another process is during
       an StringOfFileItem call and has everything locked.  >>
    -- Enter[fs,, 900];
    -- BEGIN ENABLE UNWIND => Exit[fs, 900];
    IF fs.columns.length >= 2
      THEN strings[1] ← msg↑
      ELSE strings[0] ← msg↑;
    addData ← [
      clientData: NIL,
      clientDataCount: 0,
      clientStrings: DESCRIPTOR[strings]];
    [] ← ContainerCache.AppendItem [cache, addData];
    -- Exit[fs, 900];
    -- END; ** enable
    z.FREE [@strings];
    END;

  AttributeFromAttributeRecord: PROCEDURE [
    attributes: NSFile.Attributes,
    attrType: NSFile.AttributeType]
    RETURNS [attr: NSFile.Attribute] = BEGIN
    SELECT attrType FROM
      checksum => attr ← [checksum[attributes.checksum]];
      childrenUniquelyNamed => attr ← [childrenUniquelyNamed[attributes.childrenUniquelyNamed]];
      createdBy => attr ← [createdBy[attributes.createdBy]];
      createdOn => attr ← [createdOn[attributes.createdOn]];
      fileID => attr ← [fileID[attributes.fileID]];
      isDirectory => attr ← [isDirectory[attributes.isDirectory]];
      isTemporary => attr ← [isTemporary[attributes.isTemporary]];
      modifiedBy => attr ← [modifiedBy[attributes.modifiedBy]];
      modifiedOn => attr ← [modifiedOn[attributes.modifiedOn]];
      name => attr ← [name[attributes.name]];
      numberOfChildren => attr ← [numberOfChildren[attributes.numberOfChildren]];
      ordering => attr ← [ordering[attributes.ordering]];
      parentID => attr ← [parentID[attributes.parentID]];
      position => attr ← [position[attributes.position]];
      readBy => attr ← [readBy[attributes.readBy]];
      readOn => attr ← [readOn[attributes.readOn]];
      sizeInBytes => attr ← [sizeInBytes[attributes.sizeInBytes]];
      type => attr ← [type[attributes.type]];
      version => attr ← [version[attributes.version]];
      accessList => attr ← [accessList[attributes.accessList]];
      defaultAccessList => attr ← [defaultAccessList[attributes.defaultAccessList]];
      pathname => attr ← [pathname[attributes.pathname]];
      service => attr ← [service[attributes.service]];
      backedUpOn => attr ← [backedUpOn[attributes.backedUpOn]];
      filedBy => attr ← [filedBy[attributes.filedBy]];
      filedOn => attr ← [filedOn[attributes.filedOn]];
      sizeInPages => attr ← [sizeInPages[attributes.sizeInPages]];
      subtreeSize => attr ← [subtreeSize[attributes.subtreeSize]];
      subtreeSizeLimit => attr ← [subtreeSizeLimit[attributes.subtreeSizeLimit]];
      ENDCASE;
    END;

  BuildRow: PUBLIC PROCEDURE [
    fs: FS,
    writers: LONG POINTER TO WriterSeq,
    readers: LONG POINTER TO ReaderSeq,
    itemData: ItemFileDataHandle,
    containeeData: Containee.DataHandle,
    attributes: NSFile.Attributes]
    RETURNS [addData: ContainerCache.AddData] =
    BEGIN
    attr: NSFile.Attribute;
    ci: Containee.Implementation ← Containee.GetImplementation [attributes.type];

    << we pass in a pointer to a ItemFileDataHandle so that we can pass it
       back as part of the ContainerCache.AddData.  This is not great, but
       it saves us having to allocate some storage. >>

    FOR i: CARDINAL IN [0..fs.columns.length) DO
      XString.ClearWriter [@writers[i]];
      WITH column: fs.columns[i] SELECT FROM
        attribute => {
          attr ← AttributeFromAttributeRecord [
            attributes, column.attr];
          column.formatProc [ci, containeeData, attr, @writers[i]];};
        extendedAttribute => {
          attr ← ExtendedAttributeFromAttributeRecord [
```

```
            attributes, column.extendedAttr];
          column.formatProc [ci, containeeData, attr, @writers[i]];};
        multipleAttributes =>
          column.formatProc [ci, containeeData, attributes, @writers[i]];
        ENDCASE;
      ENDLOOP;

    itemData↑ ← [
      id: attributes.fileID,
      type: attributes.type];

    FOR i: CARDINAL IN [0..writers.length) DO
      readers[i] ← (XString.ReaderFromWriter [@writers[i]])↑;
      ENDLOOP;

    addData ← [
      clientData: itemData,
      clientDataCount: SIZE[ItemFileData],
      clientStrings: DESCRIPTOR[readers]];

    RETURN[addData];
    END;

  CopyColumns: PROCEDURE [columns: ColumnContents]
    RETURNS [newColumns: ColumnContentsSeqHandle] = BEGIN
    oldExtended: NSFile.ExtendedSelections;
    newColumns ← z.NEW[ColumnContentsSeq[LENGTH[columns]]];
    FOR i: CARDINAL IN [0..LENGTH[columns]) DO
      newColumns[i] ← columns[i];
      WITH oldColumns: columns[i] SELECT FROM
        multipleAttributes => oldExtended ← oldColumns.attrs.extended;
        ENDCASE;

      WITH multCol: newColumns[i] SELECT FROM
        multipleAttributes =>
          BEGIN
          IF oldExtended # NSFile.noExtendedSelections THEN {
            multCol.attrs.extended ←
              MakeExtendedSelections[LENGTH[oldExtended]];
            FOR i: CARDINAL IN [0..LENGTH[oldExtended]) DO
              multCol.attrs.extended[i] ← oldExtended[i];
              ENDLOOP;
            }
          ELSE multCol.attrs.extended ← oldExtended;
          END;
        ENDCASE;

      ENDLOOP;
    END; -- CopyColumns

  DeleteFileItems: PUBLIC ContainerSource.DeleteItemsProc =
    BEGIN

    -- deleteControls: NSFile.Controls ← [lock: exclusive, timeout: 0, access: [remove: TRUE, write: TRUE]];

    fs: FS = ValidFileSource[source];
    itemData: ItemFileDataHandle;
    childReference: NSFile.Reference;
    changeInfo: ContainerSource.ChangeInfo;
    i, thisItem: ContainerSource.ItemIndex;
    somethingDeleted: BOOLEAN ← FALSE;
    inLock: BOOLEAN;
    firstMark, thisMark, nextMark: ContainerCache.Mark;

    Update: PROC [lastItem: ContainerCache.Mark, includesLast: BOOLEAN] = {
      << 14545: if includesLast TRUE, then lastItem is actually the last item, not last item +1; if FALSE, it's lastItem+1 >>
      last: ContainerSource.ItemIndex ← ContainerCache.IndexFromMark[lastItem];
      first: ContainerSource.ItemIndex ← ContainerCache.IndexFromMark[firstMark];
      itemsDeleted: CARDINAL ← last-first + (IF includesLast THEN 1 ELSE 0);
      IF itemsDeleted # 0 THEN {
        ContainerCache.DeleteNItems [cache: fs.cache, item: first, nitems: itemsDeleted];
        fs.length ← fs.length - itemsDeleted;
        IF changeProc # NIL THEN {
          changeInfo ← [var: delete[deleteInfo:
            [afterItem: first, nItems: itemsDeleted]]];
          changeProc[changeProcData, changeInfo]}};
      };


    Enter[fs, lockIfNotBetweenCalls, 1000]; inLock ← TRUE;
    BEGIN ENABLE
      UNWIND => {
        IF ~inLock THEN Enter[fs,, 1010];
        IF somethingDeleted THEN Update[thisMark, FALSE];
        ContainerCache.FreeMark[firstMark]; ContainerCache.FreeMark[nextMark];
        ContainerCache.FreeMark[thisMark];
        Exit[fs, IF inLock THEN 1000 ELSE 1010] };

    confirmationOk: BOOLEAN ← TRUE;


    firstMark ← ContainerCache.SetMark[fs.cache, itemIndex];
    nextMark ← ContainerCache.SetMark[fs.cache, itemIndex];
    thisMark ← ContainerCache.SetMark[fs.cache, itemIndex];
    -- Delete each item from the parent container file
    FOR i IN [0..n) DO
      thisItem ← ContainerCache.IndexFromMark[nextMark];
```

```
      ContainerCache.MoveMark[thisMark, thisItem];
      ContainerCache.MoveMark[nextMark, thisItem+1];
      itemData ← GetItemDataInternal [fs.cache, thisItem];
      IF itemData = NIL THEN LOOP;
      inLock ← FALSE; Exit[fs, 1000];
      childReference ← NSFile.MakeReference[itemData.id, fs.parentReference.service];
      IF Wastebasket.Take[childReference, confirmationOk]
        -- don't confirm: let CW do that
        THEN somethingDeleted ← TRUE
        ELSE EXIT;  -- didn't confirm: getout
      IF confirmationOk THEN {
        Cursor.Set[hourGlass];
        -- set cursor back to hourglass in case we just confirmed.
        confirmationOk ← FALSE};
        -- don't confirm after first icon, just delete all
      inLock ← TRUE; Enter[fs,, 1000];
      ENDLOOP;

    IF somethingDeleted THEN {
      IF ~inLock THEN {inLock ← TRUE; Enter[fs,, 1000]};
      Update[thisMark, TRUE]; -- 14545 can't use nextMark because that item might have been removed in another process
      };
    ContainerCache.FreeMark[firstMark]; ContainerCache.FreeMark[nextMark];
    ContainerCache.FreeMark[thisMark];
    IF inLock THEN Exit[fs, 1000];
    END;  -- ENABLE
    END; -- DeleteFileItems

  DestroyContext: PROCEDURE [mydata: LONG POINTER TO ChangeProcData, window: Window.Handle] = {
    ContainerCache.FreeMark[mydata.item];
    z.FREE[@mydata]};

  ExtendedAttributeFromAttributeRecord: PROCEDURE [
    attributes: NSFile.Attributes,
    attrType: NSFile.ExtendedAttributeType]
    RETURNS [attr: NSFile.Attribute] = BEGIN
    FOR i: CARDINAL IN [0..LENGTH[attributes.extended]) DO
      IF attributes.extended[i].type = attrType THEN
      BEGIN
        attr ← [extended[attrType, attributes.extended[i].value]];
        RETURN;
      END;
    ENDLOOP;
    END;

  waitCount: CARDINAL ← 0;
  -- Locking procs
  enterCount, exitCount: CARDINAL ← 0;
  EnterOrExit: TYPE = {enter, exit};

  <<
  MaxStack: CARDINAL = 4000;
  StackRecord: TYPE = RECORD [fsid: CARDINAL[0..256), process: CARDINAL[0..256), id: CARDINAL[0..4096), count: CARDINAL[0..8), type:
  EnterOrExit, how: EnterType];
  StackSeq: TYPE = RECORD [SEQUENCE length: CARDINAL OF StackRecord];
  seq: LONG POINTER TO StackSeq ← z.NEW[StackSeq[MaxStack]];
  trace: LONG DESCRIPTOR FOR ARRAY CARDINAL OF StackRecord ← DESCRIPTOR[seq];
  tracePointer: CARDINAL ← 0;
  >>

  Enter: PUBLIC ENTRY PROC [fs: FS, how: EnterType ← normal, id: CARDINAL ← 0] =
    -- id is for debugging
    BEGIN ENABLE UNWIND => {};
    me: PROCESS = Process.GetCurrent[];
    enterCount ← enterCount+1;
    SELECT fs.lock.process FROM
      me =>
        {IF ~(how = lockIfNotBetweenCalls AND fs.lock.lockedBetweenCalls) THEN {
          fs.lock.entryCount ← fs.lock.entryCount + 1;
                -- fs.idStack[fs.lock.entryCount] ← [id, FALSE];
           IF  ~fs.lock.lockedBetweenCalls AND how = getBetweenCallsLock THEN {
             fs.lock.lockedBetweenCalls ← TRUE;
                   -- fs.idStack[fs.lock.entryCount].betweenCallsOk ← TRUE
             };
                --  ButtonsAndLights.SetBar[3, fs.lock.entryCount];
           }
         ELSE {fs.lock.lockedBetweenCalls ← FALSE;
                  --  fs.idStack[fs.lock.entryCount].betweenCallsOk ← TRUE
               };
               <<
               IF id # 800 THEN {
                 trace[tracePointer] ← [fs.id, LOOPHOLE[me], id, fs.lock.entryCount, enter, how];
                 tracePointer ← tracePointer+1;
                 IF tracePointer=MaxStack THEN tracePointer←0; };
               >>
         RETURN};
      NIL => NULL;
      ENDCASE => {
        waitCount ← waitCount+1;
              -- ButtonsAndLights.SetBar[4, waitCount];
              -- IF waitCount = 1 THEN ButtonsAndLights.SetLight[3, TRUE];
        WHILE fs.lock.process # NIL DO WAIT fs.lock.exiting; ENDLOOP;
        waitCount ← waitCount-1;
              -- IF waitCount = 0 THEN ButtonsAndLights.SetLight[3, FALSE];
              -- ButtonsAndLights.SetBar[4, waitCount]
        };
```

```
        --loop not needed since Exit is the only way this condition variable can get notified (timeouts are not enabled)
    fs.lock.process ← me;
    fs.lock.entryCount ← 1;
        -- fs.idStack[1] ← [id, FALSE];
    fs.lock.lockedBetweenCalls ← how = getBetweenCallsLock;
        <<
        IF id = 800 THEN RETURN;
        trace[tracePointer] ← [fs.id, LOOPHOLE[me], id, fs.lock.entryCount, enter, how];
        tracePointer ← tracePointer+1;
        IF tracePointer=MaxStack THEN tracePointer←0;
        >>
    END;

  UnbalancedFileContainerSourceLocks: SIGNAL = CODE;
  -- UnmatchedContainerSourceLocks: SIGNAL = CODE;

  Exit: PUBLIC ENTRY PROC [fs: FS, id: CARDINAL ← 0] =
    BEGIN ENABLE UNWIND => {};
    me: PROCESS = Process.GetCurrent[];
        --savedId: CARDINAL;
    exitCount ← exitCount+1;
    -- fs.lock.lockedBetweenCalls ← FALSE;
    << I'm not real sure whether we need to turn off lockedBetweenCalls right here or not.  I put it in, but then discovered a situation
    in CWImplC propsDown code where we do a CS Lock, a CW Lock, then a VerifySelection to lock the source between calls, then an CS
    unlock.  The point is that we needed to lock the CW, so we had to lock the source first to prevent possible deadlocks; but then we
    wanted to release the source lock as soon as the CS props code was called.  >>
    IF fs.lock.entryCount = 0 THEN
      SIGNAL UnbalancedFileContainerSourceLocks[];
        <<
        savedId ← fs.idStack[fs.lock.entryCount].id;
        IF  ~(savedId = id OR
          (fs.idStack[fs.lock.entryCount].betweenCallsOk AND (savedId = 100))) THEN
           SIGNAL UnmatchedContainerSourceLocks[];
        >>
    fs.lock.entryCount ← fs.lock.entryCount - 1;
        -- ButtonsAndLights.SetBar[3, fs.lock.entryCount];
        <<
        IF id # 800 THEN {
          trace[tracePointer] ← [fs.id, LOOPHOLE[me], id, fs.lock.entryCount, exit, normal];
          tracePointer ← tracePointer+1;
          IF tracePointer=MaxStack THEN tracePointer←0};
        >>
    IF fs.lock.entryCount = 0 THEN
      {fs.lock.process ← NIL;
      NOTIFY fs.lock.exiting};
        --must not be BROADCAST; only the next process on the queue should be allowed to run
    END;

  newReferenceIcon: NSFile.Type = 4427; -- From ProtoStarFileTypes.mesa

  FillCacheInBackground: PUBLIC ContainerCache.FillProc =
   <<[cache: Handle] RETURNS [errored: BOOLEAN ← FALSE]>>
    BEGIN
    fs: FS ← ContainerCache.Clients[cache];
    caughtError: BOOLEAN ← FALSE;
    errorMsg: XString.ReaderBody;
    writers: WriterSeqPtr ← AllocateWriters [fs.columns.length];
    readers: ReaderSeqPtr ← z.NEW [ReaderSeq[fs.columns.length]];
    postCacheMessage: XFormat.Handle ← @formatObject;
    formatObject: XFormat.Object ← [proc: PostCacheMessage];
    itemNumber: CARDINAL ← 0; --to tell when the first 25 guys are done for process stuff
    retryCount: CARDINAL ← 0;

    PostCacheMessage: XFormat.FormatProc = {
      AppendMessageToCache[fs, cache, r]};

    Enumerator: NSFile.AttributesProc = BEGIN
      itemData: ItemFileData;
      addData: ContainerCache.AddData;
      data: Containee.Data ← [NSFile.nullReference];

      IF fs.needsDataHandle OR attributes.type = StarFileTypes.reference OR attributes.type = newReferenceIcon THEN
        data ← [NSFile.MakeReference[fileID: attributes.fileID,
          service: fs.parentReference.service]];

      SELECT ContainerCache.StatusOfFill[cache] FROM
        inProgress => NULL;
        inProgressPendingAbort, inProgressPendingJoin => RETURN[FALSE];
        ENDCASE => ERROR;

      addData ← BuildRow [fs, writers, readers, @itemData, @data, attributes];
      [] ← ContainerCache.AppendItem [cache, addData];
     --  Enter[fs];
      -- BEGIN ENABLE UNWIND => Exit[fs];

      IF fs.nonNullScope THEN
        fs.length ← ContainerCacheExtra.GetLength[cache];

      -- Exit[fs];
      -- END; enable

      itemNumber ← itemNumber + 1;
      IF itemNumber > 24 THEN --reset the priority --
         Process.SetPriority [Process.priorityBackground];
      RETURN;
      END;
```

```
      BEGIN ENABLE {
        ABORTED => GOTO Aborted;
        NSFile.Error => {
          IF error = [handle[obsolete]] THEN
            IF RevalidateParent[fs, retryCount] THEN {
              retryCount ← retryCount+1;
              RETRY};
          ServicesErrorMessage.MsgFromNSFileError[error, postCacheMessage];
          GOTO GetOut};
        Courier.Error => {
          ServicesErrorMessage.MsgFromCourierError[errorCode, postCacheMessage];

          GOTO GetOut};
        }; -- ENABLE

        --set the priority of the process to normal for the first 25 items, then to background
        Process.SetPriority [Process.priorityNormal];
        NSFile.List [ directory: fs.parentHandle, proc: Enumerator,
          selections: fs.selections, scope: fs.scope, session: fs.session];
        fs.enumerationFinished ← TRUE;
        EXITS
          Aborted => {
            errorMsg ← XMessage.Get[BWSMessages.GetMessageHandle[],
               BWSMessages.kaborted];
            AppendMessageToCache [fs, cache, @errorMsg];
            fs.length ← ContainerCacheExtra.GetLength[fs.cache]; -- 16433
            errored ← TRUE; };
          GetOut => {
            fs.length ← ContainerCacheExtra.GetLength[fs.cache]; -- 16433
            errored ← TRUE};
        END;

      z.FREE [@readers];
      FreeWriters [writers];

      RETURN;
      END;

    FSGetGlobal: ContainerSourceExtra.GetGlobalChangeProcProc = {
      fs: FS = ValidFileSource[source];
      RETURN [fs.globalChangeProc, fs.globalChangeData, fs.containerWindow]};

    FSSetGlobal: ContainerSourceExtra.SetGlobalChangeProcProc = {
      fs: FS = ValidFileSource[source];
      fs.globalChangeProc ← changeProc; fs.globalChangeData ← data;
      fs.containerWindow ← window};

    GetFileLength: PUBLIC ContainerSource.GetLengthProc =
      BEGIN
      fs: FS = ValidFileSource[source];
      attributes: NSFile.AttributesRecord ← TRASH;
      raiseError: XFormat.Object ← [proc: RaiseErrorProc];
      nsErrorRecord: NSFile.ErrorRecord;
      coError: Courier.ErrorCode;
      wasNsError: BOOLEAN;
      signal: --GENERIC-- SIGNAL ← NIL;
      retryCount: CARDINAL ← 0;

      RaiseErrorProc: XFormat.FormatProc = {
        ERROR ContainerSource.Error[code: fileError, msg: r, error: signal, errorData:
          IF wasNsError THEN @nsErrorRecord ELSE @coError];
        };

      Enter[fs,, 1100];
      BEGIN ENABLE UNWIND => Exit[fs, 1100];
      IF fs.length # CARDINAL.LAST THEN {
        length ← fs.length;
        totalOrPartial ← IF fs.nonNullScope AND ~fs.enumerationFinished THEN partial ELSE total;
        Exit[fs, 1100];
        RETURN};
      IF fs.nonNullScope THEN {
        fs.length ← length ← ContainerCacheExtra.GetLength[fs.cache];
        totalOrPartial ← IF fs.enumerationFinished THEN total ELSE partial;
        Exit[fs, 1100];
        RETURN};

      BEGIN ENABLE
        BEGIN
        Courier.Error => {
          coError ← errorCode;
          signal ← LOOPHOLE[Courier.Error, SIGNAL];
          wasNsError ← FALSE;
          GOTO Error};
        NSFile.Error => {
          IF error = [handle[obsolete]] THEN
            IF RevalidateParent[fs, retryCount] THEN {
              retryCount ← retryCount+1;
              RETRY};
          nsErrorRecord ← error;
          signal ← LOOPHOLE[NSFile.Error, SIGNAL];
          wasNsError ← TRUE;
          GOTO Error};
        END;

      NSFile.GetAttributes[
        file: fs.parentHandle,
        selections: [interpreted: [numberOfChildren: TRUE]],
```

```
      attributes: @attributes,
      session: fs.session ]; <<catch phrase!!>>
    fs.length ← attributes.numberOfChildren;
    length ← attributes.numberOfChildren;
    Exit[fs, 1100];
    EXITS
      Error => IF wasNsError THEN
        ServicesErrorMessage.MsgFromNSFileError[nsErrorRecord,
          @raiseError]
        ELSE ServicesErrorMessage.MsgFromCourierError[coError,
          @raiseError];
    END; -- enable NSFile, Courier errors
    END; -- enable Exit
    END; -- GetFileLength

  FileItemGeneric: PUBLIC ContainerSource.ItemGenericProc =
  <<[source: ContainerSource.Handle,
    itemIndex: ContainerSource.ItemIndex,
    atom: Atom.ATOM,
    changeProc: ContainerSource.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL]
    RETURNS [LONG UNSPECIFIED]>>
    BEGIN
    fs: FS = ValidFileSource[source];
    ci: Containee.Implementation ← TRASH;
    cd: Containee.Data;
    myChangeProcData: LONG POINTER TO ChangeProcData;
    lu: LONG UNSPECIFIED;
    backgroundCopy: BOOLEAN ← atom = takeSelectionBackground OR atom = takeSelectionCopyBackground;
    inLock: BOOLEAN;
    itemData: ItemFileDataHandle;

    Enter[fs, lockIfNotBetweenCalls, 1200];
    inLock ← TRUE;
    BEGIN ENABLE UNWIND => IF inLock THEN Exit[fs, 1200];
    itemData ← GetItemDataInternal [fs.cache, itemIndex];

    IF itemData = NIL THEN {Exit[fs, 1200]; RETURN[LONG[NIL]]};

    -- Get the implementation for this type of item.
    ci ← Containee.GetImplementation[itemData.type];

    -- Construct a Containee.Data (NSFile.Reference) using
    --  the systemElement and volumeID of the parent (directory) file.
    cd ← [
      reference: NSFile.MakeReference[
        fileID: itemData.id,
        service: fs.parentReference.service]];

    myChangeProcData ← z.NEW [
      ChangeProcData ← [
        fs: fs,
        item: ContainerCache.SetMark[fs.cache, itemIndex],
        background: backgroundCopy,
        callersChangeProc: changeProc,
        callersChangeProcData: changeProcData ] ];

    Exit[fs, 1200]; inLock ← FALSE;
    IF backgroundCopy THEN {
      -- set name of item we're copying to
      name: XString.ReaderBody;
      ticket: Containee.Ticket;
      [name, ticket] ← IconAndNameFromFile[fs, @cd];
      SetNameForProcess[name];
      Containee.ReturnTicket[ticket]};

    myChangeProcData.attachedToContext ← TRUE; --what a kludge see AR 15968
    -- Call the item's generic proc
    lu ← IF ci.genericProc # NIL THEN
      IF atom = freeMenu THEN
      ci.genericProc[atom, @cd, NIL, changeProcData] ELSE
      ci.genericProc[atom, @cd, SingleItemChangeProc, myChangeProcData]
      ELSE LONG[NIL];
      -- if signal, free proc must be called

    IF (LOOPHOLE[lu, LONG POINTER] # NIL) AND ((atom = open) OR (atom = props))
      THEN {
        << problem: normally we destroy the changeProcData when the change proc
           is called.  However, if this is a psheet, and the user bugs Apply, the
           change proc will be called and the data will be destroyed.  If the user
           bugs apply again, we crash because the data is gone.

           Real solution would be to have a changeProc data free proc.
           Temp hack: create a window context when we see a psheet going up.
           This way, we let the context destroy proc free the data.  We keep a
           boolean in the change proc data let us know whether or not we are
           playing this trick so we can destroy the data in the normal case. >>
        myChangeProcData.attachedToContext ← TRUE;
        Context.Create[changeProcContext, myChangeProcData, DestroyContext, LOOPHOLE[lu]];
        -- 12615 don't SetContainee if it's already set
        IF StarWindowShell.GetContainee[LOOPHOLE[lu]].reference =  NSFile.nullReference THEN
          StarWindowShell.SetContainee[LOOPHOLE[lu], @cd];
        }
    ELSE myChangeProcData.attachedToContext ← FALSE;

    RETURN[
      IF LOOPHOLE[lu, LONG POINTER] = NIL THEN
```

```
        SELECT atom FROM
          canYouTakeSelection,
          takeSelection,
          takeSelectionCopy => @false,
          ENDCASE => LONG[NIL]
        ELSE lu ];
    END; -- enable
    END;

  IconAndNameFromFile: PROC [fs: FS, data: Containee.DataHandle]
    RETURNS [rb: XString.ReaderBody, ticket: Containee.Ticket] = {
    name: XString.ReaderBody;

    [name, ticket] ← ContaineeExtra.GetCachedNameX[data: data, session: fs.session];
    RETURN[IconAndName[data, @name, ContaineeExtra.GetCachedTypeX[data: data, session: fs.session], FALSE], ticket];
      }; -- IconAndNameFromFile


    << NoticeAppendedFiles calls FileContainerSource and informs it that files have been appended to the end of the source.  It returns a
    ContainerSource.ChangeInfo record with variant insert and info about which item the new items were inserted after (the last one). BIG
    NOTE: This ACTUALLY does NOT return the insert variant because of problems with allocating and deallocating from zones. Briefly, if
    the insert variant is passed back, some storage must be allocated. ContainerWindow is the only guy that ever sees the changedInfo
    and is, therefore, the guy who has to deallocate the storage. BUT if inserts come from various places, we need a convention that
    says that all allocations must come from zone x so that the container window always knows where to deallocate it from. Because of
    this problem, this proc will actually return the all variant. We do need to fix this in the future, however. [SAJ July 8, 1985] >>

  --the items below that are commented out should be reinstated when the insert variant is passed back.

  NoticeAppendedFiles: PUBLIC ContainerWindow.SourceModifyProc =
  <<PROCEDURE [window: Window.Handle, source: ContainerSource.Handle] RETURNS [changeInfo: ContainerSource.ChangeInfo]>>
  BEGIN
   fs: FS = ValidFileSource[source];
   scope: NSFile.Scope ← fs.scope; --need to change the filter slightly

  Enter[fs,, 1300]; -- enter before we do anything else
  BEGIN ENABLE UNWIND => Exit[fs, 1300];

   item: ContainerSource.ItemIndex ← ContainerCacheExtra.GetLength[fs.cache];
   ref: NSFile.Reference ← IF item # 0 THEN GetItemInfo[source, item-1].file ELSE NSFile.nullReference;
   handle: NSFile.Handle ← IF ref # NSFile.nullReference THEN NSFile.OpenByReference[ref,, fs.session] ELSE NSFile.nullHandle;
   attr: NSFile.AttributesRecord;
   posSel: NSFile.Selections ← [interpreted: [position: TRUE]];
  -- shellZone: UNCOUNTED ZONE ← StarWindowShell.GetZone[StarWindowShell.ShellFromChild[window]];
   --the following is allocated out of the shellZone since the client has to free it.
  -- itemArray: Seq ← shellZone.NEW[Seq[1] ← [ [IF item = 0 THEN item ELSE item-1, 0] ]];
   writers: WriterSeqPtr ← AllocateWriters [fs.columns.length];
   readers: ReaderSeqPtr ← z.NEW [ReaderSeq[fs.columns.length]];
    errorMsg: XString.ReaderBody;
   postCacheMessage: XFormat.Handle ← @formatObject;
   formatObject: XFormat.Object ← [proc: PostCacheMessage];
   retryCount: CARDINAL ← 0;

    PostCacheMessage: XFormat.FormatProc = {
      AppendMessageToCache[fs, fs.cache, r]};

   Enumerator: NSFile.AttributesProc = BEGIN
      itemData: ItemFileData;
      addData: ContainerCache.AddData;
      data: Containee.Data ← [NSFile.nullReference];

      IF fs.needsDataHandle OR attributes.type = StarFileTypes.reference OR attributes.type = newReferenceIcon THEN
        data ← [NSFile.MakeReference[fileID: attributes.fileID,
          service: fs.parentReference.service]];

       SELECT ContainerCache.StatusOfFill[fs.cache] FROM
        inProgress, yes => NULL;
        inProgressPendingAbort, inProgressPendingJoin => RETURN[TRUE];
        ENDCASE => ERROR;

      addData ← BuildRow [fs, writers, readers, @itemData, @data, attributes];
      Exit[fs, 1300];
      [] ← ContainerCache.AppendItem [fs.cache, addData];
      Enter[fs,, 1300];
      --update the number of items in the return guy
     -- itemArray[0].nItems ← itemArray[0].nItems + 1;
      IF fs.nonNullScope THEN
        fs.length ← ContainerCacheExtra.GetLength[fs.cache];
      RETURN;

   END;

   BEGIN ENABLE {
      ABORTED => GOTO Aborted;
      NSFile.Error => {
        IF error = [handle[obsolete]] THEN
          IF RevalidateParent[fs, retryCount] THEN {
            retryCount ← retryCount+1;
            RETRY};
        ServicesErrorMessage.MsgFromNSFileError[error, postCacheMessage];
        fs.length ← ContainerCacheExtra.GetLength[fs.cache]; -- 16433
        CONTINUE};
      Courier.Error => {
        ServicesErrorMessage.MsgFromCourierError[errorCode, postCacheMessage];
        fs.length ← ContainerCacheExtra.GetLength[fs.cache];
        CONTINUE};
      }; -- ENABLE
```

```
    IF handle # NSFile.nullHandle THEN {
         NSFile.GetAttributes[handle, posSel, @attr, fs.session];
         -- 10484 move catch scope to include this statement

         --change the scope so that the enumeration starts at previous last one
         IF scope.filter = NSFile.nullFilter THEN
            scope.filter ← [var: greater [attribute: [position[attr.position]]] ]
         ELSE {array: ARRAY [0..2) OF NSFile.Filter ← [ [var: greater [attribute: [position[attr.position]]] ], scope.filter];
          scope.filter ← [and [DESCRIPTOR[array]]];--need to take the previous filter and add this guy to it
          }
         };


    NSFile.List [directory: fs.parentHandle, proc: Enumerator,
     selections: fs.selections, scope: scope, session: fs.session];

    IF ~fs.nonNullScope THEN fs.length ← CARDINAL.LAST;

    EXITS
      Aborted => {
         errorMsg ← XMessage.Get[BWSMessages.GetMessageHandle[],
            BWSMessages.kaborted];
         AppendMessageToCache [fs, fs.cache, @errorMsg]};
    END; -- enable

    IF handle # NSFile.nullHandle THEN NSFile.Close[handle, fs.session
       ! NSFile.Error, Courier.Error => CONTINUE];
    z.FREE [@readers];
    FreeWriters [writers];
    Exit[fs, 1300];
    END; -- enable
    RETURN[ --[insert[DESCRIPTOR[itemArray]]]-- [all[]] ]; --item is the former last guy
   END;


   FreeColumns: PROCEDURE [columns: ColumnContentsSeqHandle] = {
     FOR i: CARDINAL IN [0..columns.length) DO
       WITH newColumns: columns[i] SELECT FROM
         multipleAttributes => FreeExtendedSelections[newColumns.attrs.extended];
         ENDCASE;
       ENDLOOP;
     z.FREE[@columns];
     columns ← NIL};

   FreeExtendedSelections: PROCEDURE [
     extSelections: NSFile.ExtendedSelections] = {
     IF BASE[extSelections] # NIL
       THEN Heap.FreeNode[z:z, p: BASE[extSelections]]};

   FreeSelections: PROCEDURE [selections: NSFile.Selections] = {
     FreeExtendedSelections[selections.extended]};

   GetItemDataInternal: PUBLIC PROCEDURE [cache: ContainerCache.Handle, item: CARDINAL]
     RETURNS [ItemFileDataHandle] =
     -- INTERNAL: must be locked to call this proc
     BEGIN
     ih: ContainerCache.ItemHandle ← ContainerCache.GetNthItem [cache, item];
     IF ih = NIL THEN RETURN [NIL];
     RETURN [ ContainerCache.ItemClients[ih] ];
     END;

   MakeExtendedSelections: PROCEDURE [n: CARDINAL]
     RETURNS [NSFile.ExtendedSelections] = {
     p: LONG POINTER TO ARRAY OF NSFile.ExtendedAttributeType;
     p ← Heap.MakeNode[z:z, n: SIZE[NSFile.ExtendedAttributeType] * n ];
     FOR i: CARDINAL IN [0..n) DO p[i] ← 0; ENDLOOP;
     RETURN[DESCRIPTOR[p, n]]};

   MakeSelections: PROCEDURE [columns: ColumnContents]
     RETURNS [selections: NSFile.Selections] =
     BEGIN
     -- This proc builds an NSFile.Selections from the client's Columns,
     --  to be used when doing the NSFile.List.

     countExtAttr, extAttrIndex: CARDINAL ← 0;

     -- Count the client's extended attributes.

     FOR i: CARDINAL IN [0..LENGTH[columns]) DO
       WITH column: columns[i] SELECT FROM
         extendedAttribute => countExtAttr ← countExtAttr + 1;
         multipleAttributes =>
           countExtAttr ← countExtAttr + LENGTH[column.attrs.extended];
         ENDCASE;
       ENDLOOP;
     IF countExtAttr > 0 THEN
       selections.extended ← MakeExtendedSelections[countExtAttr]
     ELSE selections.extended ← NSFile.noExtendedSelections;

     -- Set the client's selections

     FOR i: CARDINAL IN [0..LENGTH[columns]) DO
       WITH column: columns[i] SELECT FROM
         attribute =>
           selections.interpreted[column.attr] ← TRUE;
         extendedAttribute => {
           selections.extended[extAttrIndex] ← column.extendedAttr;
```

```
          extAttrIndex ← extAttrIndex + 1};
        multipleAttributes =>
          BEGIN
          found: BOOLEAN;
          -- OR in the interpreted attributes
          FOR ati: NSFile.AttributeType IN NSFile.AttributeType DO
            selections.interpreted[ati] ←
              column.attrs.interpreted[ati] OR selections.interpreted[ati];
            ENDLOOP;
          -- "OR" in the extended attributes
          FOR j: CARDINAL IN [0..LENGTH[column.attrs.extended]) DO
            -- If the attribute is already in selections, ignore it
            found ← FALSE;
            FOR k: CARDINAL IN [0..LENGTH[selections.extended]) DO
              IF selections.extended[k] = column.attrs.extended[j] THEN
                 {found ← TRUE; EXIT};
              ENDLOOP;
            IF ~found THEN {
              selections.extended[extAttrIndex] ← column.attrs.extended[j];
              extAttrIndex ← extAttrIndex + 1};
            ENDLOOP;
          END;
        ENDCASE;
      ENDLOOP;

    -- Set my (the FileContainerSource) selections

    selections.interpreted[fileID] ← TRUE;
    selections.interpreted[position] ← TRUE;
    selections.interpreted[type] ← TRUE;
     -- cut back base in case we had duplicated attributes
    selections.extended ← DESCRIPTOR [BASE[selections.extended], extAttrIndex];
    RETURN[selections];
    END; -- MakeSelections

  RemoveSource: PROC [fs: FS] = {
    Enter[@allSourcesEnumLock];  -- 14807 lock two locks
    Enter[@allSourcesAddLock];
    BEGIN
    last: FS ← NIL;
    cur: FS ← allSources;
    WHILE cur # NIL AND cur # fs DO last ← cur; cur ← cur.nextSource; ENDLOOP;
    IF cur # NIL THEN {
      IF last # NIL THEN last.nextSource ← cur.nextSource
      ELSE allSources ← cur.nextSource};
    END;
    Exit[@allSourcesAddLock];
    Exit[@allSourcesEnumLock]
    }; -- RemoveSource

  RevalidateParent: PUBLIC PROCEDURE [fs: FS, retryCount: CARDINAL]
    RETURNS [ok: BOOLEAN] = {
    IF retryCount > 0 THEN RETURN [FALSE];
    -- Enter[fs];
    {ENABLE {NSFile.Error, Courier.Error => CONTINUE; };
      fs.parentHandle ← NSFile.OpenByReference[fs.parentReference, [timeout: 10],fs.session];
      -- Exit[fs];
      RETURN[TRUE]};
     -- continue after error
    -- Exit[fs];
    RETURN[FALSE]};

  ValidFileSource: PUBLIC PROCEDURE [source: ContainerSource.Handle]
    RETURNS [fs: FS] =
    BEGIN
    IF source = NIL THEN ERROR ContainerSource.Error [invalidParameters] ;
    fs ← LOOPHOLE[source];
    IF fs.procs # fileSourceProcs THEN
      ERROR ContainerSource.Error[invalidParameters];
    END;

  InitAtoms: PROCEDURE = {
    open ← Atom.MakeAtom["Open"L];
    props ← Atom.MakeAtom["Props"L];
    takeSelectionBackground ← Atom.MakeAtom["TakeSelectionBackground"L];
    takeSelectionCopyBackground ← Atom.MakeAtom["TakeSelectionCopyBackground"L];
    freeMenu ← Atom.MakeAtom["FreeMenu"L];
    canYouTakeSelection ← Atom.MakeAtom["CanYouTakeSelection"L];
    takeSelection ← Atom.MakeAtom["TakeSelection"L];
    takeSelectionCopy ← Atom.MakeAtom["TakeSelectionCopy"L]};


  InitAtoms[];

  END.

 5-Sep-84 - Holbrook - Fix AppendMessageToCache to append null strings for extra columns
26-Sep-84 - Holbrook - Fixes for ARs 11349, 11354
15-Oct-84 - Holbrook - In CopyMoveFile, get new reference for moved file
29-Oct-84 - Holbrook - Temp till defs change: call SWSExtra.SetContainee
31-Oct-84 - Holbrook - Changes to FixupPosition to support insert items in correct cache position after item inserted into a sorted
folder.  Also support relist in ActOn proc  (AR 10476)
16-Nov-84 - Holbrook - Update to 4.0a defs
 7-Dec-84 - Holbrook - Use XTime.dateAndTime for date format, AR 12468, 12466, 12463, 12460, 12461, 12460, 12459
11-Dec-84 - Holbrook - Add ChangeScope proc
12-Dec-84 - Holbrook - add needsDataHandle
13-Dec-84 - Holbrook - add stuff to move/copy in same source
```

16-Jan-85 - Holbrook - Put SWS.SetContainee back in
22-Jan-85 - Holbrook - Error handling
15-Mar-85 - Holbrook - AR 13415
28-Mar-85 - Holbrook - allocate source proc object from heap
11-Apr-85 - Holbrook - Zap strings in favor of messages
 3-May-85 - Holbrook - 14812: change DateFormatProc to show date for application folder
17-May-85 - Holbrook - Comment in BuildRow
30-May-85 - Holbrook - AR 15325; show date for all non-folders (15310), clean up a little
18-Jun-85 - SAJohnson.ES - AR 15971: Initialize array in MakeExtendedSelections
20-Jun-85 - SAJohnson.ES - AR 14907: Workaround for Filing problem to do with aborting processes.
21-Jun-85 - SAJohnson.ES - AR 15309: If it's a folder show the number of children; all others show the size in pages.
 1-Jul-85 - SAJohnson.ES - AR 16352: Implement NoticeAppendedFiles.
 8-Jul-85 - Holbrook - Cache parent handle
 2-Aug-86 - Holbrook - 17240 (catch Courier.Error in NSFile.Close)
22-Aug-85 - Holbrook - 18726: reset length in NoticeAppendedFiles
 3-Sep-85 - Holbrook - 19529: Added SpecialContainer2.WaitSourceIdle
25-Sep-85 - Breisacher - Use subtreeSize instead of sizeInPages for Size column.
19-Jun-86 - Holbrook - 4.3 Background stuff: ContainerSourceExtra procedures
25-Jul-86 - Holbrook - session stuff
 7-Aug-86 - Holbrook - monitoring stuff
30-dec-86 - Holbrook - don't lock in AppendMessageToCache: locks up with StringOfFileItem.
 2-Feb-87 - Holbrook - 10484 fix NoticeAppendedFiles to put catch phrase around GetAttributes call
26-May-87 - LFB - AR 12269 - check for new file type of Reference Icons.
16-Jun-87 - jph - 12615 don't do SetContainee if it's already set
17-Jul-87 - jph - 13248 add SpecialContainer3.AboutToDestroySource to avoid race condition; 13215 monitor source enumerate stuff
28-Aug-87 - jph - 14024 Creating one session per open container window is too expensive. First thing we try is simplest approach: one common session for all container windows. It's created the first time someone asks for it, and it goes away at logoff. New procs: GetSession, DestroySession
14-Sep-87 - jph - 13756 Relist forces refetch of source length; 14431 remove all session stuff from here and call SessionCache
22-Sep-87 - jph - 14545 in DeleteFileItems, don't use nextItem as bound for deleted items, because it may have been moved in the background
 7-Oct-87 - jph - 14807 Use more than one lock in EnumerateSources/AddSource to avoid deadlock
12-Nov-87 - jph - 15592 ActOnFile does RemoveSource before doing an Enter to avoid deadlock.
 7-Dec-87 - LFB - 15843 15845. Hardwire books and mail folders in the DateFormatProc.
27-Jan-88 - jph - 16433 On error/abort of listing proc, set fs.length to actual number of items in cache.

```
-- File: FileContainerSourceImplB.mesa - last edit:
-- Holbrook            24-Nov-87 15:13:43
-- Breisacher          17-Nov-87 14:33:07
-- Holbrook.ES         14-Oct-86  9:50:29
-- Riggle.PA           23-May-86 13:58:09
-- Mita.ES             17-Mar-86 11:04:25
-- SAJohnson.ES        19-Jun-85 15:02:44

-- Copyright (C) 1984, 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Attention USING [Post],
  BackgroundProcess USING [UserAbort],
  BackgroundProcessExtra USING [GetName, SetName],
  BWSAttributeTypes USING [remoteName],
  Containee USING [ChangeProc, Data, DataHandle, GetImplementation, Implementation, InvalidateCache, SmallPictureProc],
  ContaineeExtra USING [GetCachedBusy, SetCachedBusy],
  ContainerCache USING [AddData, AppendItem,
    DeleteNItems, FreeMark, Handle,
    IndexFromMark, InsertItem, ItemHandle,
    Mark, MarkObject, MoveMark, Object, ReplaceItem, SetMark],
  ContainerSource USING [beforeItemZero, CanYouTakeProc, ChangeInfo,
    ChangeProc, ConvertItemProc, EditInfo, Error,
    ErrorCode, Handle, ItemIndex, nullItem,
    TakeProc],
  ContainerSourceExtra,
  Courier USING [Error],
  DtCtnrMessages,
  Inline USING [LongCOPY],
  FileContainerSourceOps,
  FileContainerSource,
  FileContainerSourceExtra2,
  FileContainerSourceExtra4,
  NSFile USING [Attribute, AttributeList, Attributes, AttributesProc, AttributesRecord, AttributeType, ChangeAttributes, ClearAttributes,
  ChangeControls, Close, Controls, Copy, defaultTimeout, EncodeString, Error, ErrorRecord, ExtendedAttributeType, firstPosition,
  FreeWords, GetAttributes, GetControls, GetReference, Handle, ID, List, MakeReference, Move, nullAttributeList, nullHandle,
  nullOrdering, nullReference, nullSession, OpenChild, OpenByReference, Ordering, Reference, Selections, Session, String, Type, Words],
  NSFileName USING [AppendVPNToString, VPNRecord],
  NSName USING [maxFullNameLength],
  NSString USING [FreeString, MakeString, nullString, String,
    StringBoundsFault],
  Process USING [GetCurrent],
  Selection USING [CanYouConvert, ConversionInfo, CopyMove, Difficulty, EnumerationProc, Error, Free, NopFree, nullValue, QueryElement,
  Target, Value, ValueCopyMoveProc, ValueFreeProc, ValueProcs],
  SelectionX,
  SessionCache USING [GetSession, ReturnSession],
  SpecialContainerCache USING [WaitCacheIdle],
  TIP USING [UserAbort],
  TIPXX USING [GetNotifierProcess],
  XChar USING [Make],
  XCharSet0,
  XCharSets,
  XMessage USING [ComposeOne, Get, MsgKey],
  XString;

FileContainerSourceImplB: PROGRAM
  IMPORTS Attention, BackgroundProcess, BackgroundProcessExtra, Containee, ContaineeExtra, ContainerCache, ContainerSource, Courier,
  DtCtnrMessages, FileContainerSourceOps, Inline, NSFile, NSFileName, NSString, Process, Selection, SelectionX, SessionCache,
  SpecialContainerCache, TIP, TIPXX, XChar, XMessage, XString
  EXPORTS FileContainerSourceOps, FileContainerSourceExtra2, FileContainerSourceExtra4 =
  BEGIN OPEN FileContainerSourceOps, FileContainerSource;

  -- TYPEs

  MarksSeq: TYPE = RECORD [SEQUENCE length: CARDINAL OF ContainerCache.Mark];
  MarksSeqPtr: TYPE = LONG POINTER TO MarksSeq;
  InsertInfoDesc: TYPE = LONG DESCRIPTOR FOR ARRAY OF ContainerSource.EditInfo;
  InsertInfoSequence: TYPE = RECORD[SEQUENCE COMPUTED CARDINAL OF ContainerSource.EditInfo];

  Mark: TYPE = ContainerCache.Mark;

  -- Data

  fileTimeout: CARDINAL ← 8; -- make it different than desktop's 10
  ValueProcHandle: TYPE = LONG POINTER TO Selection.ValueProcs;
  fileValueProcs: ValueProcHandle ← z.NEW[Selection.ValueProcs ← [FreeFile, CopyMoveFile]]; -- used by ConvertFileItem for converting
  files
  filetypeValueProcs:  ValueProcHandle ← z.NEW[Selection.ValueProcs ← [Selection.NopFree, NIL]]; -- used by ConvertFileItem for type
  fileType

  remoteNameAllocate: CARDINAL = NSName.maxFullNameLength + 100;
    -- 100 extra chars for path.


  AnySelectionsInCommon: PROCEDURE [sel1, sel2: NSFile.Selections]
    RETURNS [yes: BOOLEAN] =
    BEGIN
    FOR i: NSFile.AttributeType IN NSFile.AttributeType DO
      IF sel1.interpreted[i] AND sel2.interpreted[i] THEN RETURN[TRUE];
      ENDLOOP;
    FOR j: CARDINAL IN [0..LENGTH[sel1.extended]) DO
      FOR k: CARDINAL IN [0..LENGTH[sel2.extended]) DO
        IF sel1.extended[j] = sel2.extended[k] THEN RETURN[TRUE];
        ENDLOOP;
      ENDLOOP;
    RETURN[FALSE];
    END;
```

```
CanITake: PUBLIC ContainerSource.CanYouTakeProc =
  BEGIN
  RETURN[CanITakeX[source, FALSE]];
  END;

CanITakeX: PUBLIC ContainerSourceExtra.CanYouTakeProcX  =
  BEGIN
  fs: FS = ValidFileSource[source];
    -- make call just to verify we have legal FCS
  IF background THEN RETURN[TRUE];
  RETURN[Selection.CanYouConvert[target: file, enumeration: FALSE] OR
      Selection.CanYouConvert[target: file, enumeration: TRUE] ];
  END;

ConvertItemContextObject: TYPE = MACHINE DEPENDENT RECORD [
   itemRef (0:0..111): NSFile.Reference ← NSFile.nullReference,
   -- itemRef must be first; client expect it to be a reference
   destHandle (7:0..31): NSFile.Handle ← NSFile.nullHandle,
   session (9:0..31): NSFile.Session ← NSFile.nullSession,
   destRef (11:0..111): NSFile.Reference ← NSFile.nullReference,
   parentHandle (18:0..31): NSFile.Handle ← NSFile.nullHandle,
   freeContext (20:0..0): BOOLEAN,
   takeInSameSource (20:1..15): BOOLEAN,
   remoteNameString (21:0..63): NSString.String ← NSString.nullString];
ConvertItemContext: TYPE = LONG POINTER TO ConvertItemContextObject;

ConvertFileItem: PUBLIC ContainerSource.ConvertItemProc =
  <<[source: ContainerSource.Handle,
    itemIndex: ContainerSource.ItemIndex,
    n: CARDINAL ← 1,
    target: Selection.Target,
    zone: UNCOUNTED ZONE,
    info: Selection.ConversionInfo ← [0, 0, 0, 0],
    changeProc: ContainerSource.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL] RETURNS [value: Selection.Value]>>

  BEGIN
  fs: FS = ValidFileSource[source];
  itemData: ItemFileDataHandle;
  preFetchUsed: BOOLEAN ← FALSE;
  deleteItems: DeleteItemsHandle ← NIL;
  ci: Containee.Implementation ← TRASH;  --for a single item this is fine; must change it if it's an enumeration (AR 15692)
  changeInfo: ContainerSource.ChangeInfo;
  itemMarks: MarksSeqPtr ← NIL;
  itemContext: ConvertItemContext ← NIL;
  inLock: BOOLEAN; -- if we UNWIND, are we in the lock and need to get out?
  takeInSameSource: BOOLEAN ← fs.foregroundTakeProcess = Process.GetCurrent[];

   GetDifficulty: PROCEDURE [itemData: ItemFileDataHandle, target: Selection.Target, checkEnumeration: BOOLEAN ← FALSE]
    RETURNS [difficulty: Selection.Difficulty ← impossible] =
    BEGIN
    ci: Containee.Implementation = Containee.GetImplementation [itemData.type];
    qe: ARRAY[0..1) OF Selection.QueryElement ← [[target, checkEnumeration, impossible]];
    cd: Containee.Data ← [reference:
      NSFile.MakeReference [itemData.id, fs.parentReference.service]];

      IF ci.convertProc = NIL THEN RETURN [impossible];
      value ← ci.convertProc [@cd, target, zone, [query[DESCRIPTOR [qe]]]];
      RETURN [qe[0].difficulty]
    END;

  ConvertOneItem: PROCEDURE
    [oneItem: ContainerSource.ItemIndex, info: enumeration Selection.ConversionInfo]
    RETURNS [abort: BOOLEAN ← FALSE] =
    BEGIN
    busy: BOOLEAN ← IsBusyInternal[fs, oneItem];
    -- we enter locked
    BEGIN ENABLE UNWIND => IF inLock THEN Exit[fs, 1400];
    IF preFetchUsed THEN -- first one prefetched
      itemData ← GetItemDataInternal [fs.cache, oneItem]
    ELSE preFetchUsed ← TRUE; -- can't use it again
    -- what do we do if it's null?
    SELECT target FROM
      file, SelectionX.fileWithFeedback => {
        myChangeProcData: LONG POINTER TO ChangeProcData ← z.NEW [
        ChangeProcData ← [
          fs: fs,
          item: ContainerCache.SetMark[fs.cache, oneItem],
          deleteItems: IF ~busy THEN @deleteItems ELSE NIL,
          busy: busy,
          giveFeedback: target = SelectionX.fileWithFeedback,
          callersChangeProc: changeProc,
          callersChangeProcData: changeProcData ] ];
        << Construct a NSFile.Reference using the systemElement and
           volumeID of the parent (directory) file. >>
        inLock ← FALSE;
        Exit[fs, 1400];  -- unlock before calling back to client
        BEGIN

        itemContext.itemRef ← NSFile.MakeReference [
            fileID: itemData.id,
            service: fs.parentReference.service];
        IF info.proc[[value: itemContext, ops: fileValueProcs, context: myChangeProcData]]
          THEN RETURN[TRUE];
        END; -- ENABLE
        }; -- file
```

```
      fileType => {
        inLock ← FALSE; Exit[fs, 1400];
        IF info.proc[[value: @itemData.type,
           ops: filetypeValueProcs]]
         THEN RETURN[TRUE]};
    ENDCASE =>
      -- Let the underlying Containee do the convert
      BEGIN
      localAbort: BOOLEAN ← FALSE;
      cd: Containee.Data ← [ reference:
        NSFile.MakeReference [
          fileID: itemData.id,
          service: fs.parentReference.service] ];
      value: Selection.Value;

      PassThru: PROC [v: Selection.Value] -- 15722 allow abort
        RETURNS [stop: BOOLEAN] = {
        stop ← localAbort ← info.proc[v]
        };

      inLock ← FALSE; Exit[fs, 1400];
      ci ← Containee.GetImplementation[itemData.type];
      value ← ci.convertProc[data: @cd, target: target, zone: z, info: [convert []]];
      IF value = Selection.nullValue THEN -- try an enumeration
        {value ← ci.convertProc[data: @cd, target: target, zone: z, info: [enumeration[PassThru]]];  -- 15722
        RETURN [localAbort]};
      RETURN[info.proc[value]];
      END;
    IF inLock THEN {inLock ← FALSE; Exit[fs, 1400]};
    END; -- enable
    END; -- ConvertOneItem

  Enter[fs, lockIfNotBetweenCalls, 1400]; inLock ← TRUE;
  BEGIN ENABLE UNWIND => IF inLock THEN Exit[fs, 1400];
  itemData ← GetItemDataInternal [fs.cache, itemIndex];
  IF itemData = NIL THEN {Exit[fs, 1400]; RETURN[Selection.nullValue]};
  ci ← Containee.GetImplementation[itemData.type];
  WITH i:info SELECT FROM
    query => {
      inLock ← FALSE; Exit[fs, 1400];
      FOR c: CARDINAL IN [0..i.query.LENGTH) DO
        i.query[c].difficulty ←
          IF i.query[c].enumeration THEN
             SELECT i.query[c].target FROM
              fileType => impossible,
              file, SelectionX.fileWithFeedback => impossible,
              ENDCASE => GetDifficulty [itemData, i.query[c].target, TRUE]
          ELSE -- not an enumeration
            SELECT i.query[c].target FROM
              fileType => easy,
              file, SelectionX.fileWithFeedback => easy,
              ENDCASE => GetDifficulty [itemData, i.query[c].target];
        ENDLOOP};
    enumeration => {
      thisItem: ContainerSource.ItemIndex;
      thisMark, nextMark: ContainerCache.Mark;
      index: CARDINAL;
      abort: BOOLEAN ← FALSE;
      unbusyItems: BOOLEAN ←  (target=fileType) OR (target=file) OR (target=SelectionX.fileWithFeedback);

      EnumerateItems: PROCEDURE [count: CARDINAL, convert: BOOLEAN] =
        BEGIN
        firstTime: BOOLEAN ← TRUE;

        FOR index IN [0..count) DO
          -- MUST be locked here
          -- if convert is false, we are unwinding and unbusying items without
          -- converting them.  We start with thisMark because we need to unbusy
          -- them item that just failed to convert
          thisItem ← IF ~takeInSameSource
            THEN (IF convert OR ~firstTime THEN ContainerCache.IndexFromMark[nextMark]
                 ELSE ContainerCache.IndexFromMark[thisMark])
            ELSE ContainerCache.IndexFromMark[itemMarks[index]];
          firstTime ← FALSE;
          ContainerCache.MoveMark[thisMark, thisItem];
          ContainerCache.MoveMark[nextMark, thisItem+1];
          -- Convert unlocks to call client: if abort, we don't unlock
          IF convert AND ~abort THEN {
            abort ← ConvertOneItem[thisItem, i];
            Enter[fs,, 1400]; inLock ← TRUE};
          thisItem ← ContainerCache.IndexFromMark[thisMark];
          IF unbusyItems AND thisItem # ContainerSource.nullItem
            AND IsBusyInternal[fs, thisItem] THEN
              [] ← SetBusyInternal[fs, thisItem, FALSE, changeProc, changeProcData];
        ENDLOOP;
        END; -- EnumerateItems

      FinishEnumeration: PROC =
        BEGIN
        IF inLock THEN {inLock ← FALSE; Exit[fs, 1400]};
        IF takeInSameSource THEN {
          FOR i: CARDINAL IN [0..n) DO
             ContainerCache.FreeMark[itemMarks[i]];
             ENDLOOP;
          z.FREE[@itemMarks]};

        IF itemContext.remoteNameString # NSString.nullString THEN {
```

```
            NSString.FreeString[z, itemContext.remoteNameString];
            itemContext.remoteNameString ← NSString.nullString};
          -- free enumeration context and close destination
          IF itemContext.destHandle # NSFile.nullHandle THEN {
            NSFile.Close[itemContext.destHandle, itemContext.session
               ! NSFile.Error, Courier.Error => CONTINUE];
            NSFile.Close[itemContext.parentHandle, itemContext.session
               ! NSFile.Error, Courier.Error => CONTINUE];
            SessionCache.ReturnSession[itemContext.session ! NSFile.Error, Courier.Error => CONTINUE]};

          z.FREE[@itemContext];

          IF deleteItems # NIL THEN {
            allChanged: BOOLEAN ← (deleteItems.next # NIL);
            << If there is one contiguous group of deleted items, we'll call the
              change proc with the delete variant.  Else we call with all variant. >>
            Enter[fs,, 1400]; inLock ← TRUE;
            BEGIN ENABLE UNWIND => Exit[fs, 1400];
            WHILE deleteItems # NIL DO
              temp: DeleteItemsHandle ← deleteItems;
              first: ContainerSource.ItemIndex ← ContainerCache.IndexFromMark[temp.firstItem];
              fs.length ← fs.length - temp.count;
              IF ~allChanged THEN
                changeInfo ← [var:
                 delete[deleteInfo: [afterItem: first, nItems: temp.count]]
                   ];
              ContainerCache.DeleteNItems[fs.cache, first, temp.count];
              ContainerCache.FreeMark[temp.firstItem];
              deleteItems ← temp.next;
              z.FREE[@temp];
              ENDLOOP;
            IF allChanged THEN changeInfo ← [var: all[]];
            IF changeProc # NIL THEN
              changeProc[changeProcData, changeInfo];
            Exit[fs, 1400]; inLock ← FALSE;
            END; -- enable
            }; -- deleteItems # NIL
          END; -- FinishEnumeration

        -- enumeration =>;  in lock at this point

              IF takeInSameSource THEN  -- copy/move within same source
          {
          itemMarks ← z.NEW[MarksSeq[n]];
          FOR i: CARDINAL IN [0..n) DO
            itemMarks[i] ← ContainerCache.SetMark[fs.cache, i+itemIndex];
            ENDLOOP;
          };

        BEGIN ENABLE
          UNWIND => {
            ContainerCache.FreeMark[thisMark];
            ContainerCache.FreeMark[nextMark];
            FinishEnumeration[]};

        itemContext ← z.NEW[ConvertItemContextObject ← [freeContext: FALSE, takeInSameSource: takeInSameSource]];
        IF fs.isRemote THEN
          itemContext.remoteNameString ← NSString.MakeString[z, remoteNameAllocate];

        nextMark ← ContainerCache.SetMark[fs.cache, itemIndex];
        thisMark ← ContainerCache.SetMark[fs.cache, itemIndex];
        EnumerateItems[n, TRUE !  -- unbusy old items
          UNWIND => {
            IF ~inLock THEN {Enter[fs,, 1400]; inLock ← TRUE};
            EnumerateItems[n-index, FALSE]}];
        ContainerCache.FreeMark[thisMark];
        ContainerCache.FreeMark[nextMark];
        END; -- ENABLE
        -- FinishEnumeration will get out of lock if we are in it
        FinishEnumeration[]; -- close destination and free itemContext
        };  -- enumeration

      convert => SELECT target FROM
        file, SelectionX.fileWithFeedback => {
          -- I can convert files
          myChangeProcData: LONG POINTER TO ChangeProcData ← z.NEW [
            ChangeProcData ← [
              fs: fs,
              item: ContainerCache.SetMark[fs.cache, itemIndex],
              busy: IsBusyInternal[fs, itemIndex],
              giveFeedback: target=SelectionX.fileWithFeedback,
              deleteItems: NIL,
              callersChangeProc: changeProc,
              callersChangeProcData: changeProcData ] ];

            << Construct a NSFile.Reference using the systemElement and volumeID
               of the parent (directory) file. >>
          Exit[fs, 1400]; inLock ← FALSE;
          itemContext ← z.NEW[ConvertItemContextObject ←
                              [itemRef: NSFile.MakeReference [
                                        fileID: itemData.id,
                                        service: fs.parentReference.service],
                               freeContext: TRUE,
                               takeInSameSource: takeInSameSource]];

          RETURN[[value: itemContext, ops: fileValueProcs, context: myChangeProcData]] };
        fileType => {Exit[fs, 1400];
```

```
                    RETURN[[value: @itemData.type, ops: filetypeValueProcs]]};
      ENDCASE =>
        -- Let the underlying Containee do the convert
        BEGIN
        cd: Containee.Data ← [ reference:
          NSFile.MakeReference [
            fileID: itemData.id,
            service: fs.parentReference.service] ];
        Exit[fs, 1400];
        RETURN[ci.convertProc[data: @cd, target: target, info: info, zone: z]];
        END;
    ENDCASE; -- WITH
  IF inLock THEN Exit[fs, 1400];
  RETURN[Selection.nullValue]; -- just in case we get here
  END; -- enable
  END; -- ConvertFileItem

FileTake: PUBLIC ContainerSource.TakeProc = {
  RETURN[FileTakeX[source, copyOrMove, afterHint, withinSameSource, changeProc, changeProcData, SelectionX.nullManager]]};

FileTakeX: PUBLIC ContainerSourceExtra.TakeProcX =
<< source: Handle,
   copyOrMove: Selection.CopyOrMove,
   afterHint: ItemIndex ← nullItem,
   withinSameSource: BOOLEAN ← FALSE,
   changeProc: ChangeProc ← NIL,
   changeProcData: LONG POINTER ← NIL,
   mgr: SelectionX.Saved ← SelectionX.nullManager]
   RETURNS [ok: BOOLEAN] >>
  BEGIN OPEN ContainerSource;

  fs: FS = ValidFileSource[source];
  v: Selection.Value;
  writers: WriterSeqPtr ← AllocateWriters [fs.columns.length];
  readers: ReaderSeqPtr ← z.NEW [ReaderSeq[fs.columns.length]];
  nextItemMark: ContainerCache.Mark;
  firstItemMark: ContainerCache.Mark ← NIL;
  nextIsBeforeItemZero: BOOLEAN ← afterHint = beforeItemZero;
  anyChanges, allChanged: BOOLEAN ← FALSE;
  background: BOOLEAN ← mgr # SelectionX.nullManager;
  newPosition: ItemIndex ← nullItem;
  insertInfo: RECORD [
    length: CARDINAL ← 0,
    info: InsertInfoDesc ← NewInsertInfo[insertInfo.info, 10, FALSE]];
  inLock: BOOLEAN;

  TakeFile: Selection.EnumerationProc =
  <<[element: Selection.Value, data: Selection.RequestorData] RETURNS [stop: BOOLEAN ← FALSE]>>
    BEGIN ENABLE
      UNWIND => Selection.Free[@element];

    f: LONG POINTER TO NSFile.Reference;
    after: ItemIndex;
    lastItemPos: ItemIndex;

    IF element.value = NIL THEN GOTO GetOut;

    << 20181: don't start move or copy if destination ctnr source is still
       enumerating and we are moving into a sorted source.  Reason: we
       can end up having something inserted into the cache twice. >>
    IF (IF background THEN BackgroundProcess.UserAbort[] ELSE TIP.UserAbort[NIL])
        THEN {
      PostMsg[IF copyOrMove=copy THEN DtCtnrMessages.kcopyStopped ELSE DtCtnrMessages.kmoveStopped];
      stop ← TRUE;
      GOTO GetOut};
    f ← element.value;
    IF fs.takeFilter # NIL AND ~fs.takeFilter [LOOPHOLE [fs], f↑] THEN GOTO GetOut; -- AR 13817.
    IF ~fs.ascendingOrDescending THEN
      SpecialContainerCache.WaitCacheIdle[fs.cache];
    Selection.CopyMove [v: @element, op: copyOrMove, data: @fs.parentReference
      ];
    f ← element.value;
    -- get in lock to update cache
    Enter[fs,, 1500]; inLock ← TRUE;
    after ← IF nextIsBeforeItemZero
        THEN beforeItemZero
        ELSE ContainerCache.IndexFromMark[nextItemMark];
    IF nextIsBeforeItemZero THEN nextIsBeforeItemZero ← FALSE;
    lastItemPos ← newPosition;
    -- if in background, add new items only at end of cache
    newPosition ← FixupPosition [fs: fs, file: f↑, session: fs.session,
      parentHandle: fs.parentHandle, afterHint: after, addAtEnd: background,
      readers: readers, writers: writers];

    fs.length ← fs.length+1;

    -- set up insertInfo stuff
    IF background THEN { -- call display proc right away
      insertInfo.info[0] ← ContainerSource.EditInfo[newPosition, 1];
      IF changeProc # NIL THEN
        changeProc[changeProcData, [insert[DESCRIPTOR [BASE[insertInfo.info], 1]]]]}
    ELSE { --
    << 10747: if items are inserted in more than one location in sorted ctnr, force entire display to be changed when display is
    updated >>
      IF lastItemPos = nullItem OR ((lastItemPos + 1) # newPosition) THEN {
        -- new contigious sequence of arrays
        -- grow array if necessary
```

```
      IF ~fs.ascendingOrDescending AND lastItemPos#nullItem THEN allChanged ← TRUE
      ELSE {
        IF insertInfo.info.LENGTH >= insertInfo.length THEN
          insertInfo.info ←
            NewInsertInfo[insertInfo.info, insertInfo.length+10, TRUE];

        insertInfo.info[insertInfo.length] ← ContainerSource.EditInfo[newPosition, 1];
        insertInfo.length ← insertInfo.length+1}}
    ELSE  -- right after the last one
      insertInfo.info[insertInfo.length-1].nItems ← insertInfo.info[insertInfo.length-1].nItems+1
  };
  -- always mark the first item for
  IF firstItemMark = NIL THEN firstItemMark ← ContainerCache.SetMark[fs.cache, newPosition];

  after ← IF after = beforeItemZero
            THEN 0 ELSE ContainerCache.IndexFromMark[nextItemMark]+1;
  ContainerCache.MoveMark[nextItemMark, after];
  Selection.Free[@element];
  Exit[fs, 1500]; inLock ← FALSE;
  anyChanges ← TRUE;

  EXITS
    GetOut => {Selection.Free [@element]; RETURN};

  END;

CleanUp: PROC =
  BEGIN
  IF ~inLock THEN Enter[fs,, 1500];
  IF ~background AND changeProc # NIL THEN -- already called for background items
    changeProc[changeProcData, IF allChanged THEN [all[]] ELSE IF anyChanges THEN [insert[DESCRIPTOR [BASE[insertInfo.info],
    insertInfo.length]]] ELSE [noChanges[]] ];
  Exit[fs, 1500];
  ContainerCache.FreeMark[nextItemMark];
  {lp: LONG POINTER ← BASE[insertInfo.info];
   z.FREE[@lp]};
  z.FREE [@readers];
  IF firstItemMark # NIL THEN ContainerCache.FreeMark[firstItemMark];
  FreeWriters [writers];
  IF ~background THEN fs.foregroundTakeProcess ← NIL;
  END;

Enter[fs, lockIfNotBetweenCalls, 1500]; inLock ← TRUE;
BEGIN ENABLE UNWIND => CleanUp[];
qe: ARRAY[0..3) OF Selection.QueryElement ← [[file, TRUE, impossible], [SelectionX.fileWithFeedback, TRUE, impossible], [file, FALSE,
impossible]];
feedbackOk: BOOLEAN;

nextItemMark ← ContainerCache.SetMark [cache: fs.cache, index: IF ((afterHint = beforeItemZero) OR (afterHint = nullItem)) THEN 0
ELSE afterHint];
IF ~background THEN fs.foregroundTakeProcess ← Process.GetCurrent[];
Exit[fs, 1500]; inLock ← FALSE;
IF background THEN {
  name: XString.ReaderBody ← IconAndNameFromFS[fs];
  SetNameForProcess[name];
  XString.FreeReaderBytes[@name, z]};


SelectionX.QueryX[DESCRIPTOR [qe], mgr];
feedbackOk ← background AND qe[1].difficulty # impossible;
insertInfo.info[0].nItems ← 0;

SELECT TRUE FROM
  qe[0].difficulty # impossible =>
  --- CanYouConvert [target: file, enumeration: TRUE] =>
    BEGIN
    [] ← SelectionX.EnumerateX [target: IF feedbackOk THEN SelectionX.fileWithFeedback ELSE file, proc: TakeFile, manager: mgr];
    Enter[fs,, 1500]; inLock ← TRUE;
    IF firstItemMark # NIL THEN {
      insertInfo.info[0].afterItem ←
        ContainerCache.IndexFromMark[firstItemMark];
    CleanUp[];
    RETURN [ok: TRUE]; };
    END;
  -- CanYouConvert [target: file, enumeration: FALSE] =>
  qe[2].difficulty # impossible =>
    BEGIN
    v ← SelectionX.ConvertX[target: file, manager: mgr];
    IF v.value = NIL THEN {
      RETURN [FALSE]}
    ELSE {
      ok ← ~TakeFile[v, NIL];
      IF NOT ok THEN { CleanUp[]; RETURN[ok]};
      Enter[fs,, 1500]; inLock ← TRUE;
      IF firstItemMark # NIL THEN {
        insertInfo.info[0].afterItem ←
          ContainerCache.IndexFromMark[firstItemMark];
        };
      background ← FALSE; -- force changeProc to get called
      CleanUp [];
      RETURN [ok]; };
    END;
  ENDCASE => {CleanUp []; RETURN [ok: FALSE]};
CleanUp [];
END; -- enable
END;  -- FileTake
```

```
  CopyMoveFile: Selection.ValueCopyMoveProc =
    << [v: Selection.ValueHandle, op: {copy, move}, data: LONG POINTER], where data is interpreted as an NSFile.Reference of the
    destination directory file.  A copy operation returns the new file's reference as v.value.>>
    BEGIN

<<SHOULD Put in NSFile catch phrases!!!>>
    changeData: LONG POINTER TO ChangeProcData ← v.context;
    fs: FS = changeData.fs;

    itemContext: ConvertItemContext ← v.value;
    destRef: LONG POINTER TO NSFile.Reference ← data;
    fileRef: NSFile.Reference ← itemContext.itemRef;
    retryCount: CARDINAL ← 0;
    dest: NSFile.Handle;
    file: NSFile.Handle;
    changeItem: ContainerSource.ItemIndex;
    attributes: NSFile.AttributesRecord ← TRASH;
    stringWasNil: BOOLEAN ← (itemContext.remoteNameString = NSString.nullString);
    attr: ARRAY [0..1) OF NSFile.Attribute;
    attrList: NSFile.AttributeList ← NSFile.nullAttributeList;
    stringWords: NSFile.Words;
    vpn: NSFileName.VPNRecord;
    doOperation: BOOLEAN ← TRUE;
    inLock: BOOLEAN ← FALSE;

    CloseFiles: PROC = {
      NSFile.Close [file, itemContext.session ! NSFile.Error, Courier.Error => CONTINUE];
      };

    BEGIN ENABLE
      UNWIND => CloseFiles [];

    myProcess: PROCESS ← NIL;
    background: BOOLEAN;

    GetReference: PROCEDURE [file: NSFile.Handle] RETURNS [ref: NSFile.Reference ← NSFile.nullReference] =
      BEGIN
      ENABLE
      -- 7975 return nullReference and only on insufficient rights
        NSFile.Error =>  WITH e: error SELECT FROM
          access => IF e.problem = accessRightsInsufficient
            OR e.problem = accessRightsIndeterminate THEN CONTINUE ELSE REJECT;
              -- don't let access problem stop us
          ENDCASE => REJECT;
      ref ← NSFile.GetReference [file, itemContext.session];
      retryCount ← 0;
      END; -- GetReference

    GetAbort: PROCEDURE RETURNS [BOOLEAN] =    -- 14183
      BEGIN
      IF myProcess = NIL THEN {
        myProcess ← Process.GetCurrent[];
        background ← myProcess # TIPXX.GetNotifierProcess[]};
      RETURN[IF background THEN BackgroundProcess.UserAbort[] ELSE TIP.UserAbort[NIL]];
      END;

    -- check for move in same source: don't bother doing it
    IF itemContext.takeInSameSource AND op = move THEN doOperation ← FALSE;
      -- try caching destination handle
    IF destRef↑ = itemContext.destRef
      THEN dest ← itemContext.destHandle
      ELSE {IF itemContext.destHandle # NSFile.nullHandle
            THEN NSFile.Close[itemContext.destHandle, itemContext.session !
                NSFile.Error, Courier.Error => CONTINUE];
          itemContext.session ← SessionCache.GetSession[fs.parentReference];
            -- get a session from the source; is this right?
          dest ← NSFile.OpenByReference[destRef↑, [timeout: fileTimeout], itemContext.session !
        NSFile.Error => IF error = [access[fileInUse]] AND ~GetAbort[] AND retryCount < 6 THEN {
            retryCount ← retryCount+1;
            RETRY}];
          retryCount ← 0;
          itemContext.parentHandle ← NSFile.OpenByReference[fs.parentReference,
            [timeout: fileTimeout], itemContext.session !
        NSFile.Error => IF error = [access[fileInUse]] AND ~GetAbort[] AND retryCount < 6 THEN {
            retryCount ← retryCount+1;
            RETRY}];
          retryCount ← 0;
          itemContext.destRef ← destRef↑;  itemContext.destHandle ← dest};
    << Hanzel 10093: get filing to help resolve access conflicts by applying locks. >>
    file ←
      NSFile.OpenChild [directory: itemContext.parentHandle,
        controls: [timeout: fileTimeout,
        lock: SELECT op FROM
            copy => share,
            move => exclusive,
            ENDCASE => none],
        id: itemContext.itemRef.fileID, session: itemContext.session !
          NSFile.Error => IF error = [access[fileInUse]] AND ~GetAbort[] AND retryCount < 6 THEN {
            retryCount ← retryCount+1;
            RETRY}];
    retryCount ← 0;

    IF fs.isRemote THEN { -- get remote pathname.  also get name and type
      IF stringWasNil THEN
         itemContext.remoteNameString ←  NSString.MakeString[z, remoteNameAllocate];
      NSFile.GetAttributes[file, [[name: changeData.giveFeedback, type: changeData.giveFeedback, pathname: TRUE]], @attributes,
      itemContext.session];
```

```
    retryCount ← 0;
      -- get remotepathname
    vpn ← [pathname: attributes.pathname, service: fileRef.service];
    itemContext.remoteNameString ←
      NSFileName.AppendVPNToString[s: itemContext.remoteNameString, vpn: @vpn,
        resetLengthFirst: TRUE
        ! NSString.StringBoundsFault =>
          {oldLength: CARDINAL ← old.length;
           NSString.FreeString[z, itemContext.remoteNameString];
           new ← NSString.MakeString[z, oldLength+increaseBy];
           RESUME[itemContext.remoteNameString ← new]}
       ];


    stringWords ← NSFile.EncodeString[itemContext.remoteNameString];
    attr[0] ←
      [extended[type: BWSAttributeTypes.remoteName, value: stringWords]];
    attrList ← DESCRIPTOR[attr];
    } -- isRemote
  ELSE IF changeData.giveFeedback THEN -- name and type for feedback?
    << note: we don't use Containee.GetCachedName, even though we could,
       for two reasons: files from containers are not as likely to be in
       the cache, and we don't really want to put remote names in the cache. >>
    NSFile.GetAttributes[file, [[name: TRUE, type: TRUE]], @attributes, itemContext.session];

  IF changeData.giveFeedback THEN {
    nameRb: XString.ReaderBody ← XString.FromNSString[attributes.name];
    data: Containee.Data ← [itemContext.itemRef];
    nameAndIcon: XString.ReaderBody ← IconAndName[@data, @nameRb, attributes.type];
    Attention.Post[s: @nameAndIcon, clear: FALSE];
    XString.FreeReaderBytes[@nameAndIcon, z]};
  retryCount ← 0;

  BEGIN ENABLE BEGIN
    UNWIND => IF inLock THEN {Exit[fs, 1600]; inLock ← FALSE};
    NSFile.Error => IF error = [insertion[positionUnavailable]] THEN {
      orderAttr: NSFile.AttributesRecord ← TRASH;
      attributes: ARRAY [0..1) OF NSFile.Attribute;
      NSFile.GetAttributes[dest, [interpreted: [ordering: TRUE]], @orderAttr, itemContext.session];
      attributes[0] ← [ordering[orderAttr.ordering]];
      NSFile.ChangeAttributes[dest, DESCRIPTOR [attributes], itemContext.session];
      RETRY}
    ELSE IF error = [access[fileInUse]] AND ~GetAbort[] AND retryCount < 6 THEN {
          retryCount ← retryCount+1;
          RETRY};
    END; -- enable

  SELECT op FROM
    copy => {
      IF fileRef = destRef↑ THEN { -- Can't do a move to itself!
        CloseFiles[];
        Selection.Error [invalidOperation]; }
      ELSE {
        newFile: NSFile.Handle;
        newFile ← NSFile.Copy [file, dest, attrList,[timeout: fileTimeout], itemContext.session];
        itemContext.itemRef ← GetReference[newFile]; -- Return the reference for the new file
        NSFile.Close [newFile, itemContext.session];
        IF changeData.busy THEN {
          Enter[fs,, 1600]; inLock ← TRUE;
          [] ← SetBusyInternal[fs,
            ContainerCache.IndexFromMark[changeData.item], FALSE,
            changeData.callersChangeProc, changeData.callersChangeProcData
              ! UNWIND => {Exit[fs, 1600]; inLock ← FALSE}];
          Exit[fs, 1600]; inLock ← FALSE;
          changeData.busy ← FALSE; -- so FreeFile doesn't call again
          };
         };
        };
    move => {
      changeInfo: ContainerSource.ChangeInfo;


      IF doOperation THEN {
        IF fileRef = destRef↑ THEN { -- Can't do a move to itself!
          NSFile.Close [file, itemContext.session];
          NSFile.Close [dest, itemContext.session];
          Selection.Error [invalidOperation]; };
        BEGIN -- 8323
        controls: NSFile.Controls =
          NSFile.GetControls[fs.parentHandle, [access: TRUE], fs.session];
        IF ~controls.access[remove] THEN  -- don't even initiate the Move
          NSFile.Error[[access[accessRightsInsufficient]]];
        END;
        NSFile.Move [file, dest, attrList, itemContext.session
          <<! NSFile.Error => don't update cache or call changeProc>>];
      -- IF a move actually happened, i.e. no errors, THEN
      --   call the changeProc with a deleted item.
      --   Also delete the item from the cache.
        {oldRef: Containee.Data ← [itemContext.itemRef];

        itemContext.itemRef ← GetReference [file]; -- Return the reference for the moved file
        IF itemContext.itemRef.service # oldRef.reference.service
          -- if moved system elements, remove from cache
          THEN Containee.InvalidateCache[@oldRef]};

        IF changeData.busy THEN {
          Enter[fs,, 1600]; inLock ← TRUE;
```

```
            changeItem ← ContainerCache.IndexFromMark[changeData.item];
            [] ← SetBusyInternal[fs, changeItem, FALSE];
                -- don't call changeProc
            Exit[fs, 1600]; inLock ← FALSE;

            changeData.busy ← FALSE; -- so FreeFile doesn't call again
            -- delete item now: we can't mark busy anymore, so if repaint happens, it looks wrong
            };
          }; --  doOperation

        << Don't actually do delete yet, just make a note of it>>
        Enter[fs,, 1600]; inLock ← TRUE;
        changeItem ← ContainerCache.IndexFromMark[changeData.item];
        IF changeData.deleteItems  = NIL THEN {
          -- this is not an enumeration, (or it's a background move)
          --   so we have to delete the item here
          ContainerCache.DeleteNItems [fs.cache, changeItem, 1];
          fs.length ← fs.length-1;

          changeInfo ← [var: delete[deleteInfo: [afterItem: changeItem, nItems: 1]] ];
          IF changeData.callersChangeProc # NIL THEN
            changeData.callersChangeProc
              [changeData.callersChangeProcData, changeInfo]; }

        ELSE IF (changeData.deleteItems↑ = NIL) OR ((ContainerCache.IndexFromMark[changeData.deleteItems↑↑.firstItem] +
        changeData.deleteItems↑↑.count) # changeItem) THEN {
          deleteTemp: DeleteItemsHandle ← z.NEW[
            DeleteItemsData ← [firstItem: ContainerCache.SetMark[fs.cache, changeItem], count: 1, next: changeData.deleteItems↑]];
          changeData.deleteItems↑ ← deleteTemp}
        ELSE
          changeData.deleteItems↑↑.count ← changeData.deleteItems↑↑.count + 1;
        IF inLock THEN {Exit[fs, 1600]; inLock ← FALSE};
        };


      ENDCASE;
    END; -- enable unwind Exit

    IF inLock THEN {Exit[fs, 1600]; inLock ← FALSE};
    IF fs.isRemote OR changeData.giveFeedback THEN
      NSFile.ClearAttributes[@attributes];
    CloseFiles[];
    IF fs.isRemote THEN {
      NSFile.FreeWords[stringWords];
      IF stringWasNil THEN {
        NSString.FreeString[z, itemContext.remoteNameString];
        itemContext.remoteNameString ← NSString.nullString}};
    END; -- enable
    END; -- CopyMoveFile

  FixupPosition: PROC [
    fs: FS, file: NSFile.Reference, session: NSFile.Session, parentHandle: NSFile.Handle, afterHint: ContainerSource.ItemIndex,
    addAtEnd: BOOLEAN, readers: ReaderSeqPtr, writers: WriterSeqPtr]
    RETURNS [newPosition: ContainerSource.ItemIndex] =
    << addAtEnd supports background copy into a container: in this case,
       we only add new items at the end to avoid perturbing earlier items.
       If the ctnr is unsorted, however, we still change attributes to put the
       new file in the correct place (afterHint)>>

    BEGIN
    -- If the parent is unsorted (i.e. sorted by position), then this proc puts file after afterHint.
    -- WE MUST BE LOCKED TO CALL THIS PROCEDURE!

    attrList: ARRAY [0..1) OF NSFile.Attribute;
    movedHandle: NSFile.Handle ← NSFile.OpenByReference [file,, session];
    data: Containee.Data ← [file];
    afterHandle: NSFile.Handle;
    attributes: NSFile.AttributesRecord;
    addData: ContainerCache.AddData;
    itemData: ItemFileData;  -- not used except to pass into BuildRow
    insertTheItem: BOOLEAN ← FALSE; -- if true, insert it instead of appending
    gotPosition: BOOLEAN ← FALSE;
    afterInfo: ItemFileDataHandle;
    retryCount: CARDINAL ← 0;

    GetNextFileProc: NSFile.AttributesProc = BEGIN
      itemData: ItemFileDataHandle;

      -- find itemindex of item with same position as given by NSFile
      FOR i: ContainerSource.ItemIndex IN [0..fs.length) DO
        itemData ← GetItemDataInternal [fs.cache, i];
        IF itemData = NIL THEN EXIT; -- patch to fix AR #7029
        IF itemData.id = attributes.fileID THEN {
          afterHint ← IF i = 0 THEN ContainerSource.beforeItemZero ELSE i-1;
          insertTheItem ← TRUE;
          EXIT};
        ENDLOOP;
      END; -- GetNextFileProc

    BEGIN ENABLE {
      <<NSFile.Error => IF error = [handle[obsolete]] THEN
          IF RevalidateParent[fs, retryCount] THEN {
            retryCount ← retryCount+1;
            RETRY};>>
      UNWIND => NSFile.Close[movedHandle, session ! NSFile.Error, Courier.Error => CONTINUE];
      };
```

```
    IF fs.ascendingOrDescending THEN {
      afterInfo ← IF afterHint # ContainerSource.nullItem
        THEN GetItemDataInternal [fs.cache,
          IF afterHint = ContainerSource.beforeItemZero THEN 0 ELSE afterHint]
        ELSE NIL;

      IF afterInfo # NIL THEN {
        openedFile: BOOLEAN ← FALSE;
        BEGIN ENABLE NSFile.Error, Courier.Error => {
         addAtEnd ← TRUE;
         IF openedFile THEN NSFile.Close[afterHandle ! NSFile.Error, Courier.Error => CONTINUE];
         PostMsg[DtCtnrMessages.kcouldNotInsertAtPosition];
         CONTINUE;
         }; -- enable


        afterHandle ← NSFile.OpenChild [directory: parentHandle, id: afterInfo.id, session: session];
        openedFile ← TRUE;
        NSFile.GetAttributes [afterHandle, [[position: TRUE]], @attributes, session];
        attrList[0] ← IF afterHint = ContainerSource.beforeItemZero
          THEN [ position[NSFile.firstPosition] ]
          ELSE [ position[attributes.position] ];
        gotPosition ← TRUE;
        NSFile.Close [afterHandle, session];
        insertTheItem ← TRUE;
        END; -- enable
        };
      } -- ascendingOrDescending
    ELSE { -- sorted order
      -- get position of newly moved item
      NSFile.GetAttributes [movedHandle, [[position: TRUE]], @attributes, session];

      NSFile.List [directory: fs.parentHandle,
        proc: GetNextFileProc, selections: [[fileID: TRUE]],
        scope: [ count: 1, direction: forward,
                 filter: [greater [[position[attributes.position]]]]],
        session: session];
    }; -- sorted order


    NSFile.GetAttributes [movedHandle, fs.selections, @attributes, session];
    addData ← BuildRow [fs, writers, readers, @itemData, @data, @attributes];
    IF insertTheItem AND ~addAtEnd
      THEN {
        [] ← ContainerCache.InsertItem [
          cache: fs.cache,
          before: IF afterHint = ContainerSource.beforeItemZero
            THEN 0 ELSE afterHint+1,
          addData: addData];
        newPosition ← IF afterHint = ContainerSource.beforeItemZero THEN 0 ELSE afterHint+1}
      ELSE {[] ← ContainerCache.AppendItem [fs.cache, addData];
            newPosition ← fs.length};

    NSFile.ClearAttributes[@attributes];

    -- if inserted into an unsorted folder, update item with new position
    IF fs.ascendingOrDescending AND afterInfo # NIL AND gotPosition THEN
      BEGIN ENABLE
       NSFile.Error => IF error = [insertion[positionUnavailable]] THEN {
        orderAttr: NSFile.AttributesRecord ← TRASH;
        attributes: ARRAY [0..1) OF NSFile.Attribute;
        NSFile.GetAttributes[fs.parentHandle, [interpreted: [ordering: TRUE]], @orderAttr, session];
        attributes[0] ← [ordering[orderAttr.ordering]];
        NSFile.ChangeAttributes[fs.parentHandle, DESCRIPTOR [attributes], session];
        RETRY};

       NSFile.ChangeAttributes [movedHandle, DESCRIPTOR[attrList], session];
       END; -- begin enable
    NSFile.Close [movedHandle, session];
    END; -- ENABLE
    END; -- FixupPosition

  FreeFile: Selection.ValueFreeProc =
    BEGIN
    IF v.value # NIL THEN {
      itemContext: ConvertItemContext ← v.value;
      IF itemContext.freeContext THEN {
        IF itemContext.destHandle # NSFile.nullHandle THEN {
          NSFile.Close[itemContext.destHandle, itemContext.session
              ! NSFile.Error, Courier.Error => CONTINUE];
          NSFile.Close[itemContext.parentHandle, itemContext.session
              ! NSFile.Error, Courier.Error => CONTINUE];
          SessionCache.ReturnSession[itemContext.session
              ! NSFile.Error, Courier.Error => CONTINUE]};
        z.FREE [@itemContext]}};

    IF v.context # NIL THEN {
      changeData: LONG POINTER TO ChangeProcData ← v.context;
      IF changeData.busy THEN  { -- mark unbusy
        Enter[changeData.fs,, 1700];
        [] ← SetBusyInternal[changeData.fs,
             ContainerCache.IndexFromMark[changeData.item], FALSE, changeData.callersChangeProc, changeData.callersChangeProcData !
             UNWIND => Exit[changeData.fs, 1700]];
        Exit[changeData.fs, 1700]};
      ContainerCache.FreeMark[changeData.item];
      z.FREE[@changeData]
      };
```

```
    END;

  FSFreeMark: PUBLIC ContainerSourceExtra.FreeMarkProc =
    {IF mark # NIL THEN ContainerCache.FreeMark[mark]};

  FSIndexFromMark: PUBLIC ContainerSourceExtra.IndexFromMarkProc = {
    fs: FS = ValidFileSource[source];
    IF lockSource THEN Enter[fs, getBetweenCallsLock, 100];
    index ← IF mark # NIL THEN ContainerCache.IndexFromMark[mark]
      ELSE ContainerSource.nullItem;
    -- don't unlock: that will be done in callbacks.
    };

  FSMoveMark: PUBLIC ContainerSourceExtra.MoveOrCreateMarkProc = {
    fs: FS = ValidFileSource[source];
    Enter[fs, lockIfNotBetweenCalls, 1800];
    IF mark = NIL THEN {
      newMark ← ContainerCache.SetMark[fs.cache, newIndex]}
    ELSE {
      ContainerCache.MoveMark[mark, newIndex];
      newMark ← mark};
    Exit[fs, 1800];
    };

  FSSetMark: PUBLIC ContainerSourceExtra.SetMarkProc = {
    fs: FS = ValidFileSource[source];
    Enter[fs, lockIfNotBetweenCalls, 1900];
    mark ← ContainerCache.SetMark[fs.cache, index];
    Exit[fs, 1900]};

  IsBusy: PUBLIC ContainerSourceExtra.IsBusyProc = {
    fs: FS ← ValidFileSource[source];
    RETURN [IsBusyInternal[fs, item]]};

  IsBusyInternal: PROCEDURE [fs: FS, item: ContainerSource.ItemIndex]
    RETURNS [busy: BOOLEAN] = {
    itemData: ItemFileDataHandle ← GetItemDataInternal [fs.cache, item];
    cd: Containee.Data;
    IF itemData = NIL THEN ERROR ContainerSource.Error[noSuchItem];
    -- 9804: don't use NSFile.MakeReference here: it's VERY slow (8ms/call) in
    -- svs 8.0 (fixed in 10.0), and
    --  we can assume that the service from the parent id is a good one
    cd ← [reference: [itemData.id, fs.parentReference.service]];
    RETURN [ContaineeExtra.GetCachedBusy[@cd]]};

  IconAndName: PUBLIC PROC [data: Containee.DataHandle, name: XString.Reader,
    type: NSFile.Type, doPadding: BOOLEAN ← TRUE]
   RETURNS [rb: XString.ReaderBody] = {
   wb: XString.WriterBody ← XString.NewWriterBody[maxLength: XString.ByteLength[name]+3, z: z];
   smallPictureProc: Containee.SmallPictureProc ← Containee.GetImplementation[
      type].smallPictureProc;
   pad: XString.ReaderBody ← XString.FromSTRING["  "L];

   XString.AppendChar[to: @wb, c: smallPictureProc[
      data: data, type: type, normalOrReference: normal]];
   XString.AppendReader[to: @wb, from: name];
   IF doPadding THEN XString.AppendReader[to: @wb, from: @pad];
   rb ← (XString.ReaderFromWriter[@wb])↑;
   }; -- IconAndName

  IconAndNameFromFS: PROC [fs: FS] RETURNS [rb: XString.ReaderBody] = {
    attributes: NSFile.AttributesRecord ← TRASH;
    nameRb: XString.ReaderBody;
    data: Containee.Data ← [fs.parentReference];
    retryCount: CARDINAL ← 0;
    myProcess: PROCESS ← NIL;
    background: BOOLEAN;

    GetAbort: PROCEDURE RETURNS [BOOLEAN] =    -- 14183
      BEGIN
      IF myProcess = NIL THEN {
        myProcess ← Process.GetCurrent[];
        background ← myProcess # TIPXX.GetNotifierProcess[]};
        RETURN[IF background THEN BackgroundProcess.UserAbort[] ELSE TIP.UserAbort[NIL]];
      END;

    NSFile.ChangeControls[fs.parentHandle, [timeout: TRUE], [timeout: fileTimeout], fs.session];
    NSFile.GetAttributes[fs.parentHandle, [[name: TRUE, type: TRUE]], @attributes, fs.session !
      NSFile.Error => IF error = [access[fileInUse]] AND ~GetAbort[] AND retryCount < 6 THEN {
        retryCount ← retryCount +1; RETRY}];
    NSFile.ChangeControls[fs.parentHandle, [timeout: TRUE], [timeout: NSFile.defaultTimeout], fs.session];
    nameRb ← XString.FromNSString[attributes.name];
    rb ← IconAndName[@data, @nameRb, attributes.type, FALSE];
    NSFile.ClearAttributes[@attributes]};

  NewInsertInfo: PROC [old: InsertInfoDesc, items: CARDINAL, copy: BOOLEAN]
    RETURNS [new: InsertInfoDesc] = {
    new ← DESCRIPTOR[
      z.NEW[InsertInfoSequence[items]],
      items];
    IF copy THEN {
      oldBase: LONG POINTER ← BASE[old];

      Inline.LongCOPY [
        from: oldBase,
        to: BASE[new],
        nwords: InsertInfoSequence[old.LENGTH].SIZE];
```

```
      z.FREE[@oldBase]};
    };

  PostMsg: PROC [key: XMessage.MsgKey, clear: BOOLEAN ← TRUE] =
    BEGIN
    space: XString.Character = XChar.Make[XCharSets.Sets.latin.ORD, XCharSet0.Codes0.space.ORD];
    msg: XString.ReaderBody ← XMessage.Get [h: DtCtnrMessages.GetMessageHandle[], msgKey: key];
    msgwb: XString.WriterBody ← XString.CopyToNewWriterBody [r: @msg, z: z, extra: 1];
    XString.AppendChar[@msgwb, space];
    msg ← XString.ReaderFromWriter [@msgwb]↑;

    Attention.Post [s: @msg, clear: clear];
    XString.FreeWriterBytes [@msgwb];
    END;

  RebuildItem: PUBLIC PROCEDURE
    [source: ContainerSource.Handle, item: ContainerSource.ItemIndex] =
    BEGIN
    fs: FS ← ValidFileSource[source];
    Enter[fs,, 2000];
    RebuildSingleItem[fs, item ! UNWIND => Exit[fs, 2000]];
    Exit[fs, 2000]
    END;

  RebuildSingleItem: PROCEDURE [fs: FS, item: ContainerSource.ItemIndex] =
    BEGIN
    -- MUST BE LOCKED TO CALL THIS ROUTINE
    attributes: NSFile.AttributesRecord;
    fh: NSFile.Handle;
    writers: LONG POINTER TO WriterSeq ← AllocateWriters [fs.columns.length];
    readers: LONG POINTER TO ReaderSeq ← z.NEW [ReaderSeq[fs.columns.length]];
    addData: ContainerCache.AddData;
    containeeData: Containee.Data;
    itemDataHandle: ItemFileDataHandle ← GetItemDataInternal [fs.cache, item];
    itemDataObject: ItemFileData;

    CleanUp: PROC = {
      NSFile.Close [fh ! NSFile.Error, Courier.Error => CONTINUE];
      FreeWriters [writers];
      z.FREE [@readers]};

    BEGIN ENABLE UNWIND => CleanUp[];
    IF itemDataHandle = NIL THEN ERROR ContainerSource.Error[noSuchItem];
    itemDataObject ← itemDataHandle↑;
    containeeData ← [NSFile.MakeReference[
      fileID: itemDataHandle.id,
      service: fs.parentReference.service]];

    fh ← NSFile.OpenByReference [containeeData.reference];
      -- don't use non-default session: documents, for example, calls change
      -- proc with exclusive lock in the default session.
    -- Update the cache, but not until we see if we can open the file
    NSFile.GetAttributes [fh, fs.selections, @attributes];
    Containee.InvalidateCache[@containeeData];
    addData ← BuildRow [fs, writers, readers, @itemDataObject, @containeeData, @attributes];
    NSFile.ClearAttributes[@attributes];
    [] ← ContainerCache.ReplaceItem [
      cache: fs.cache,
      item: item,
      addData: addData];
    CleanUp[];
    END;
    END;

  SetBusy: PUBLIC ContainerSourceExtra.SetBusyProc = {
    fs: FS ← ValidFileSource[source];
    RETURN[SetBusyInternal[fs, item, newBusyState]]};

  SetBusyInternal: PROCEDURE [fs: FS, item: ContainerSource.ItemIndex, newBusyState: BOOLEAN, changeProc: ContainerSource.ChangeProc ←
  NIL, changeProcData: LONG POINTER ← NIL]
    RETURNS [succeeded: BOOLEAN] = {
    changeInfo: ContainerSource.ChangeInfo;
    IF item # ContainerSource.nullItem THEN {
      Enter[fs, lockIfNotBetweenCalls, 2100];
      BEGIN ENABLE UNWIND => Exit[fs, 2100];
      itemData: ItemFileDataHandle ← GetItemDataInternal [fs.cache, item];
      -- make ref is VERY slow
      cd: Containee.Data ← [reference:
          [itemData.id, fs.parentReference.service]];

      IF succeeded ← ContaineeExtra.GetCachedBusy[@cd] # newBusyState
        THEN ContaineeExtra.SetCachedBusy[@cd, newBusyState];
      -- $$ probably need to move call back out of monitor to allow callback to IsItemBusy
      changeInfo ← [var: replace[item] ];
      IF changeProc # NIL THEN changeProc[changeProcData, changeInfo];
      Exit[fs, 2100];
      END;-- enable
      }
    ELSE -- lock source
      IF newBusyState THEN Enter[fs,, 2150] ELSE Exit[fs, 2150]};

  SetNameForProcess: PUBLIC PROC [iconAndName: XString.ReaderBody] = {
    jobTitle: XString.ReaderBody ← BackgroundProcessExtra.GetName[];
    wb: XString.WriterBody ← XString.NewWriterBody[XString.ByteLength[@jobTitle] + XString.ByteLength[@iconAndName]+10, z];
    XMessage.ComposeOne[@jobTitle, @wb, @iconAndName];
    BackgroundProcessExtra.SetName[newName: XString.ReaderFromWriter[@wb]];
    XString.FreeWriterBytes[@wb]};
```

```
  SetTakeFilterProc: PUBLIC --Extra4-- PROCEDURE [fs: ContainerSource.Handle,
    p: FileContainerSourceExtra4.TakeFilterProc] = {
    realFS: FS ← ValidFileSource[fs];
    realFS.takeFilter ← p;
    };

  SingleItemChangeProc: PUBLIC Containee.ChangeProc
<< [changeProcData: LONG POINTER, data: Containee.DataHandle ← NIL, changedAttributes: NSFile.Selections ← [noInterpretedSelections,
noExtendedSelections], noChanges: BOOLEAN ← FALSE];
>> =

    BEGIN
    -- This proc is called when a single item has changed.  This proc is passed into a Containee.Implementation.genericProc, so it will
    typically be called when a property sheet is being taken down, or when a SWS is being closed (and a containee's
    "date-of-last-change" changes).

    myChangeProcData: LONG POINTER TO ChangeProcData = changeProcData;
    fs: FS = myChangeProcData.fs;
    changeInfo: ContainerSource.ChangeInfo;

    -- If any of the changedAttributes selections is also in fs.selections, then it means one of the attibutes used to construct this
    item has changed, so we should rebuild the item.

    Enter[fs,, 2200];
    BEGIN ENABLE UNWIND => Exit[fs, 2200];
    IF myChangeProcData.background OR ~noChanges AND AnySelectionsInCommon [fs.selections, changedAttributes] THEN
      BEGIN
      item: ContainerSource.ItemIndex ←
        ContainerCache.IndexFromMark[myChangeProcData.item];
      RebuildSingleItem[fs, item
        ! NSFile.Error, Courier.Error => CONTINUE];
      -- Call the caller's change proc.
      IF myChangeProcData.background THEN [] ← SetBusyInternal[fs, item, FALSE];
      changeInfo ← [var: replace[item] ];
      IF myChangeProcData.callersChangeProc # NIL THEN
        myChangeProcData.callersChangeProc [myChangeProcData.callersChangeProcData, changeInfo];

      END
    ELSE { -- call caller change proc with noChanges
      changeInfo ← [var: noChanges []];
      IF myChangeProcData.callersChangeProc # NIL THEN
        myChangeProcData.callersChangeProc [myChangeProcData.callersChangeProcData, changeInfo];
      };

    Exit[fs, 2200];
    END; -- enable

    IF ~myChangeProcData.attachedToContext THEN {
      ContainerCache.FreeMark[myChangeProcData.item];
      z.FREE[@changeProcData]}
    END;

  END.

-- See ImplA for rest of history
22-Jan-85 - Holbrook - Error handling
18-Mar-85 - Holbrook - Enumeration stuff in ConvertFileProc: for each icon, try to convert to single target, if that fails, try
enumeration
28-Mar-85 - Holbrook - Reduce global frame size
 4-Apr-85 - Holbrook - ARs 13781, 14085
19-Apr-85 - Holbrook - AR 13740 (misleading erro rmessage), 13131 (ChangeProc for folders cause ADDRESS FAULT
 1-May-85 - Holbrook - Fix last part of Enumeration stuff: if queried for convert to enumeration, ask containee instead of saying
 impossible
17-May-85 - Holbrook - Add RebuildItem (AR 14542)
30-May-85 - Holbrook - Clean up code slightly, remove SortedInsertBehavior stuff
18-Jun-85 - SAJohnson.es - In ConvertFileItem, if it's an enumeration get the containee implementation for each item so the correct
containee procs will be called.
19-Jun-85 - SAJohnson.es - Kludgy fix for AFR 15968 for PCE and other emulators.
 8-Jul-85 - Holbrook - Use cached parent handle
18-Jul-85 - Holbrook - Cache destination handle during copy
30-Jul-85 - Holbrook - Merge in Beta fixes 16972 (don't confirm in Wastebasket.Take) 17240 (catch Courier.Error in NSFile.Close)
 2-Aug-85 - Holbrook - 17190 (make sure destination files closed if copy fails)
 9-Aug-85 - Holbrook - 16320: construct insertInfo change proc data so container window can highlight object moved into containers
28-Aug-85 - Holbrook - 18799: when moving back into the same source, don't actually do an NSFile.Move, which increments the version
number.
16-Sep-85 - Holbrook - 19812: closing ctnr when server down crashes.  Put catch in SingleItemChangeProc; 20181: make move or copy wait
until cache is done enumerating.
 5-Nov-85 - Breisacher - Check for NIL element.value in TakeFile in FileTake. AR 21918.
14-Nov-85 - Holbrook - 22768 subtract correct amount from length in Update proc.
17-Mar-86 - Mita -  Support Iconpopup free menu
23-May-86 - Riggle - 7029: occasional crashes when copying icons into a file drawer.  Put a test for itemData = NIL into FixupPosition[].
23-Jun-86 - Holbrook - 4.3 background stuff
 7-Aug-86 - Holbrook - monitoring for concurrency
17-Sep-86 - Holbrook - 7975 Can crash copying into container if insufficient rights to get reference
 3-Feb-87 - Holbrook - 9804 Container scrolling slow: take NSFile.MakeReference call out of IsBusyInternal
 9-Mar-87 - Holbrook - 10747 change TakeFile to call display changed with all variant if more than one item inserted; inserted Hanzel's
 4.2.4 fix for 10093: Changes for VOA 'disk nibbling' problem; define locks in NSFile.Open call in CopyMoveFile.
24-Mar-87 - Holbrook - Move retry catch code in CopyMove proc to cover all other filing ops besides copy/move
20-Apr-87 - Holbrook - 11485 in CopyMoveFile move catch phrases for fileInUse to around every open call instead of around entire block;
8323 don't initiate move if access rights not sufficient.
14-Sep-87 - ph - 14183 Check for user abort whenever we catch fileInUse and retry; 14431 get cache for copies from SessionCache
16-Nov-87 - ph - 15722 In endcase where we ask object to try to enumerate, make sure we can abort
17-Nov-87 - LFB - AR 13817 - add takeFilter stuff in TakeFile.
24-Nov-87 - ph - 16022 Don't unbusy icons during enumerate if target is something other than file
```

```
-- File: ListFilesImpl.mesa
-- last edited by MSchneider              24-Jul-89 11:08:28

-- Copyright (C) 1989 by Xerox Corporation

-- This module registers an Attention Menu command in ViewPoint. This command puts up a
-- sheet from which the user can list all of the objects on the current user's desktop, and
-- puts the information into a file which is placed on the desktop.


DIRECTORY
  Attention,
  BackgroundProcess,
  BWSZone,
  DocInterchangeDefs,
  DocInterchangePropsDefs,
  FormWindow,
  Heap,
  MenuData,
  NSFile,
  NSFileStream,
  NSString,
  Process,
  PropertySheet,
  Runtime,
  StarDesktop,
  StarWindowShell,
  Stream,
  System,
  Window,
  XFormat,
  XString;

ListFilesImpl: PROGRAM
  IMPORTS Attention, BackgroundProcess, BWSZone, DocInterchangeDefs, DocInterchangePropsDefs, FormWindow, Heap, MenuData, NSFile,
  NSFileStream, NSString, Process, PropertySheet, Runtime, StarDesktop, StarWindowShell, Stream, XFormat, XString =
  BEGIN OPEN FW: FormWindow, SWS: StarWindowShell;

  Items: TYPE = {outputType, typeOption, dateOption, sizeOption, sizeType};
  OutputType: TYPE = {simple, vp};
  SizeType: TYPE = {bytes, pages};

  SheetData: TYPE = LONG POINTER TO SheetDataObject;
  SheetDataObject: TYPE = RECORD [
    zone: UNCOUNTED ZONE,
    outputType: OutputType ← simple,
    type: BOOLEAN ← FALSE,
    date: BOOLEAN ← FALSE,
    size: BOOLEAN ← FALSE,
    sizeType: SizeType ← pages];

  pointsPerSpace: CARDINAL = 6;
  pointsPerInch: CARDINAL = 72;

  ApplyAnyChanges: PROCEDURE [window: Window.Handle, myData: SheetData] RETURNS [ok: BOOLEAN ← TRUE] =
  BEGIN
    IF FW.HasAnyBeenChanged[window] THEN {
      FOR myItem: Items IN Items DO
        itemKey: FW.ItemKey = myItem.ORD;
        IF ~FW.HasBeenChanged[window, itemKey] THEN LOOP;
        SELECT myItem FROM
          outputType => myData.outputType ← VAL[FW.GetChoiceItemValue[window, itemKey]];
          typeOption => myData.type ← FW.GetBooleanItemValue[window, itemKey];
          dateOption => myData.date ← FW.GetBooleanItemValue[window, itemKey];
          sizeOption => myData.size ← FW.GetBooleanItemValue[window, itemKey];
          sizeType => myData.sizeType ← VAL[FW.GetChoiceItemValue[window, itemKey]];
        ENDCASE => ERROR;
      ENDLOOP};
  END; -- ApplyAnyChanges

  CreateObject: PROCEDURE [sheetData: SheetData] RETURNS [to: LONG POINTER, ok: BOOLEAN ← TRUE] =
  BEGIN
    IF sheetData.outputType = simple THEN {
      fileName: NSString.String ← NSString.StringFromMesaString["AllFiles.Listing"L];
      attributes: ARRAY [0..2) OF NSFile.Attribute ← [[name[fileName]], [type[2]]];
      file: NSFile.Handle ← NSFile.Create[directory: NSFile.nullHandle, attributes: DESCRIPTOR[attributes]];
      stream: NSFileStream.Handle ← NSFileStream.Create[file: file, closeOnDelete: FALSE];
      handle: XFormat.Handle ← BWSZone.shortLifetime.NEW[XFormat.Object];
      handle↑ ← XFormat.StreamObject[stream];
      to ← handle;
    }
    ELSE {
      fontProps: DocInterchangePropsDefs.FontPropsRecord;
      paraProps: DocInterchangePropsDefs.ParaPropsRecord;
      pageProps: DocInterchangePropsDefs.PagePropsRecord;
      start: DocInterchangeDefs.StartCreationStatus;
      doc: DocInterchangeDefs.Doc;
      tabArray: ARRAY [0..3) OF DocInterchangePropsDefs.TabStop ← ALL[DocInterchangePropsDefs.nullTabStop];
      includeArray: ARRAY [0..3) OF BOOLEAN ← [sheetData.type, sheetData.date, sheetData.size];
      nTabs: CARDINAL ← 0;
      IF ~Runtime.IsBound[LOOPHOLE[DocInterchangeDefs.StartCreation]] THEN {
        msg: XString.ReaderBody ← XString.FromSTRING["You must run the Document Editor before executing that operation."L];
        Attention.Post[@msg];
        RETURN[to: NIL, ok: FALSE]};
      DocInterchangePropsDefs.GetFontPropsDefaults[@fontProps];
      DocInterchangePropsDefs.GetParaPropsDefaults[@paraProps];
      DocInterchangePropsDefs.GetPagePropsDefaults[@pageProps];
```

```
    -- font will be modern 10, with 1/2 inch page margins on all sides.
    fontProps.fontDesc.pointSize ← 10;
    pageProps.topMarginHeight ← pointsPerInch/2;
    pageProps.bottomMarginHeight ← pointsPerInch/2;
    pageProps.leftMarginWidth ← pointsPerInch/2;
    pageProps.rightMarginWidth ← pointsPerInch/2;
    -- set up tabs
    IF sheetData.type THEN {
      tabArray[nTabs].tabStopOffset ← SELECT TRUE FROM
        sheetData.date AND sheetData.size => 60*pointsPerSpace,
        sheetData.date => 72*pointsPerSpace,
        sheetData.size => 76*pointsPerSpace,
        ENDCASE => 89*pointsPerSpace;
      tabArray[nTabs].tabStopAlignment ← right;
      nTabs ← nTabs + 1};
    IF sheetData.date THEN {
      tabArray[nTabs].tabStopOffset ← IF sheetData.size THEN 62*pointsPerSpace ELSE 74*pointsPerSpace;
      nTabs ← nTabs + 1};
    IF sheetData.size THEN {
      tabArray[nTabs].tabStopOffset ← 89*pointsPerSpace;
      tabArray[nTabs].tabStopAlignment ← right;
      nTabs ← nTabs + 1};
    paraProps.tabStops ← DESCRIPTOR[BASE[tabArray], nTabs];
    [doc: doc, status: start] ← DocInterchangeDefs.StartCreation[
      paginateOption: simple, initialFontProps: @fontProps,
      initialParaProps: @paraProps, initialPageProps: @pageProps ! NSFile.Error => {
        IF error = [space[mediumFull]] THEN
          start ← notEnoughDiskSpace
        ELSE
          start ← lastAvailable;
        CONTINUE}];
    IF start = ok THEN {-- append headings
      string: XString.ReaderBody ← XString.FromSTRING["NAME"L];
      endContext: XString.Context ← XString.ComputeEndContext[@string];
      underlinedProps: DocInterchangePropsDefs.FontPropsRecord ← fontProps;
      underlinedProps.nUnderlines ← 1;
      DocInterchangeDefs.AppendText[to: [doc[doc]], text: @string, textEndContext: endContext, fontProps: @underlinedProps];
      IF sheetData.type THEN {
        string ← XString.FromSTRING["\tTYPE"L];
        endContext ← XString.ComputeEndContext[@string];
        DocInterchangeDefs.AppendText[to: [doc[doc]], text: @string, textEndContext: endContext]};
      IF sheetData.date THEN {
        string ← XString.FromSTRING["\tDATE"L];
        endContext ← XString.ComputeEndContext[@string];
        DocInterchangeDefs.AppendText[to: [doc[doc]], text: @string, textEndContext: endContext]};
      IF sheetData.size THEN {
        string ← XString.FromSTRING["\tSIZE"L];
        endContext ← XString.ComputeEndContext[@string];
        DocInterchangeDefs.AppendText[to: [doc[doc]], text: @string, textEndContext: endContext]};
      DocInterchangeDefs.AppendNewParagraph[to: [doc[doc]], fontProps: @fontProps]}
    ELSE {
      msg: XString.ReaderBody ← XString.FromSTRING["ERROR creating VP Document."L];
      Attention.Post[@msg];
      RETURN [to: NIL, ok: FALSE]};
    to ← doc};
  END; -- CreateObject

  DestroyObject:  PROCEDURE [to: LONG POINTER, output: OutputType] RETURNS [file: NSFile.Handle] =
  BEGIN
    IF output = simple THEN {
      handle: XFormat.Handle ← to;
      stream: Stream.Handle ← handle.data;
      BWSZone.shortLifetime.FREE[@handle];
      file ← NSFileStream.FileFromStream[[stream]];
      Stream.Delete[stream]}
    ELSE {
      doc: DocInterchangeDefs.Doc ← to;
      nameString: LONG STRING ← "AllFiles.Listing"L;
      nameNS: NSString.String ← NSString.StringFromMesaString[nameString];
      attributes: ARRAY [0..1) OF NSFile.Attribute ← [ [name[nameNS]] ];
      file ← DocInterchangeDefs.FinishCreation[docPtr: @doc].docFile;
      -- change the name
      NSFile.ChangeAttributes[file, DESCRIPTOR[attributes]]};
  END; -- DestroyObject

  DoLayout: FW.LayoutProc =
  BEGIN
    myData: SheetData = clientData;
    defaultSpace: CARDINAL = 5;
    line: FW.Line;
    tabStopInterval: CARDINAL = 50;
    extra: CARDINAL = 10;

    -- Set the tabs for FormWindow
    tabChoice: fixed FW.TabStops = [fixed[tabStopInterval]];
    FW.SetTabStops[window: window, tabStops: tabChoice];

    line ← FW.AppendLine[window, defaultSpace];
    FW.AppendItem[window: window, item: Items.outputType.ORD, line: line, preMargin: extra];
    line ← FW.AppendLine[window, defaultSpace];
    FW.AppendItem[window: window, item: Items.typeOption.ORD, line: line, preMargin: extra];
    FW.AppendItem[window: window, item: Items.dateOption.ORD, line: line];
    FW.AppendItem[window: window, item: Items.sizeOption.ORD, line: line];
    line ← FW.AppendLine[window, defaultSpace];
    FW.AppendItem[window: window, item: Items.sizeType.ORD, line: line, preMargin: extra];
  END; -- DoLayout
```

```
ListFiles: PROCEDURE [sheetData: SheetData] =
BEGIN
  output: OutputType = sheetData.outputType;
  processName: XString.ReaderBody ← XString.FromSTRING[
    IF output = simple THEN "Listing Desktop Files to simple document"L
    ELSE "Listing Desktop Files to ViewPoint document"L];
  zone: UNCOUNTED ZONE ← sheetData.zone;

  DoListFiles: BackgroundProcess.CallBackProc =
  BEGIN
    nFiles: CARDINAL ← 0;
    selections: NSFile.Selections ← [interpreted: [name: TRUE, sizeInBytes: sheetData.size AND sheetData.sizeType = bytes, sizeInPages:
    sheetData.size AND sheetData.sizeType = pages, type: sheetData.type, createdOn: sheetData.date, isDirectory: TRUE, fileID: TRUE,
    service: TRUE]];
    desktopRef: NSFile.Reference ← StarDesktop.GetCurrentDesktopFile[];
    desktop: NSFile.Handle;
    typeTag: XString.ReaderBody ← XString.FromSTRING["TYPE: "L];
    sizeTag: XString.ReaderBody ← XString.FromSTRING["SIZE: "L];
    dateTag: XString.ReaderBody ← XString.FromSTRING["DATE: "L];
    separator: XString.ReaderBody ← XString.FromSTRING[", "L];
    tab: XString.ReaderBody ← XString.FromSTRING["\t"L];
    units: XString.ReaderBody ← XString.FromSTRING[IF sheetData.sizeType = bytes THEN " bytes"L ELSE " pages"L];
    depth: CARDINAL ← 0;
    doneMsg: XString.ReaderBody ← XString.FromSTRING[" Total Files Listed. DONE."L];
    to: LONG POINTER;
    file: NSFile.Handle;
    fileRef: NSFile.Reference;
    ok: BOOLEAN;

    PostAborted: PROCEDURE = INLINE
    BEGIN
      msg: XString.ReaderBody ← XString.FromSTRING["*** Listing Aborted by User ***"L];
      PutReader[to, @msg, sheetData.outputType];
      Attention.Post[@msg];
    END; -- PostAborted

    ListFiles: NSFile.AttributesProc =
    BEGIN
      length: CARDINAL ← NSString.LogicalLength[attributes.name];
      nameRB: XString.ReaderBody ← XString.FromNSString[attributes.name];
      IF BackgroundProcess.UserAbort[] THEN {
        PostAborted[];
        finalStatus ← aborted;
        continue ← FALSE;
        RETURN};
      PutBlanks[to, depth, output]; -- indenting
      PutNSString[to, attributes.name, output];
      IF sheetData.type THEN {
        IF output = simple THEN {
          PutReaderToSimple[to, @separator];
          PutReaderToSimple[to, @typeTag]}
        ELSE PutReaderToVPDoc[to, @tab];
        PutDecimal[to, attributes.type, output]};
      IF sheetData.date THEN {
        IF output = simple THEN {
          PutReaderToSimple[to, @separator];
          PutReaderToSimple[to, @dateTag]}
        ELSE PutReaderToVPDoc[to, @tab];
        PutDate[to, attributes.createdOn, output]};
      IF sheetData.size THEN {
        IF output = simple THEN {
          PutReaderToSimple[to, @separator];
          PutReaderToSimple[to, @sizeTag]}
        ELSE PutReaderToVPDoc[to, @tab];
        PutDecimal[to, IF sheetData.sizeType = bytes THEN attributes.sizeInBytes ELSE attributes.sizeInPages, output];
        PutReader[to, @units, output]};
      PutCR[to, output];
      nFiles ← nFiles + 1;
      IF nFiles MOD 10 = 0 THEN { -- post a message after every 10 files
        msg: XString.ReaderBody ← XString.FromSTRING[" Files Listed ... Continuing.  "L];
        XFormat.Decimal[Attention.formatHandle, nFiles];
        Attention.Post[s: @msg, clear: FALSE]};
      IF attributes.service.systemElement # NSFile.localSystemElement THEN { -- not a local file
        notLocal: XString.ReaderBody ← XString.FromSTRING["  -> REMOTE FILE"L];
        PutBlanks[to, depth+2, output];
        PutReader[to, @notLocal, output];
        PutCR[to, output]}
      ELSE IF attributes.isDirectory THEN {
        ref: NSFile.Reference ← [attributes.fileID, attributes.service];
        file: NSFile.Handle ← NSFile.OpenByReference[ref];
        depth ← depth + 4;
        NSFile.List[directory: file, proc: ListFiles, selections: selections];
        depth ← depth - 4;
        NSFile.Close[file]};
    END; -- ListFiles

    -- main code for DoFilingTest
    IF BackgroundProcess.UserAbort[] THEN {
      PostAborted[];
      RETURN[aborted]};
    finalStatus ← quietSuccess;
    [to, ok] ← CreateObject[sheetData];
    IF ~ok THEN RETURN[importantFailure];
    desktop ← NSFile.OpenByReference[desktopRef];
    NSFile.List[directory: desktop, proc: ListFiles, selections: selections];
    PutCR[to, output];
    PutDecimal[to, nFiles, output];
```

```
      PutReader[to, @doneMsg, output];
      PutCR[to, output];
      file ← DestroyObject[to, output];
      fileRef ← NSFile.GetReference[file];
      NSFile.Move[file: file, destination: desktop];
      NSFile.Close[file];
      NSFile.Close[desktop];
      StarDesktop.AddReferenceToDesktop[fileRef];
    END; -- DoListFiles

    -- mainline code for FilingTest
    Process.SetPriority[MAX[Process.GetPriority[], 1] - 1];
    [] ← BackgroundProcess.ManageMe[
      name: @processName, callBackProc: DoListFiles, abortable: FALSE];
    Heap.Delete[zone];
  END; -- ListFiles

  MakeItems: FW.MakeItemsProc =
  BEGIN
    myData: SheetData = clientData;

    -- output type
    BEGIN
    firstChoiceString: XString.ReaderBody ← XString.FromSTRING["Simple Document"L];
    secondChoiceString: XString.ReaderBody ← XString.FromSTRING["VP Document"L];
    choices: ARRAY [0..2) OF FW.ChoiceItem ← [ [string[0, firstChoiceString]], [string[1, secondChoiceString]]];
    tag: XString.ReaderBody ← XString.FromSTRING["Output Format"L];
    FW.MakeChoiceItem[
      window: window, myKey: Items.outputType.ORD, tag: @tag, values: DESCRIPTOR[choices],
      initChoice: SELECT myData.outputType FROM simple => 0, vp => 1, ENDCASE => ERROR];
    END;

    BEGIN -- type boolean
    label: XString.ReaderBody ← XString.FromSTRING["File Type"L];
    tag: XString.ReaderBody ← XString.FromSTRING["Display Options"L];
    FW.MakeBooleanItem[window: window, myKey: Items.typeOption.ORD,
      tag: @tag, label: [string[label]], initBoolean: myData.type];
    END;

    BEGIN -- date boolean
    label: XString.ReaderBody ← XString.FromSTRING["Create Date"L];
    FW.MakeBooleanItem[window: window, myKey: Items.dateOption.ORD,
      label: [string[label]], initBoolean: myData.date];
    END;

    BEGIN -- size boolean
    label: XString.ReaderBody ← XString.FromSTRING["File Size"L];
    FW.MakeBooleanItem[window: window, myKey: Items.sizeOption.ORD,
      label: [string[label]], initBoolean: myData.size, changeProc: SizeChangeProc];
    END;

    BEGIN -- size type
    firstChoiceString: XString.ReaderBody ← XString.FromSTRING["Bytes"L];
    secondChoiceString: XString.ReaderBody ← XString.FromSTRING["Pages"L];
    choices: ARRAY [0..2) OF FW.ChoiceItem ← [ [string[0, firstChoiceString]], [string[1, secondChoiceString]]];
    tag: XString.ReaderBody ← XString.FromSTRING["Display Size In"L];
    FW.MakeChoiceItem[
      window: window,  Key: Items.sizeType.ORD, tag: @tag, values: DESCRIPTOR[choices],
      initChoice: SEL  myData.sizeType FROM bytes => 0, pages => 1, ENDCASE => ERROR,
      visibility: IF myData.size THEN visible ELSE invisibleGhost];
    END;
  END; -- MakeItems

  MakeSheet: MenuData.MenuProc =
  BEGIN
    zone: UNCOUNTED ZONE ← Heap.Create[2];
    sheetData: SheetData ← zone.NEW[SheetDataObject];
    shell: SWS.Handle;
    title: XString.ReaderBody ← XString.FromSTRING["List Files Options"L];
    sheetData.zone ← zone;
    shell ← PropertySheet.Create[
      formWindowItems: MakeItems,
      menuItemProc: MenuItemProc,
      menuItems: [cancel: TRUE, start: TRUE],
      title: @title,
      size: [0, 0],
      formWindowItemsLayout: DoLayout,
      display: FALSE,
      clientData: sheetData];
    SWS.Push[shell];
  END; -- MakeSheet

  MenuItemProc: PropertySheet.MenuItemProc =
  BEGIN
    myData: SheetData ← clientData;
    SELECT menuItem FROM
      cancel => {
        zone: UNCOUNTED ZONE ← myData.zone;
        Heap.Delete[zone];
        ok ← TRUE};
      start => {
        ok ← ApplyAnyChanges[formWindow, myData];
        IF ok THEN {
          msg: XString.ReaderBody ← XString.FromSTRING["Listing Files in the Background."L];
          Attention.Post[@msg];
          Process.Detach[FORK ListFiles[myData]]}};
    ENDCASE => RETURN[ok: FALSE];
```

```
    END; -- MenuItemProc

    PutBlanks: PROCEDURE [to: LONG POINTER, number: CARDINAL, output: OutputType] =
    BEGIN
      IF output = simple THEN {
        h: XFormat.Handle ← to;
        XFormat.Blanks[to, number]}
      ELSE {
        doc: DocInterchangeDefs.Doc ← to;
        endContext: XString.Context;
        wb: XString.WriterBody ← XString.NewWriterBody[number, BWSZone.shortLifetime];
        object: XFormat.Object ← XFormat.WriterObject[@wb];
        XFormat.Blanks[@object, number];
        endContext ← XString.ComputeEndContext[XString.ReaderFromWriter[@wb]];
        DocInterchangeDefs.AppendText[to: [doc[doc]], text: XString.ReaderFromWriter[@wb], textEndContext: endContext];
        XString.FreeWriterBytes[@wb]};
    END; -- PutBlanks

    PutDate: PROCEDURE [to: LONG POINTER, date: System.GreenwichMeanTime, output: OutputType] =
    BEGIN
      IF output = simple THEN {
        h: XFormat.Handle ← to;
        XFormat.Date[to, date]}
      ELSE {
        doc: DocInterchangeDefs.Doc ← to;
        endContext: XString.Context;
        wb: XString.WriterBody ← XString.NewWriterBody[20, BWSZone.shortLifetime];
        object: XFormat.Object ← XFormat.WriterObject[@wb];
        XFormat.Date[@object, date];
        endContext ← XString.ComputeEndContext[XString.ReaderFromWriter[@wb]];
        DocInterchangeDefs.AppendText[to: [doc[doc]], text: XString.ReaderFromWriter[@wb], textEndContext: endContext];
        XString.FreeWriterBytes[@wb]};
    END; -- PutDate

    PutDecimal: PROCEDURE [to: LONG POINTER, number: LONG CARDINAL, output: OutputType] =
    BEGIN
      IF output = simple THEN {
        h: XFormat.Handle ← to;
        XFormat.Decimal[to, number]}
      ELSE {
        doc: DocInterchangeDefs.Doc ← to;
        endContext: XString.Context;
        wb: XString.WriterBody ← XString.NewWriterBody[10, BWSZone.shortLifetime];
        object: XFormat.Object ← XFormat.WriterObject[@wb];
        XFormat.Decimal[@object, number];
        endContext ← XString.ComputeEndContext[XString.ReaderFromWriter[@wb]];
        DocInterchangeDefs.AppendText[to: [doc[doc]], text: XString.ReaderFromWriter[@wb], textEndContext: endContext];
        XString.FreeWriterBytes[@wb]};
    END; -- PutDecimal

    PutNSString: PROCEDURE [to: LONG POINTER, string: NSString.String, output: OutputType] =
    BEGIN
      rb: XString.ReaderBody ← XString.FromNSString[string];
      PutReader[to, @rb, output];
    END; -- PutNSString

    PutCR: PROCEDURE [to: LONG POINTER, output: OutputType] =
    BEGIN
      IF output = simple THEN {
        h: XFormat.Handle ← to;
        XFormat.CR[to]}
      ELSE {
        doc: DocInterchangeDefs.Doc ← to;
        DocInterchangeDefs.AppendNewParagraph[to: [doc[doc]]]};
    END; -- PutCR

    PutReader: PROCEDURE [to: LONG POINTER, reader: XString.Reader, output: OutputType] =
    BEGIN
      IF output = simple THEN PutReaderToSimple[to, reader]
      ELSE PutReaderToVPDoc[to, reader];
    END; -- PutReader

    PutReaderToSimple: PROCEDURE [to: XFormat.Handle, reader: XString.Reader] = INLINE {
        XFormat.Reader[to, reader]};

    PutReaderToVPDoc: PROCEDURE [to: DocInterchangeDefs.Doc, reader: XString.Reader] = INLINE {
        endContext: XString.Context ← XString.ComputeEndContext[reader];
        DocInterchangeDefs.AppendText[to: [doc[to]], text: reader, textEndContext: endContext]};

    SizeChangeProc: FW.BooleanChangeProc =
    BEGIN
      FW.SetVisibility[window, Items.sizeType.ORD, IF newValue THEN visible ELSE invisibleGhost, TRUE];
    END; -- SizeChangeProc

    Init: PROCEDURE =
    BEGIN
      -- put the command in the Attention window menu
      listFiles: XString.ReaderBody ← XString.FromSTRING["List Files"L];
      menuItem: MenuData.ItemHandle ←
        MenuData.CreateItem[
          zone: BWSZone.permanent,
          name: @listFiles,
          proc: MakeSheet];
      Attention.AddMenuItem[menuItem];
    END; -- Init

    Init[];
```

```
END.

LOG
29-Jun-89  9:09:12 - MSchneider - CREATED
21-Jul-89 10:21:54 - MSchneider - Made the UI sheet-based
24-Jul-89  9:08:21 - MSchneider - Made the VPDoc output columnized using tabs
```

```
-- File: DeleterImpl.mesa - last edit:
-- Minamizaki:IWA:Fuji Xerox   30-Apr-88 16:40 JST

-- Copyright (C) 1988 by Fuji Xerox Co., Ltd, Tokyo, Japan. All rights reserved.

DIRECTORY
  Atom          USING [ATOM, <<GetProp,>> MakeAtom, null],
--Auth          USING [IdentityHandle],
  DirectoryFile USING [GetUserFileDirectory],
  Event         USING [AddDependencies, AgentProcedure, EventType],
  Heap          USING [systemZone],
  NSFile        USING [AttributesProc, Close, DeleteChild, Error, Handle, List,
  <<Logoff, Logon,>> OpenByName, OpenByReference, Reference<<, Session>>],
--NSFileExtra   USING [EnablePrivileges],
  NSString      USING [FreeString, String],
  Process       USING [<<Detach,>> priorityBackground, SetPriority],
  StarDesktop   USING [logoff],
  TIP           USING [Error],
  WastebasketOps USING [GetMessageHandle, MessageKey],
  XMessage      USING [Get, Handle],
  XString       USING [NSStringFromReader, ReaderBody];

DeleterImpl: PROGRAM
  IMPORTS StarDesktop, WastebasketOps, NSFile, NSString, Atom, --NSFileExtra,
    Process, Event, DirectoryFile, XMessage, XString, Heap, TIP = {

<<<<<<<<<<<<<<<<<<<<<<<<  DeleteChild  >>>>>>>>>>>>>>>>>>>>>>>>

  Deleter: PROC = {
    mh: XMessage.Handle ← WastebasketOps.GetMessageHandle [];
    wbParentRef: NSFile.Reference ← DirectoryFile.GetUserFileDirectory [];
    parentFh: NSFile.Handle ← NSFile.OpenByReference [wbParentRef];
    wbNameRB: XString.ReaderBody ←
      XMessage.Get [mh, WastebasketOps.MessageKey.wbName.ORD];
    nsWbname: NSString.String ←
      XString.NSStringFromReader [@wbNameRB, Heap.systemZone];

    DeleteEachFile: NSFile.AttributesProc = {
      parent: NSFile.Handle = LOOPHOLE [clientData];
      NSFile.DeleteChild [parent, attributes.fileID<<, session>>
        ! NSFile.Error => CONTINUE]
      }; -- of DeleteEachFile
  <<identity: Auth.IdentityHandle ←
      Atom.GetProp [currentUser, identityHandle]↑.value;
    session: NSFile.Session ← NSFile.Logon [identity !
      NSFile.Error => CONTINUE];
    NSFileExtra.EnablePrivileges [session];>>

    {ENABLE UNWIND => NULL;
    <<{NSFile.Close [wbfh, session ! NSFile.Error => CONTINUE];
       NSFile.Logoff [session]};>>
    wbfh: NSFile.Handle ←
      NSFile.OpenByName [parentFh, nsWbname ! NSFile.Error => GOTO Exit];
    Process.SetPriority [Process.priorityBackground];
    NSFile.List [wbfh, DeleteEachFile,
      [interpreted: [fileID: TRUE]],,LOOPHOLE[wbfh]];
    NSFile.Close [wbfh<<, session>> ! NSFile.Error => CONTINUE];
    NSString.FreeString [Heap.systemZone, nsWbname];
  <<NSFile.Logoff [session];>>
    }; -- enable
    EXITS
     Exit => NULL;
    }; -- of Deleter

<<<<<<<<<<<<<<<<<<<<<<<< AgentProc >>>>>>>>>>>>>>>>>>>>>>>>

  logonCompleted: Atom.ATOM ← Atom.MakeAtom ["LogonCompleted"L];
  events: ARRAY [0..2) OF Event.EventType ← [
    logonCompleted, StarDesktop.logoff];

  AgentProc: Event.AgentProcedure = {
    ENABLE TIP.Error => CONTINUE;
    SELECT event FROM
      logonCompleted => {
        };
      StarDesktop.logoff => {
      --Process.Detach [FORK Deleter []]};
        Deleter []};
      ENDCASE;
    }; -- of AgentProc

<<<<<<<<<<<<<<<<<<<<<<<< Init >>>>>>>>>>>>>>>>>>>>>>>>

  currentUser, identityHandle: Atom.ATOM ← Atom.null;

  Init: PROC = {
    currentUser ← Atom.MakeAtom ["CurrentUser"L];
    identityHandle ← Atom.MakeAtom ["IdentityHandle"L];
    [] ← Event.AddDependencies [
      agent: AgentProc, myData: NIL, events: DESCRIPTOR [events]];
    }; -- of Init

<<<<<<<<<<<<<<<<<<<<<<<< Main >>>>>>>>>>>>>>>>>>>>>>>>

  Init [];

  }.
```

```
-- File: Applize.config - last edit:
-- Mita:OSBU South:Xerox 28-Jul-87 10:23:01
-- Lewis:OSBU South:Xerox 23-Jun-87 12:26:52
-- Camacho.ES          14-Jun-85 13:46:45
-- Breisacher.ES       28-Mar-85 12:39:49

-- Copyright (C) 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

Applize: CONFIGURATION LINKS:FRAME

IMPORTS Atom, Attention, Containee, Courier, Heap, MenuData, NSFile, NSFileExtra, NSString, OptionFile, Selection, SpecialDesktop,
StarDesktop, System, XFormat, XString, XTime

EXPORTS ApplizeFriends
CONTROL ApplizeImpl =
  BEGIN

  ApplizeImpl;

  END.
```

```
-- File: ApplizeFriends.mesa - last edit:
-- Lewis:OSBU South:Xerox  9-Jul-87 13:10:59

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.


DIRECTORY
  NSFile USING [Handle],
  System USING [gmtEpoch, GreenwichMeanTime],
  XString USING [nullReaderBody, ReaderBody];


ApplizeFriends: DEFINITIONS =
BEGIN


--==================== TYPES ==========================

Props: TYPE = LONG POINTER TO PropsRecord;
PropsRecord: TYPE = RECORD [
   version: XString.ReaderBody,
   invisible, autorun, ignoreVersionMismatch: BOOLEAN,
   links: Links,
   createdOn: System.GreenwichMeanTime] ;

Links: TYPE = {code, frame};


--==================== CONSTANTS ======================

nullPropsRecord: PropsRecord = [
   XString.nullReaderBody, FALSE, FALSE, FALSE, code, System.gmtEpoch];


--==================== EXCEPTIONS =====================

Error: SIGNAL [type: ErrorType];
ErrorType: TYPE = {invalidFileType, invalidProps, unknown};


--==================== OPERATIONS =====================

-- Change the folder into an application with the specified properties.  May signal Error.
FolderToAppl: PROC [file: NSFile.Handle, props: Props, neverBackup, notifyDesktop: BOOLEAN];

-- Change the given file from an application into a folder.  May signal Error.
ApplToFolder: PROC [file: NSFile.Handle, notifyDesktop: BOOLEAN];

-- Answer the props of application file.  Allocated from client's zone.  May signal Error.
GetProps: PROC [file: NSFile.Handle, zone: UNCOUNTED ZONE] RETURNS [props: Props];

END...

LOG
23-Jun-87 - Lewis.ES - Create.
09-Jul-87 - Lewis.ES - Add createdOn to Props, GetProps.
```

```
-- File: ApplizeImpl.mesa - last edit:
-- Mita:OSBU South:Xerox  27-Aug-87 10:29:07
-- Lewis:OSBU South:Xerox  9-Jul-87 14:13:39
-- Breisacher.ES       26-Sep-85 18:37:29
-- Camacho.ES          14-Jun-85 13:46:32

-- Copyright (C) 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

<<
This tool allows a user (presumably a developer/integrator) to change an ordinary folder into an Application and vice versa.

Major procedures:
o Applize - handles menu commands to turn selected folder(s) into applications - sets appl props to standard state (visible, version
"OS6")
o UnApplize - handles menu command to turn selected application(s) into folders - zaps type and (supposedly) version, but doesn't touch
any other appl props [odd.  Ed.]

This module exports utility operations to the ApplizeFriends interface.

All procs in this module are in alphabetic order except for 'Init', which is at the start of the module.
>>


DIRECTORY
  ApplizeFriends USING [ErrorType, Props, PropsRecord],  -- exported interface
  Atom USING [ATOM, MakeAtom, null],
  ApplicationFolderExtra2,
  Attention USING [AddMenuItem, Clear, formatHandle, Post],
  Containee USING [GenericProc, GetImplementation, Implementation, SetImplementation],
  BWSLoaderInternal,
  Courier USING [Error],
  Heap USING [Create],
  MenuData USING [CreateItem, MenuProc],
  NSFile,
  NSFileExtra USING [ChangeAttributesPrivileged],
  NSString USING [StringFromMesaString, FreeString, nullString, String],
  OptionFile,
  Selection USING [CanYouConvert, Convert, Enumerate, EnumerationProc, Free, Value],
  StarFileTypes USING [folder],
    -- plus: applicationFolder, invisibleApplicationFolder, profile
  SpecialDesktop USING [IconChanged],
  StarDesktop,
  System,
  XString USING [
    CopyToNewReaderBody, Empty, FreeReaderBytes, FromSTRING, FromNSString, nullReaderBody, NSStringFromReader, Reader, ReaderBody],
  XFormat USING [NSString, String],
  XTime USING [Current];

ApplizeImpl: PROGRAM
IMPORTS Atom, Attention, Containee, Courier, Heap, MenuData, NSFile, NSFileExtra,
        NSString, OptionFile, Selection, SpecialDesktop, StarDesktop, System, XFormat, XString, XTime
EXPORTS ApplizeFriends  -- Error, FolderToAppl, ApplToFolder, GetProps
=

BEGIN
OPEN MyIF: ApplizeFriends, BWSLI: BWSLoaderInternal;

  --================== TYPEs ==========================

  --

  Handle: TYPE = LONG POINTER TO Object;

  Object: TYPE = RECORD [
    internalName: XString.ReaderBody ← XString.nullReaderBody,
    fontFile: BOOLEAN ← FALSE,
    priority: CARDINAL ← 0];

  --================== Constants ==========================

  -- NS files types //COPIED FROM BWSLoaderInternal//
  applicationFolder: NSFile.Type = 4387;
  invisibleApplicationFolder: NSFile.Type = 4423;
  profile: NSFile.Type = 4385;  -- used for WS profiles and ADF files


  -- use the folder Open operation on applizations for Applize users
  openAtom: Atom.ATOM ← Atom.null;  -- set by Init
  oldApplicationGenericProc: Containee.GenericProc ← NIL;  -- set by Init
  folderGenericProc: Containee.GenericProc ← NIL;  -- set by Init


  localZ: UNCOUNTED ZONE = Heap.Create [initial:1];

  psData: Handle ← localZ.NEW[Object];

  true: BOOLEAN ← TRUE;
  false: BOOLEAN ← FALSE;

  --================== Data ==========================

  Error: PUBLIC SIGNAL [type: MyIF.ErrorType] = CODE;
```

```
--================== MODULE INITIALIZATION ==========================

Init: PROC =
  BEGIN

  -- register attention window menu commands to applize/deapplize
  BEGIN
  applize: XString.ReaderBody ← XString.FromSTRING["Folder => Application"L];
  applizeNeverBackup: XString.ReaderBody ← XString.FromSTRING["Folder => Application (neverBackup)"L];
  unapplize: XString.ReaderBody ← XString.FromSTRING["Application => Folder"L];
  upgrade: XString.ReaderBody ← XString.FromSTRING["UpDate Application Folder"L];
  Attention.AddMenuItem [
    MenuData.CreateItem [
      zone: NIL,
      name: @applize,
      proc: Applize,
      itemData: @false] ]; -- itemData is neverBackup: BOOLEAN ← FALSE
  Attention.AddMenuItem [
    MenuData.CreateItem [
      zone: NIL,
      name: @applizeNeverBackup,
      proc: Applize,
      itemData: @true] ]; -- itemData is neverBackup: BOOLEAN ← TRUE
  Attention.AddMenuItem [
    MenuData.CreateItem [
      zone: NIL,
      name: @unapplize,
      proc: UnApplize] ];
  IF System.switches['d] = down THEN
    Attention.AddMenuItem [
    MenuData.CreateItem [
      zone: NIL,
      name: @upgrade,
      proc: UpDate,
      itemData: @false] ]; -- itemData is neverBackup: BOOLEAN ← FALSE
  END;

  -- make appls behave like folders (e.g., can open them) but use old Props
  BEGIN
  applImpl: Containee.Implementation ← Containee.GetImplementation [applicationFolder];
  folderImpl: Containee.Implementation ←
      Containee.GetImplementation [StarFileTypes.folder];
  oldApplicationGenericProc ← applImpl.genericProc;
  folderGenericProc ← folderImpl.genericProc;
  applImpl.genericProc ← ApplGenericProc;
  [] ← Containee.SetImplementation [applicationFolder, applImpl];
  [] ← Containee.SetImplementation [invisibleApplicationFolder, applImpl];
  openAtom ← Atom.MakeAtom ["Open"L];
  END;

  END;  -- Init
```

```
--================== Procedures ==========================

-- application folder ops (override standard behavior with folder ops, except Props)
ApplGenericProc: Containee.GenericProc =
<<PROC [atom: Atom.ATOM, data: Containee.DataHandle, changeProc: Containee.ChangeProc ← NIL, changeProcData: LONG POINTER ← NIL]
RETURNS [LONG UNSPECIFIED]>>
  BEGIN
  IF atom = openAtom THEN
    RETURN [folderGenericProc[atom, data, changeProc, changeProcData]]
  ELSE
    RETURN [oldApplicationGenericProc[atom, data, changeProc, changeProcData]];
  END;  -- ApplGenericProc


-- applize command handler - enumerate selection and applize each folder
Applize: MenuData.MenuProc = {
<<PROC [window: Window.Handle, menu: MenuData.MenuHandle, itemData: LONG UNSPECIFIED]>>

  wrongSelectionType: XString.ReaderBody ← XString.FromSTRING ["The selection must be a Folder!"L];
  neverBackup: BOOLEAN = LOOPHOLE [itemData, LONG POINTER TO BOOLEAN]↑;

  EachFile: Selection.EnumerationProc =
  <<[element: Selection.Value, data: Selection.RequestorData] RETURNS [stop: BOOLEAN ← FALSE]>>
    BEGIN ENABLE
      UNWIND => Selection.Free[@element];

    file: LONG POINTER TO NSFile.Reference ← element.value;
    fh: NSFile.Handle ← NSFile.OpenByReference [file↑];
    fileType: NSFile.Type ← NSFile.GetType [fh];
    IF fileType # StarFileTypes.folder THEN {
      Attention.Post [@wrongSelectionType];
      NSFile.Close [fh];
      Selection.Free[@element];
      RETURN;
      };
    GetInfo[fh, FALSE];
    ApplizeFile [fh, neverBackup];
    PostMessage [fh, " Applicationized"L];
    NSFile.Close [fh];
    Selection.Free[@element];
    END;  -- EachFile

  SELECT TRUE FROM
    Selection.CanYouConvert [target: file, enumeration: FALSE] => {
      v: Selection.Value ← Selection.Convert[file];
      IF v.value = NIL THEN {Attention.Post [@wrongSelectionType];RETURN};
      [] ← EachFile[v, NIL];
      };
    Selection.CanYouConvert [target: file, enumeration: TRUE] =>
      [] ← Selection.Enumerate [EachFile, file, NIL];
    ENDCASE => Attention.Post [@wrongSelectionType];
  };  -- Applize


-- turn the given folder file into an application by changing its types and attrs
ApplizeFile: PROCEDURE [file: NSFile.Handle, neverBackup: BOOLEAN] = {
  attrList: ARRAY [0..7) OF NSFile.Attribute;
  version: XString.ReaderBody ← XString.FromSTRING ["VP ??"L];
  nssversion: NSString.String ← XString.NSStringFromReader [@version, localZ];
  nsinternalname: NSString.String ← XString.NSStringFromReader [@psData.internalName, localZ];
  attrList[0] ← [type[value: applicationFolder]];
  attrList[1] ← [createdOn[value: XTime.Current[]]];
  attrList[2] ← [extended[type: BWSLI.VersionAttribute,
    value: NSFile.EncodeString[nssversion]]];

  attrList[3] ← [extended[type: BWSLI.InternalNameAttribute,
    value: NSFile.EncodeString[nsinternalname]]];

  attrList[4] ← [extended[type: BWSLI.PriorityAttribute,
    value: NSFile.EncodeCardinal[psData.priority]]];
  attrList[5] ← [extended[type: BWSLI.FontsAttribute,
    value: NSFile.EncodeBoolean[psData.fontFile]]];
  attrList[6] ← [extended[type: BWSLI.VersionStampAttribute,
    value: NSFile.EncodeCardinal[ApplicationFolderExtra2.versionStamp]]];

  IF neverBackup THEN NeverBackupChildren [file];
  --ISSUE: Always sets createdOn, but only sets version if not previously set? [DJL 6/87]
  NSFileExtra.ChangeAttributesPrivileged [file: file,
    attributes: DESCRIPTOR[attrList]];
  NSString.FreeString [localZ, nssversion];
  NSString.FreeString [localZ, nsinternalname];
  NotifyIconChanged [file];
  };  -- ApplizeFile


-- change the given application back into a normal folder
ApplToFolder: PUBLIC PROC [file: NSFile.Handle, notifyDesktop: BOOLEAN] =
  BEGIN
  type: NSFile.Type ← NSFile.GetType[file];
  IF (type = applicationFolder OR type = invisibleApplicationFolder) THEN
    UnApplizeFile[file, notifyDesktop]  -- HACK - should clean up all the attr's, no?
  ELSE
    SIGNAL Error[invalidFileType];
  END;  -- ApplToFolder

FindOptionFile: PUBLIC PROC [directory: NSFile.Handle, remote: BOOLEAN]
```

```
    RETURNS [optionFile: NSFile.Reference] =
    BEGIN
    fh: NSFile.Handle ← NSFile.nullHandle;
    dest: NSFile.Handle ← NSFile.nullHandle;
    filters: ARRAY [0..2) OF NSFile.Filter ← [
      [matches[[name[NSString.StringFromMesaString["*.adf"L]]]]],
      [equal[[type[BWSLI.userProfile]]]]];

    -- Find the ADF (Application Description File) which is an "OptionFile" (aka UserProfile)
    -- Look in there for the names of all the bcds to be started.
    fh ← NSFile.Find[
      directory: directory, scope: [filter: [and[DESCRIPTOR [filters]]]] ! NSFile.Error => CONTINUE];
    IF remote THEN
    {-- Copy ADF from Remote
    desktopRef: NSFile.Reference ← StarDesktop.GetCurrentDesktopFile[];
    dest ← NSFile.OpenByReference[desktopRef];
    fh ← NSFile.Copy[file: fh, destination: dest!NSFile.Error=> CONTINUE];
    };
    IF fh # NSFile.nullHandle THEN {
      optionFile ← NSFile.GetReference[fh];
      NSFile.Close [fh]}
    ELSE optionFile ← NSFile.nullReference;
    IF dest# NSFile.nullHandle THEN {
      NSFile.Close [dest]};
    END;  -- of FindOptionFile

  FixBackupOnChildren: PROCEDURE [file: NSFile.Handle] = {
    attrList: ARRAY [0..1) OF NSFile.Attribute ← [[backedUpOn[value: NSFile.nullTime]]];
    EachChild: NSFile.AttributesProc = {
      child: NSFile.Handle;
      IF attributes.backedUpOn = NSFile.neverBackup THEN {
        child ← NSFile.OpenChild [file, attributes.fileID];
        NSFile.ChangeAttributes [file: child, attributes: DESCRIPTOR[attrList]];
        NSFile.Close [child]};
      };  -- EachChild
    NSFile.List [directory: file, proc: EachChild, selections: [
      interpreted: [fileID: TRUE, backedUpOn: TRUE]]];
    };  -- FixBackupOnChildren


  -- turn the given folder into an application with the specified props
  FolderToAppl: PUBLIC PROC [file: NSFile.Handle, props: MyIF.Props, neverBackup, notifyDesktop: BOOLEAN] =
    BEGIN

    fileType: NSFile.Type ← NSFile.GetType[file];
    nsVersion: NSString.String;
    defaultVersion: XString.ReaderBody ← XString.FromSTRING ["VP ???"L];
    nsinternalname: NSString.String;
    privAttrList: ARRAY [0..1) OF NSFile.Attribute;  -- createdOn attr is special
    attrList: ARRAY [0..9) OF NSFile.Attribute;

    IF fileType # StarFileTypes.folder THEN {SIGNAL Error[invalidFileType]; RETURN; };

    -- Gets InternalName, Priority, Fonts info. from adf.
    GetInfo[file, FALSE];

    -- set up the NSFile attributes from the given props
    nsinternalname ← XString.NSStringFromReader [@psData.internalName, localZ];
    privAttrList[0] ← [createdOn[value: props.createdOn]];
    attrList[0] ← [type[value:
        (IF props.invisible THEN invisibleApplicationFolder ELSE applicationFolder)]];
    nsVersion ← XString.NSStringFromReader [
        (IF props.version # XString.nullReaderBody THEN @props.version
         ELSE @defaultVersion), localZ];
    attrList[1] ← [extended[type: BWSLI.VersionAttribute,
        value: NSFile.EncodeString[nsVersion]]];
    attrList[2] ← [extended[type: BWSLI.RunAtStartupAttribute,
        value: NSFile.EncodeBoolean[props.autorun]]];
    attrList[3] ← [extended[type: BWSLI.LinksAttribute,
        value: NSFile.EncodeBoolean[(props.links = code)]]];
    attrList[4] ← [extended[type: BWSLI.VersionMismatchAttribute,
        value: NSFile.EncodeBoolean[props.ignoreVersionMismatch]]];

    attrList[5] ← [extended[type: BWSLI.PriorityAttribute,
      value: NSFile.EncodeCardinal[psData.priority]]];
    attrList[6] ← [extended[type: BWSLI.FontsAttribute,
      value: NSFile.EncodeBoolean[psData.fontFile]]];
    attrList[7] ← [extended[type: BWSLI.VersionStampAttribute,
      value: NSFile.EncodeCardinal[ApplicationFolderExtra2.versionStamp]]];
    attrList[8] ← [extended[type: BWSLI.InternalNameAttribute,
      value: NSFile.EncodeString[nsinternalname]]];

    IF neverBackup THEN NeverBackupChildren [file];
    -- gotta make two calls cause ChangeAttributesPrivileged only adds, doesn't change...
    NSFileExtra.ChangeAttributesPrivileged [file, DESCRIPTOR[privAttrList]];
    NSFile.ChangeAttributes [file, DESCRIPTOR[attrList]];

    -- clean up storage for extended attributes
    NSString.FreeString [localZ, nsVersion];
    NSString.FreeString [localZ, nsinternalname];
    NSFile.ClearAttributeList [DESCRIPTOR[attrList]];

    IF notifyDesktop THEN NotifyIconChanged [file];

    END;  -- FolderToAppl

  -- applize command handler - enumerate selection and applize each folder
```

```
UpDate: MenuData.MenuProc = {
<<PROC [window: Window.Handle, menu: MenuData.MenuHandle, itemData: LONG UNSPECIFIED]>>

  wrongSelectionType: XString.ReaderBody ← XString.FromSTRING ["The selection must be a Folder!"L];
  neverBackup: BOOLEAN = LOOPHOLE [itemData, LONG POINTER TO BOOLEAN]↑;

  EachFile: Selection.EnumerationProc =
  <<[element: Selection.Value, data: Selection.RequestorData] RETURNS [stop: BOOLEAN ← FALSE]>>
    BEGIN ENABLE
      UNWIND => Selection.Free[@element];

    file: LONG POINTER TO NSFile.Reference ← element.value;
    fh: NSFile.Handle ← NSFile.OpenByReference [file↑!Courier.Error, NSFile.Error=> GO TO exit];
    fileType: NSFile.Type ← NSFile.GetType [fh];
    IF fileType # applicationFolder AND fileType # invisibleApplicationFolder THEN {
      Attention.Post [@wrongSelectionType];
      NSFile.Close [fh];
      Selection.Free[@element];
      RETURN;
      };
    UpdateApplizeFile [fh];
    PostMessage [fh, " Applicationized"L];
    NSFile.Close [fh];
    Selection.Free[@element];
    EXITS exit => {s: LONG STRING ← "Remote Error"L;
      Attention.Clear[];
      XFormat.String [Attention.formatHandle, s];};
    END;  -- EachFile

  SELECT TRUE FROM
    Selection.CanYouConvert [target: file, enumeration: FALSE] => {
      v: Selection.Value ← Selection.Convert[file];
      IF v.value = NIL THEN {Attention.Post [@wrongSelectionType];RETURN};
      [] ← EachFile[v, NIL];
      };
    Selection.CanYouConvert [target: file, enumeration: TRUE] =>
      [] ← Selection.Enumerate [EachFile, file, NIL];
    ENDCASE => Attention.Post [@wrongSelectionType];
  };  -- Applize


-- turn the given folder file into an application by changing its types and attrs
UpdateApplizeFile: PROCEDURE [file: NSFile.Handle] = {
  attrList: ARRAY [0..5) OF NSFile.Attribute;
  attributeX: NSFile.AttributesRecord ← TRASH;
  remote: BOOLEAN ← FALSE;
  selections: NSFile.Selections ← [
    interpreted: [service: TRUE, type: TRUE, createdOn: TRUE]];
  nsinternalname: NSString.String;
  -- Gets attributes
  NSFile.GetAttributes[file, selections, @attributeX];
  remote ← attributeX.service.systemElement#NSFile.localSystemElement;
  GetInfo[file, remote];
  nsinternalname ← XString.NSStringFromReader [@psData.internalName, localZ];
  attrList[0] ← [createdOn[value: XTime.Current[]]];
  attrList[1] ← [extended[type: BWSLI.InternalNameAttribute,
    value: NSFile.EncodeString[nsinternalname]]];

  attrList[2] ← [extended[type: BWSLI.PriorityAttribute,
    value: NSFile.EncodeCardinal[psData.priority]]];
  attrList[3] ← [extended[type: BWSLI.FontsAttribute,
    value: NSFile.EncodeBoolean[psData.fontFile]]];
  attrList[4] ← [extended[type: BWSLI.VersionStampAttribute,
    value: NSFile.EncodeCardinal[ApplicationFolderExtra2.versionStamp]]];
  --ISSUE: Always sets createdOn, but only sets version if not previously set? [DJL 6/87]
  NSFile.ChangeAttributes [file: file,
    attributes: DESCRIPTOR[attrList]];
  NSFile.ClearAttributes [@attributeX];
  NSString.FreeString [localZ, nsinternalname];
  NotifyIconChanged [file];
  };  -- ApplizeFile


-- Get InternalName and Font, Priority from ADF..
GetInfo: PROC [fh: NSFile.Handle, remote: BOOLEAN] =
  BEGIN
  prioEntry: XString.ReaderBody ← XString.FromSTRING["Priority"L];
  fontEntry: XString.ReaderBody ← XString.FromSTRING["NovaFontFile"L];
  optionFile: NSFile.Reference;
  fileHandle: NSFile.Handle;

  GetFont: PROC [value: XString.Reader] = {
      psData.fontFile ← ~XString.Empty[value];
        };-- GetFont
  optionFile ← FindOptionFile[directory: fh, remote: remote];
  XString.FreeReaderBytes[@psData.internalName, localZ];
  psData.internalName ← GetInternalName[optionFile];
  psData.priority ← CARDINAL[OptionFile.GetIntegerValue [
     section: @psData.internalName, entry: @prioEntry, file: optionFile !
      OptionFile.Error => {psData.priority ← 0; CONTINUE}]];
  OptionFile.GetStringValue [
     section: @psData.internalName, entry: @fontEntry, callBack: GetFont, file:  optionFile !
      OptionFile.Error => {psData.fontFile ← FALSE;CONTINUE}];
  IF remote THEN {fileHandle ← NSFile.OpenByReference[optionFile!NSFile.Error, Courier.Error=>CONTINUE];
  NSFile.Delete[fileHandle!NSFile.Error, Courier.Error =>CONTINUE];
  NSFile.Close[fileHandle!NSFile.Error=>CONTINUE];};
  END;  -- of GetInfo
```

```
  -- Answer the props of application file.

  GetInternalName: PUBLIC PROC [optionFile: NSFile.Reference]
    RETURNS [internalName: XString.ReaderBody ← XString.nullReaderBody] =
    BEGIN
    CopyName: OptionFile.SectionEnumProc = {
      internalName ← XString.CopyToNewReaderBody[r: section, z: localZ];
      stop ← TRUE << Only one name to a customer!!! >> };

    OptionFile.EnumerateSections[callBack: CopyName, file: optionFile];
    END;  -- of GetInternalName

GetProps: PUBLIC PROC [file: NSFile.Handle, zone: UNCOUNTED ZONE]
  RETURNS [props: MyIF.Props] =
    BEGIN

    type: NSFile.Type ← NSFile.GetType[file];
    attributes: NSFile.AttributesRecord;
    rbVersion: XString.ReaderBody ← XString.nullReaderBody;
    nsVersion: NSString.String ← NSString.nullString;
    invisible, autorun, ignoreVersionMismatch, codeLinks: BOOLEAN ← FALSE;

    extendedSelections: ARRAY [0..4) OF NSFile.ExtendedAttributeType ← [
        BWSLI.VersionAttribute, BWSLI.RunAtStartupAttribute, BWSLI.LinksAttribute, BWSLI.VersionMismatchAttribute];
    selections: NSFile.Selections ← [
      interpreted: [createdOn: TRUE], extended: DESCRIPTOR[extendedSelections]];

    IF NOT (type = applicationFolder OR type = invisibleApplicationFolder) THEN
       {SIGNAL Error[invalidFileType]; RETURN [NIL]; };

    NSFile.GetAttributes [file, selections, @attributes];
    invisible ← (type = invisibleApplicationFolder);
    nsVersion ← NSFile.DecodeString [attributes.extended[0].value
        ! Courier.Error => CONTINUE];
    rbVersion ← XString.FromNSString [nsVersion];
    autorun ← NSFile.DecodeBoolean[attributes.extended[1].value
        ! Courier.Error => CONTINUE];
    codeLinks ← NSFile.DecodeBoolean[attributes.extended[2].value
        ! Courier.Error => CONTINUE];
    ignoreVersionMismatch ← NSFile.DecodeBoolean[attributes.extended[3].value
        ! Courier.Error => CONTINUE];

    props ← zone.NEW [MyIF.PropsRecord ← [
        version: XString.CopyToNewReaderBody [@rbVersion, zone],
        invisible: invisible,
        autorun: autorun,
        ignoreVersionMismatch: ignoreVersionMismatch,
        links: (IF codeLinks THEN code ELSE frame),
        createdOn: attributes.createdOn ]];
    NSFile.ClearAttributes [@attributes];
    RETURN [props];

    END;  -- GetProps


  NeverBackupChildren: PROCEDURE [file: NSFile.Handle] = {
    attrList: ARRAY [0..1) OF NSFile.Attribute ← [[backedUpOn[value: NSFile.neverBackup]]];
    EachChild: NSFile.AttributesProc = {
      child: NSFile.Handle ← NSFile.OpenChild [file, attributes.fileID];
      NSFile.ChangeAttributes [file: child, attributes: DESCRIPTOR[attrList]];
      NSFile.Close [child];
      };
    NSFile.List [directory: file, proc: EachChild, selections: [
      interpreted: [fileID: TRUE]]];
    };  -- NeverBackupChildren


  NotifyIconChanged: PROCEDURE [file: NSFile.Handle] = {
    ref: NSFile.Reference ← NSFile.GetReference[file];
    selections: NSFile.Selections ← [];
    selections.interpreted[type] ← TRUE;
    SpecialDesktop.IconChanged[ref, selections];
    };  -- NotifyIconChanged


  -- post the message "<filename> <s>"
  PostMessage: PROCEDURE [file: NSFile.Handle, s: LONG STRING] = {
    attributes: NSFile.AttributesRecord;
    attributeSelections: NSFile.Selections ← [interpreted: [name: TRUE]];
    NSFile.GetAttributes[file, attributeSelections, @attributes];
    Attention.Clear[];
    XFormat.NSString [Attention.formatHandle, attributes.name];
    XFormat.String [Attention.formatHandle, s];
    NSFile.ClearAttributes[@attributes];
    };  -- PostMessage


  -- un-applize command handler - enumerate selection and make each appl back into folder
  UnApplize: MenuData.MenuProc = {
  <<PROC [window: Window.Handle, menu: MenuData.MenuHandle, itemData: LONG UNSPECIFIED]>>

    wrongSelectionType: XString.ReaderBody ← XString.FromSTRING ["The selection must be an Application!"L];

    EachFile: Selection.EnumerationProc =
    <<[element: Selection.Value, data: Selection.RequestorData] RETURNS [stop: BOOLEAN ← FALSE]>>
      BEGIN ENABLE
```

```
        UNWIND => Selection.Free[@element];

      file: LONG POINTER TO NSFile.Reference ← element.value;
      fh: NSFile.Handle ← NSFile.OpenByReference [file↑];
      fileType: NSFile.Type ← NSFile.GetType [fh];
      IF fileType # applicationFolder AND fileType # invisibleApplicationFolder THEN {
        Attention.Post [@wrongSelectionType];
        NSFile.Close [fh];
        Selection.Free[@element];
        RETURN;
        };
      UnApplizeFile [fh, TRUE];
      PostMessage [fh, " UnApplicationized"L];
      NSFile.Close [fh];
      Selection.Free[@element];
      END;  -- EachFile

    SELECT TRUE FROM
      Selection.CanYouConvert [target: file, enumeration: FALSE] => {
        v: Selection.Value ← Selection.Convert[file];
        IF v.value = NIL THEN {Attention.Post [@wrongSelectionType];RETURN};
        [] ← EachFile[v, NIL];
        };
      Selection.CanYouConvert [target: file, enumeration: TRUE] =>
        [] ← Selection.Enumerate [EachFile, file, NIL];
      ENDCASE => Attention.Post [@wrongSelectionType];
    };  -- UnApplize

  -- turn the given appl file into a folder by changing its types and attrs
  UnApplizeFile: PROCEDURE [file: NSFile.Handle, notifyDesktop: BOOLEAN] = {
    attrList: ARRAY [0..2) OF NSFile.Attribute;
    attrList[0] ← [type[value: StarFileTypes.folder]];
    attrList[1] ← [extended[type: BWSLI.VersionAttribute, value: NIL]];
    FixBackupOnChildren [file];
    -- ISSUE: Doesn't seem to clear version string?! [DJL 6/87]
    NSFile.ChangeAttributes [file: file, attributes: DESCRIPTOR[attrList]];
    IF notifyDesktop THEN NotifyIconChanged [file];
    };  -- UnApplizeFile


  -- Main line code
  Init[];

  END.


LOG
14-Jun-85 - Camacho.ES - (create, no?)
26-Sep-85 - Breisacher.ES - (unspecified)
23-Jun-87 - Lewis.ES - Use folder generic ops for everything except Props; export new ApplizeFriends interface.
09-Jul-87 - Lewis - FolderToAppl uses props.createdOn, add GetProps.
13-Jul-87 - Mita.ES - Minor re-structuring: Support InternalName, Priority, Font information to ExtendedAttribute. also versionStamp to 2
for new applize.
15-Jul-87 - Mita.ES - Change font to Boolean, Fix Leak for psData.internalName, change OS6.0 to VP ??
28-Jul-87  - Mita.ES - Support UpDate for the developer.
28-Jul-87  - Mita.ES - call folder generic procs only for Open
```

```
-- File: BWSTIPTest.config - last edit:
-- Breisacher.ES       29-Mar-85 20:04:32
-- JGS         24-May-84 14:17:24

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

BWSTIPTest: CONFIGURATION LINKS: CODE
  IMPORTS
    Atom, Attention, Cursor, FormWindow, Heap, MenuData,
    MessageWindow, StarWindowShell, TIP, TIPStar, XFormat, XString
  CONTROL BWSTipTestImpl= BEGIN

  BWSTipTestImpl;

  END.
```

```
-- File: BWSTipTestImpl.mesa - last edit:
-- Breisacher.ES       29-Mar-85 20:00:55
-- Sandman.pa          29-Oct-84 11:55:02

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, GetPName],
  Attention USING [AddMenuItem, Post, RemoveMenuItem],
  Cursor USING [Set],
  FormWindow,
  Heap USING [systemZone],
  LevelIVKeys USING [KeyName],
  MenuData USING [CreateItem, ItemHandle, MenuProc],
  MessageWindow USING [Create, XFormatObject],
  StarWindowShell USING [
    Create, CreateBody, Handle, Push, SetPreferredDims, TransitionProc],
  TIP USING [
    ATOM, ClearInputFocusOnMatch, CreateTable, DestroyTable, InvalidTable,
    KeyToCharProc, LosingFocusProc, NotifyProc, Results, SetCharTranslator,
    SetInputFocus, SetTable, SetTableAndNotifyProc, Table],
  TIPStar USING [NormalTable],
  Window USING [Dims, Handle],
  XFormat USING [Char, CR, Handle, Object, Octal, Reader, String],
  XString USING [AppendChar, Character, FromSTRING, Reader, ReaderBody];

BWSTipTestImpl: MONITOR
  IMPORTS
    Atom, Attention, Cursor, FormWindow, Heap, MenuData,
    MessageWindow, StarWindowShell, TIP, TIPStar, XFormat, XString = BEGIN
  OPEN SWS: StarWindowShell, XS: XString, FW: FormWindow, MD: MenuData;

-- Types

    fObject: XFormat.Object;
    fHandle: XFormat.Handle ← @fObject;
    shell: SWS.Handle;
    bodyWindow: Window.Handle ← NIL;
    testWindow: Window.Handle ← NIL;
    table: TIP.Table ← NIL;
    menuItem: MD.ItemHandle ← NIL;

-- Constants

  shellDims: Window.Dims = [500, 600];
  bodyWindowDims: Window.Dims = [480, 600];
  testWindowDims: Window.Dims = [475, 300];

-- Data

  toolName: XS.ReaderBody ← XS.FromSTRING["BWS TIP Test"L];

-- Procedures

  Init: PROC = {
    menuItem ← MD.CreateItem[zone: Heap.systemZone, name: @toolName, proc: MenuProc];
    Attention.AddMenuItem[menuItem]};

  MenuProc: MD.MenuProc = {
    Attention.RemoveMenuItem[menuItem];
    shell ← SWS.Create [name: @toolName, transitionProc: Transition];
    bodyWindow ← SWS.CreateBody[
      sws: shell,
      box: [[0,0],bodyWindowDims]];
    FW.Create[window: bodyWindow, makeItemsProc: MakeItems, layoutProc: DoLayout];
    SWS.SetPreferredDims[shell, shellDims];
    SWS.Push [shell]};

  Transition: SWS.TransitionProc = {
    SELECT state FROM
      sleeping, dead => {
        IF table # NIL THEN TIP.DestroyTable[@table];
        Attention.AddMenuItem[menuItem]};
      ENDCASE};

  -- Main line code
  ItemIndex: TYPE = {create, name, destroy, takeInput, testWindow};

  MakeItems: FW.MakeItemsProc = {
    rb: XS.ReaderBody ← XS.FromSTRING["Create"L];
    FW.MakeCommandItem[
      window: window, myKey: ItemIndex.create.ORD, commandName: @rb,
      commandProc: CreateFile];
    rb ← XS.FromSTRING["File"L];
    FW.MakeTextItem[
      window: window, myKey: ItemIndex.name.ORD, tag: @rb, width: 250];
    rb ← XS.FromSTRING["Destroy"L];
    FW.MakeCommandItem[
      window: window, myKey: ItemIndex.destroy.ORD, commandName: @rb,
      commandProc: DestroyFile];
    rb ← XS.FromSTRING["Take Input"L];
    FW.MakeCommandItem[
      window: window, myKey: ItemIndex.takeInput.ORD, commandName: @rb,
      commandProc: TakeInput];
    testWindow ← FW.MakeWindowItem[
      window: window, myKey: ItemIndex.testWindow.ORD, size: testWindowDims];
    MessageWindow.Create [window: testWindow, lines: 20];
```

```
    fObject ← MessageWindow.XFormatObject[testWindow];
    TIP.SetTableAndNotifyProc[testWindow, NIL, TIPMe]};

  spaceAboveLine: CARDINAL = 15;

  DoLayout: FW.LayoutProc = {
    line: FW.Line ← FW.AppendLine[window, spaceAboveLine];
    FW.SetTabStops[window: window, tabStops: [fixed[100]]];
    FW.AppendItem[window, ItemIndex.create.ORD, line];
    FW.AppendItem[window, ItemIndex.name.ORD, line];
    line ← FW.AppendLine[window, spaceAboveLine];
    FW.AppendItem[window, ItemIndex.destroy.ORD, line];
    FW.AppendItem[window, ItemIndex.takeInput.ORD, line];
    line ← FW.AppendLine[window, spaceAboveLine];
    FW.AppendItem[window, ItemIndex.testWindow.ORD, line];
    };

  CreateFile: FW.CommandProc = BEGIN
    rb: XS.ReaderBody ← FW.LookAtTextItemValue[window, ItemIndex.name.ORD];
    Cursor.Set[hourGlass];
    TIP.ClearInputFocusOnMatch[testWindow];
    IF table # NIL THEN TIP.DestroyTable[@table];
    table ← TIP.CreateTable[file: @rb ! TIP.InvalidTable => {
      TipProblem[message]; table ← NIL; CONTINUE}];
    FW.DoneLookingAtTextItemValue[window, ItemIndex.name.ORD];
    IF table # NIL THEN {
      [] ← TIP.SetTable[testWindow, table];
      TIP.SetInputFocus[testWindow, TRUE, LoseInputFocus];
      [] ← TIP.SetCharTranslator[table, [MakeChar, NIL]]};
    Cursor.Set[textPointer];
    RETURN;
    END;

  LoseInputFocus: TIP.LosingFocusProc = {
    [] ← TIP.SetTable[testWindow, TIPStar.NormalTable[]]};

  TipProblem: PROC [r: XS.Reader] = {
    rb: XS.ReaderBody ← XS.FromSTRING["TIP Error: "L];
    Attention.Post[@rb];
    Attention.Post[s: r, clear: FALSE]};

  DestroyFile: FW.CommandProc = BEGIN
    TIP.ClearInputFocusOnMatch[testWindow];
    IF table # NIL THEN TIP.DestroyTable[@table];
    END;

  TakeInput: FW.CommandProc = BEGIN
    IF table # NIL THEN TIP.SetInputFocus[testWindow, TRUE, LoseInputFocus];
    END;

  TIPMe: TIP.NotifyProc = BEGIN
    first: BOOLEAN ← TRUE;
    IF table = NIL THEN RETURN;
    FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
      IF first THEN first ← FALSE
      ELSE fHandle.String[", "L];
      WITH z: input SELECT FROM
        atom =>{
          fHandle.String["Atom: "L];
          fHandle.Reader[Atom.GetPName[z.a]]};
        coords => {
          fHandle.String["Coords: ["L];
          fHandle.Octal[z.place.x];
          fHandle.String[", "L];
          fHandle.Octal[z.place.y];
          fHandle.String["]"L]};
        int => {
          fHandle.String["Integer: "L];
          fHandle.Octal[z.i]};
        key => {
          fHandle.String["Key: "L];
          fHandle.Octal[z.key.ORD];
          fHandle.String[", "L];
          fHandle.String[IF z.downUp = down THEN "down"L ELSE "up"L]};
        string => {
          fHandle.String["String: '"L];
          fHandle.Reader[@z.rb];
          fHandle.Char[''.ORD]};
        time => {
          fHandle.String["Time: "L];
          fHandle.Octal[z.time]};
        bufferedChar =>  fHandle.String["BufferedChar: Bug"L];
        nop =>  fHandle.String["Nop: Bug"L];
       ENDCASE;
     ENDLOOP;
   fHandle.CR[];
   RETURN;
   END;

  HasAscii: TYPE = LevelIVKeys.KeyName [Five..Equal];
  KeysDescriptionTable: TYPE = ARRAY HasAscii OF KeyDescription;

  KeyDescription: TYPE = RECORD [
    useLock: BOOLEAN ← FALSE,
    shiftCode: KeyCode ← noSuchKeyCode,
    normalCode: KeyCode,
    hasAscii: BOOLEAN ← TRUE];
```

```
    KeyCode: TYPE = CHARACTER [0C..255C];

    noSuchKeyCode: KeyCode = KeyCode.FIRST;
    noSuchCharacter: CHARACTER = CHARACTER.LAST;

    Shifted: TYPE = {yes, no, dontCare};
    keysDescriptionTable: KeysDescriptionTable = [
      [normalCode: '5, shiftCode: '%],
      [normalCode: '4, shiftCode: 244C],
      [normalCode: '6, shiftCode: '~],
      [normalCode: 'e, shiftCode: 'E, useLock: TRUE],
      [normalCode: '7, shiftCode: '&],
      [normalCode: 'd, shiftCode: 'D, useLock: TRUE],
      [normalCode: 'u, shiftCode: 'U, useLock: TRUE],
      [normalCode: 'v, shiftCode: 'V, useLock: TRUE],
      [normalCode: '0, shiftCode: ')],
      [normalCode: 'k, shiftCode: 'K, useLock: TRUE],
      [normalCode: '-, shiftCode: 137C],  -- underbar
      [normalCode: 'p, shiftCode: 'P, useLock: TRUE],
      [normalCode: '/, shiftCode: '?],
      [normalCode: '\\, shiftCode: '|],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- PASTE
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- BS

      [normalCode: '3, shiftCode: '#],
      [normalCode: '2, shiftCode: '@],
      [normalCode: 'w, shiftCode: 'W, useLock: TRUE],
      [normalCode: 'q, shiftCode: 'Q, useLock: TRUE],
      [normalCode: 's, shiftCode: 'S, useLock: TRUE],
      [normalCode: 'a, shiftCode: 'A, useLock: TRUE],
      [normalCode: '9, shiftCode: '(],
      [normalCode: 'i, shiftCode: 'I, useLock: TRUE],
      [normalCode: 'x, shiftCode: 'X, useLock: TRUE],
      [normalCode: 'o, shiftCode: 'O, useLock: TRUE],
      [normalCode: 'l, shiftCode: 'L, useLock: TRUE],
      [normalCode: ',, shiftCode: '<],
      [normalCode: '', shiftCode: '"],
      [normalCode: '], shiftCode: '}],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],
      -- STUFF
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- COMMAND

      [normalCode: '1, shiftCode: '!],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],
      -- COMPLETE
      [normalCode: 11C, shiftCode: 11C],  -- TAB
      [normalCode: 'f, shiftCode: 'F, useLock: TRUE],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- CONTROL
      [normalCode: 'c, shiftCode: 'C, useLock: TRUE],
      [normalCode: 'j, shiftCode: 'J, useLock: TRUE],
      [normalCode: 'b, shiftCode: 'B, useLock: TRUE],
      [normalCode: 'z, shiftCode: 'Z, useLock: TRUE],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- LeftShift
      [normalCode: '., shiftCode: '>],
      [normalCode: ';, shiftCode: ':],
      [normalCode: 15C, shiftCode: 15C],  -- Return
      [normalCode: 254C, shiftCode: 255C],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],
      -- DELETE
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- NEXT

      [normalCode: 'r, shiftCode: 'R, useLock: TRUE],
      [normalCode: 't, shiftCode: 'T, useLock: TRUE],
      [normalCode: 'g, shiftCode: 'G, useLock: TRUE],
      [normalCode: 'y, shiftCode: 'Y, useLock: TRUE],
      [normalCode: 'h, shiftCode: 'H, useLock: TRUE],
      [normalCode: '8, shiftCode: '*],
      [normalCode: 'n, shiftCode: 'N, useLock: TRUE],
      [normalCode: 'm, shiftCode: 'M, useLock: TRUE],
      [normalCode: noSuchKeyCode, hasAscii: FALSE],  -- LOCK
      [normalCode: ' , shiftCode: ' ],  -- Space
      [normalCode: '[, shiftCode: '{],
      [normalCode: '=, shiftCode: '+]];

  MakeChar: TIP.KeyToCharProc =
    BEGIN
    entry: KeyDescription;
    char: XString.Character;
    SELECT key FROM
      IN HasAscii => {
        entry ← keysDescriptionTable[key];
        IF ~entry.hasAscii THEN RETURN
        ELSE {
          char ←
            IF keys[LeftShift] = down OR keys[RightShift] = down
            OR keys[Key47] = down OR keys[A12] = down
            OR (entry.useLock AND keys[Lock] = down) THEN entry.shiftCode.ORD
            ELSE entry.normalCode.ORD};
      A8, A9, A11 => char ← ' .ORD;
      ENDCASE => RETURN;
      XString.AppendChar[to: buffer, c: char]
    END;


  Init[];

  END.
```

```
-- File: CatalogDivider.config - last edit:
-- Holbrook.ES          8-May-85 15:51:36
-- Breisacher.ES       27-Mar-85 15:17:50

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

CatalogDivider: CONFIGURATION LINKS: CODE
  IMPORTS
    Atom, Catalog, Containee, Directory, Divider, FileContainerShell,
    FolderColumns, Heap, NSFile, StarWindowShell, XString
  CONTROL CatalogDividerImpl= BEGIN

  CatalogDividerImpl;

  END.
```

```
-- File: CatalogDividerImpl.mesa - last edit:
-- Holbrook.ES          8-May-85 15:50:26
-- Breisacher.ES       28-Mar-85 12:43:15
-- Mader.ES             1-Feb-85 14:34:28

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  BWSFileTypes USING [desktop, desktopCatalog, prototypeCatalog, systemFileCatalog],
  Catalog USING [CatalogProc, Enumerate, Open],
  Containee USING [DataHandle, GetImplementation, Implementation, SetImplementation,
    SmallPictureProc],
  Directory USING [AddDividerEntry],
  DirectoryFileTypes USING [catalogs],
  Divider USING [AddEntry, ConvertProc, Create, GenericProc, DividerConvertProc,
    DividerGenericProc, Handle],
  FileContainerShell USING [Create],
  FolderColumns USING [ContentSeq, FreeColumnContents, FreeColumnHeaders, HeaderSeq,
    MakeColumnContents, MakeColumnHeaders],
  Heap USING [Create],
  NSFile USING [Close, GetReference, Handle, nullReference, Reference, Type],
  StarFileTypes,
  StarWindowShell USING [Handle, SetName, SetNamePicture],
  XString USING [CopyToNewReaderBody, FromSTRING, ReaderBody];

CatalogDividerImpl: PROGRAM
  IMPORTS Atom, Catalog, Containee, Directory, Divider, FileContainerShell, FolderColumns, Heap, NSFile, StarWindowShell, XString =
BEGIN

  -- TYPEs

  CatalogTableEntry: TYPE = RECORD [
    reference: NSFile.Reference ← NSFile.nullReference,
    name: XString.ReaderBody,
    type: NSFile.Type];

  CatalogTableSeq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF  CatalogTableEntry];

  -- Constants and data

  open, props: Atom.ATOM ← Atom.null;
  folder: NSFile.Type = StarFileTypes.folder;
  folderImpl, desktopImpl: LONG POINTER TO Containee.Implementation ← NIL;

  zone: UNCOUNTED ZONE = Heap.Create [initial:1];

  catalogTable: LONG POINTER TO CatalogTableSeq ← NIL;
  catalogCt: CARDINAL = 6;

  catalogsDivider: Divider.Handle ← NIL;

  catalogs: StarFileTypes.FileType = DirectoryFileTypes.catalogs;

  desktopCatalog: StarFileTypes.FileType = BWSFileTypes.desktopCatalog;
  helpCatalog: StarFileTypes.FileType = StarFileTypes.helpCatalog;
  prototypeCatalog: StarFileTypes.FileType = BWSFileTypes.prototypeCatalog;
  scratchDocCatalog: StarFileTypes.FileType = StarFileTypes.scratchDocCatalog;
  systemFileCatalog: StarFileTypes.FileType = BWSFileTypes.systemFileCatalog;
  tempFileCatalog: StarFileTypes.FileType = StarFileTypes.tempFileCatalog;

  -- Private Procedures


  BuildCatalogsDivider: Catalog.CatalogProc =
  BEGIN
  catalog: NSFile.Handle ← Catalog.Open [catalogType];

    FOR i: CARDINAL IN [0..catalogCt) DO
     IF catalogTable[i].type = catalogType THEN
     BEGIN
       catalogTable[i].reference ← NSFile.GetReference [catalog];
       Divider.AddEntry [catalogsDivider, folder, @catalogTable[i].name, @catalogTable[i].reference, , CatalogGenericProc];
       EXIT;
     END;
    ENDLOOP;
    NSFile.Close [catalog];
  END;

  CatalogGenericProc: Divider.GenericProc =
  BEGIN
    SELECT atom FROM
    open => RETURN [MakeCatalog [data]];
    props => RETURN [LONG [NIL]];
    ENDCASE => RETURN [folderImpl.genericProc [atom, data]];
  END;

  DesktopSmallPictureProc: Containee.SmallPictureProc =
  BEGIN
    RETURN [folderImpl.smallPictureProc [data, folder, normalOrReference]];
  END;

  MakeCatalog: PROCEDURE [data: Containee.DataHandle]
    RETURNS [shell: StarWindowShell.Handle] =
  BEGIN
  headers: LONG POINTER TO FolderColumns.HeaderSeq ← FolderColumns.MakeColumnHeaders[];
  contents: LONG POINTER TO FolderColumns.ContentSeq ← FolderColumns.MakeColumnContents[];
```

```
    shell ← FileContainerShell.Create [
      file: data.reference,
      columnHeaders: DESCRIPTOR[headers],
      columnContents: DESCRIPTOR[contents],
      scope: [ordering: [key[key: name, ascending: TRUE]]]];

    FolderColumns.FreeColumnHeaders [headers];
    FolderColumns.FreeColumnContents [contents];
    FOR i: CARDINAL IN [0..catalogCt) DO
      IF catalogTable[i].reference = data.reference THEN
      BEGIN
        StarWindowShell.SetName [shell, @catalogTable[i].name];
        EXIT;
      END;
    ENDLOOP;
    StarWindowShell.SetNamePicture [shell, folderImpl.smallPictureProc [data, folder, normal]];
  END;

  Init: PROCEDURE = {
    stringCatalogs: XString.ReaderBody ← XString.FromSTRING ["Catalogs"L];
    open ← Atom.MakeAtom["Open"];
    props ← Atom.MakeAtom["Props"L];

    catalogTable ← zone.NEW [CatalogTableSeq [catalogCt]];
    catalogTable[0] ←
      [name: XString.FromSTRING ["Desktop Catalog"L], type: desktopCatalog];
    catalogTable[1] ←
      [name: XString.FromSTRING ["Help Catalog"L], type: helpCatalog];
    catalogTable[2] ←
      [name: XString.FromSTRING ["Prototype Catalog"L], type: prototypeCatalog];
    catalogTable[3] ←
      [name: XString.FromSTRING ["Scratch Doc Catalog"L], type: scratchDocCatalog];
    catalogTable[4] ←
      [name: XString.FromSTRING ["System File Catalog"L], type: systemFileCatalog];
    catalogTable[5] ←
      [name: XString.FromSTRING ["Temp File Catalog"L], type: tempFileCatalog];
    FOR i: CARDINAL IN [0..catalogCt) DO
      catalogTable[i].name ← XString.CopyToNewReaderBody [
        @catalogTable[i].name, zone];
      ENDLOOP;

    folderImpl ← zone.NEW [Containee.Implementation];
    desktopImpl ← zone.NEW [Containee.Implementation];
    folderImpl↑ ← Containee.GetImplementation [folder];
    desktopImpl↑ ← folderImpl↑;

    desktopImpl.smallPictureProc ← DesktopSmallPictureProc;
    [] ← Containee.SetImplementation [BWSFileTypes.desktop, desktopImpl↑];

    catalogsDivider ← Divider.Create [catalogs, @stringCatalogs];
    Catalog.Enumerate [BuildCatalogsDivider];

    Directory.AddDividerEntry [ws, catalogs, @stringCatalogs, catalogsDivider, Divider.DividerConvertProc, Divider.DividerGenericProc];
    };

  -- Mainline code

  Init[];

END.
```

```
-- File: DevelopmentTools.cm - Last edited:
-- guzik.ES            28-May-87  9:34:27
-- Breisacher           5-Jan-87 10:11:51
-- Holbrook.ES          1-Mar-85  8:47:59
-- JGS          24-May-84 14:15:10

-- Copyright (C) 1985, 1987 by Xerox Corporation. All rights reserved.

CheckOutLibject recompilation/r DevelopmentTools.df/s

BringOver /a DevelopmentTools.df

-- Compiles

Compile /-bj-ns-u CatalogDividerImpl SystemFolderImpl.mesa  ApplizeImpl.mesa OpenAsFolderImpl.mesa

-- Bind

Binder /c-s SystemFolder CatalogDivider Applize OpenAsFolder BWSTIPTest

SModel /a DevelopmentTools.df

VerifyDF DevelopmentTools.df

-- SModel WorkingDFLoc/c "[UCR:OSBU South:Xerox]<Fnx>DF>" IntegrationLoc/c "[UCSF:OSBU South:Xerox]<IBWS>4.3>" /za DevelopmentTools.df
```

```
-- File: OpenAsFolder.config - last edit:
-- Holbrook.ES          8-May-85 15:52:32
-- Breisacher.ES       28-Mar-85 12:42:24

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

OpenAsFolder: CONFIGURATION LINKS: CODE
  IMPORTS  Atom, Attention, Containee, Courier, FileContainerShell, FolderColumns, Heap, MenuData, NSFile, Selection, StarWindowShell,
  TIP, UserTerminal, XString
  CONTROL OpenAsFolderImpl = BEGIN

  OpenAsFolderImpl;

  END.
```

```
-- File: OpenAsFolderImpl.mesa - last edit:
-- Breisacher          5-Jan-87 10:10:25
-- Holbrook.ES         8-May-85 15:52:19
-- JGS        24-May-84 14:13:55

-- Copyright (C) 1984, 1985, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Attention USING [AddMenuItem],
  Containee USING [DataHandle, GenericProc, GetCachedType, GetImplementation, Implementation, SetImplementation],
  Courier USING [Error],
  FileContainerShell USING [Create],
  FolderColumns USING [ContentSeq, FreeColumnContents, FreeColumnHeaders, HeaderSeq, MakeColumnContents, MakeColumnHeaders],
  Heap USING [systemZone],
  MenuData USING [CreateItem, MenuProc],
  NSFile USING [AttributesRecord, Close, Error, GetAttributes, Handle, nullHandle, OpenByReference, Reference, Type],
  Selection USING [Convert, Free, Value],
  StarWindowShell USING [Handle, SetPreferredDims],
  TIP USING [ATOM, GetNotifyProc, NotifyProc, ResultObject],
  UserTerminal USING [BlinkDisplay],
  XString USING [FromSTRING, ReaderBody];

OpenAsFolderImpl: PROGRAM
  IMPORTS
    Atom, Attention, Containee, Courier, FileContainerShell, FolderColumns, Heap, MenuData, NSFile, Selection, StarWindowShell, TIP,
    UserTerminal, XString = BEGIN

-- Types

  Columns: TYPE = {icon, name, size, createDate};

-- Data

   oldImpl: LONG POINTER TO Containee.Implementation ← NIL;

   Open, OpenDown: Atom.ATOM ← Atom.null;

   zone: UNCOUNTED ZONE ← Heap.systemZone;

-- Procedures

  Init: PROC = {
    toolName: XString.ReaderBody ← XString.FromSTRING["Open As Folder"L];
    OpenDown ← Atom.MakeAtom["OpenDown"L];
    Open ← Atom.MakeAtom["Open"L];
    oldImpl ← zone.NEW [Containee.Implementation ← []];
    Attention.AddMenuItem[MenuData.CreateItem[
      zone: Heap.systemZone, name: @toolName, proc: OpenSelection]]};

  GenericProc: Containee.GenericProc = BEGIN
    sz: StarWindowShell.Handle ← [NIL];
    SELECT atom FROM
    Open =>
     BEGIN
     sz ← CreateFolderSWS [ reference: data.reference ];
     -- may return nil .. just pass it on --
      RETURN [sz];
      END;
    ENDCASE => RETURN [ oldImpl.genericProc [atom, data] ];
    END;

  CreateFolderSWS: PROCEDURE [reference: NSFile.Reference]
    RETURNS [StarWindowShell.Handle] = {
    shell: StarWindowShell.Handle;
    headers: LONG POINTER TO FolderColumns.HeaderSeq ← FolderColumns.MakeColumnHeaders[];
    contents: LONG POINTER TO FolderColumns.ContentSeq ← FolderColumns.MakeColumnContents[];
    IF NOT IsDirectory [reference] THEN RETURN [ [NIL] ];
    shell ← FileContainerShell.Create[
      file: reference,
      columnHeaders: DESCRIPTOR[headers],
      columnContents: DESCRIPTOR[contents]];
    FolderColumns.FreeColumnHeaders [headers];
    FolderColumns.FreeColumnContents [contents];
    StarWindowShell.SetPreferredDims [ shell, [700, 0] ];
    RETURN[shell]};

  IsDirectory: PROCEDURE [reference: NSFile.Reference]
    RETURNS [yes: BOOLEAN] = {
    attributes: NSFile.AttributesRecord;
    file: NSFile.Handle ← NSFile.nullHandle;
    file ← NSFile.OpenByReference [reference
      ! NSFile.Error, Courier.Error => CONTINUE];
    IF file = NSFile.nullHandle THEN RETURN [yes: FALSE];
    NSFile.GetAttributes [file: file, selections: [[isDirectory: TRUE]], attributes: @attributes];
    NSFile.Close [file];
    RETURN[attributes.isDirectory];
    };

   OpenSelection: MenuData.MenuProc = BEGIN
     type: NSFile.Type;
     new: Containee.Implementation;
     proc: TIP.NotifyProc;
     results: TIP.ResultObject ← [ body: atom[a: OpenDown], next: NIL ];
     file: Selection.Value ← Selection.Convert [target: file, zone: zone];
     windowValue: Selection.Value;
```

```
    IF file.value=NIL THEN {UserTerminal.BlinkDisplay[]; RETURN};
    windowValue ← Selection.Convert[target: window, zone: zone];
    IF windowValue.value = NIL THEN {
      Selection.Free[@file];
      UserTerminal.BlinkDisplay[];
      RETURN};

    type ← Containee.GetCachedType [LOOPHOLE[file.value,Containee.DataHandle]];
    Selection.Free[@file];
    oldImpl↑ ← new ← Containee.GetImplementation [ type ];
    new.genericProc ← GenericProc;
    [] ← Containee.SetImplementation [ type, new ];
    proc ← TIP.GetNotifyProc [LOOPHOLE[windowValue.value]];
    proc [LOOPHOLE[windowValue.value], @results];
    Selection.Free[@windowValue];
    [] ← Containee.SetImplementation [ type, oldImpl↑ ];
    END;

  -- Main line code

  Init[];

  END.
```

Edit Log

5-Jan-87 - LFB - IsDirectory return value was unitialized, but was getting tested after the NSFile.OpenByReference. I changed it to check NSFile.nullHandle instead.

```
-- File: SystemFolder.config - last edit:
-- Holbrook.ES          7-May-85 16:13:52
-- Breisacher.ES       27-Mar-85 15:17:40
-- JGS        24-May-84 14:17:24

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

SystemFolder: CONFIGURATION LINKS: CODE
  IMPORTS
    Attention, Catalog, FileContainerShell, FolderColumns, FormWindow, Heap,
    MenuData, NSFile, NSString, PropertySheet, StarWindowShell, XString
  CONTROL SystemFolderImpl= BEGIN

  SystemFolderImpl;

  END.
```

```
-- File: SystemFolderImpl.mesa - last edit:
-- Holbrook.ES          7-May-85 16:12:41
-- Breisacher.es       10-Apr-85 17:48:05
-- JGS        24-May-84 14:13:55

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Attention USING [AddMenuItem],
  BWSFileTypes USING [prototypeCatalog, systemFileCatalog],
  Catalog USING [Open],
  FileContainerShell USING [Create],
  FolderColumns,
  FormWindow,
  Heap USING [Create],
  MenuData USING [CreateItem, CreateMenu, ItemHandle, MenuHandle, MenuProc],
  NSAssignedTypes USING [tDirectory],
  NSFile USING [Close, Filter, Find, GetReference, Handle, nullFilter, Reference],
  NSString,
  PropertySheet,
  StarWindowShell USING [Handle, Push, SetPreferredDims, SetRegularCommands, StandardClose],
  Window,
  XString;

SystemFolderImpl: PROGRAM
  IMPORTS
    Attention, Catalog, FileContainerShell, FolderColumns, FormWindow,
    Heap, MenuData, NSFile, NSString, PropertySheet, StarWindowShell, XString = BEGIN
  OPEN SWS: StarWindowShell, XS: XString;

  zone: UNCOUNTED ZONE = Heap.Create [initial:1];

-- Procedures

  Init: PROC = {
    sf: XS.ReaderBody ← XS.FromSTRING["System Folder"L];
    pf: XS.ReaderBody ← XS.FromSTRING["Prototype Folder"L];
    filter: XS.ReaderBody ← XS.FromSTRING["Set System Folder Filter"L];
    Attention.AddMenuItem[MenuData.CreateItem[
      zone: NIL, name: @sf, proc: SFMenuProc]];
    Attention.AddMenuItem[MenuData.CreateItem[
      zone: NIL, name: @filter, proc: SFFilterMenuProc]];
    Attention.AddMenuItem[MenuData.CreateItem[
      zone: NIL, name: @pf, proc: PFMenuProc]];
    };

  SFMenuProc: MenuData.MenuProc = {
    shell: StarWindowShell.Handle;
    headers: LONG POINTER TO FolderColumns.HeaderSeq ← FolderColumns.MakeColumnHeaders[];
    contents: LONG POINTER TO FolderColumns.ContentSeq ← FolderColumns.MakeColumnContents[];
    returnToFilter: XString.ReaderBody ← XString.FromSTRING["Change Filter"L];
    items: ARRAY [0..1) OF MenuData.ItemHandle ← [
      MenuData.CreateItem [zone: NIL, name: @returnToFilter, proc: ReturnToFilter]];
    myMenu: MenuData.MenuHandle = MenuData.CreateMenu [
      zone: NIL, title: NIL, array: DESCRIPTOR [items]];
    shell ← FileContainerShell.Create[
      file: NSFile.GetReference [
        Catalog.Open [BWSFileTypes.systemFileCatalog] ],
      columnHeaders: DESCRIPTOR[headers],
      columnContents: DESCRIPTOR[contents],
      scope: [filter: sfFilter, ordering: [key[key: name, ascending: TRUE]] ]];
    FolderColumns.FreeColumnHeaders [headers];
    FolderColumns.FreeColumnContents [contents];
    StarWindowShell.SetRegularCommands [sws: shell, commands: myMenu];
    StarWindowShell.SetPreferredDims [ shell, [700, 0] ];
    SWS.Push[shell]};


  PFMenuProc: MenuData.MenuProc = {
    shell: StarWindowShell.Handle;
    headers: LONG POINTER TO FolderColumns.HeaderSeq ← FolderColumns.MakeColumnHeaders[];
    contents: LONG POINTER TO FolderColumns.ContentSeq ← FolderColumns.MakeColumnContents[];
    shell ← FileContainerShell.Create[
      file: OpenPrototypeFolder[],
      columnHeaders: DESCRIPTOR[headers],
      columnContents: DESCRIPTOR[contents],
      scope: [ordering: [key[key: name, ascending: TRUE]] ],
      options: [readOnly: TRUE]]; -- make prototype folder readonly
    FolderColumns.FreeColumnHeaders [headers];
    FolderColumns.FreeColumnContents [contents];
    StarWindowShell.SetPreferredDims [ shell, [700, 0] ];
    SWS.Push[shell]};

  OpenPrototypeFolder: PROCEDURE RETURNS [ref: NSFile.Reference] = {
    catalog: NSFile.Handle ← Catalog.Open [BWSFileTypes.prototypeCatalog];
    filter: NSFile.Filter = [equal[[type[NSAssignedTypes.tDirectory]]]];
    folder: NSFile.Handle ← NSFile.Find [directory: catalog, scope: [filter: filter]];
    ref ← NSFile.GetReference [folder];
    NSFile.Close [folder];
    NSFile.Close [catalog];
    };

  ReturnToFilter: MenuData.MenuProc = {
    [] ← StarWindowShell.StandardClose [[window]];
    SFFilterMenuProc[NIL, NIL, LONG[0]];
    };
```

```
  -- Filter psheet stuff

  sfFilter: NSFile.Filter ← NSFile.nullFilter;
  nsStringFilter: NSString.String ← NSString.nullString;

  SFFilterMenuProc: MenuData.MenuProc = {
    -- create a psheet whose MenuItemProc sets the global sfFilter
    filter: XString.ReaderBody ← XString.FromSTRING["System Folder Filter"L];
    [] ← PropertySheet.Create [
      formWindowItems: MakeItems,
      menuItemProc: MenuItemProc,
      size: [400,200],
      placeToDisplay: [200,200],
      menuItems: [done: TRUE, cancel: TRUE, defaults: TRUE],
      title: @filter,
      display: TRUE];
    };

  Items: TYPE = {filter};

  MakeItems: FormWindow.MakeItemsProc = {
    rb: XString.ReaderBody ← XString.FromSTRING["Name Filter"L];
    init: XString.ReaderBody ← XString.FromNSString [nsStringFilter];
    FormWindow.MakeTextItem [
      window: window,
      myKey: Items.filter.ORD,
      tag: @rb,
      width: 200,
      initString: @init ];
    };

  MenuItemProc: PropertySheet.MenuItemProc = {
    SELECT menuItem FROM
      done => {
        ok ← ApplyAnyChanges[formWindow];
        IF ok THEN SFMenuProc [NIL, NIL, LONG[0]];
        RETURN};
      cancel => RETURN[ok: TRUE];
      defaults => {SetDefaults[formWindow];RETURN[ok: FALSE]};
      ENDCASE;
    RETURN[ok: FALSE];
    };

  ApplyAnyChanges: PROC [fw: Window.Handle] RETURNS [ok: BOOLEAN ← TRUE] = {
    FOR myItem: Items IN Items DO
      itemKey: FormWindow.ItemKey = myItem.ORD;
      IF ~FormWindow.HasBeenChanged [fw, itemKey] THEN LOOP;
      SELECT myItem FROM
        filter => {
          filterString: XString.ReaderBody ← FormWindow.LookAtTextItemValue [fw, itemKey];
          NSString.FreeString [zone, nsStringFilter];
          nsStringFilter ← XString.NSStringFromReader [@filterString, zone];
          sfFilter ← IF XString.Empty [@filterString] THEN NSFile.nullFilter
            ELSE [matches[attribute: [name[nsStringFilter]]]];
          FormWindow.DoneLookingAtTextItemValue [fw, itemKey];
          };
        ENDCASE;
      ENDLOOP;
    };

  SetDefaults: PROC [fw: Window.Handle] = {
    FormWindow.SetTextItemValue [fw, Items.filter.ORD, NIL];
    };

  -- Main line code

  Init[];

  END.

5-Apr-85  9:22:23 - Holbrook - Make prototype folder readonly (AR  13882)
7-May-85 16:12:26 - Holbrook - use FolderColumns instead of FolderOps
```

```
-- PublicCommands.cm
-- Trow        4-Oct-89 12:23:58 PDT

Compiler PublicCommands.mesa PublicCommandsCourier.mesa
Compiler PublicCommandsDescription.mesa PublicCommandsClientImpl.mesa PublicCommandsServerImpl.mesa
```

```
-- PublicCommands.courier
--      Copyright 1989 by Xerox Corporation.  All rights reserved.
--      Wes Irish, October 4, 1989 11:58:33 am PDT
--      Jay Trow,  4-Oct-89 12:21:31

-- The purpose of this Courier program is to allow a client to have a server perform some "public"
commands
-- on the client's behalf.

-- The main intent of this program is to allow non SUN/UNIX systems (such as ViewPoint) to "access"
SUN/UNIX
-- systems in a simple-minded public fashion.  The idea is for the server to be able to provide a
limited
-- number of generally useful "public" commands to clients at large, such as "grep"ing of various
"public"
-- databases (files).

PublicCommands: PROGRAM 2220 VERSION 1 = BEGIN

--  Types and Constants

        Status: TYPE = INTEGER;

--  Remote Procedures

        Grep: PROCEDURE [matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: STRING, file: STRING,
        output: SINK]
            RETURNS [status: Status]
            REPORTS [ServiceError, TransferError] = 1;
            -- "pattern" to be matched (the server will provide the enclosing quotes if needed).
            -- "matchCase" if TRUE then case is significant.
            -- "matchWords" if TRUE then patterns must match whole words.
            -- "file" should contain a file name or pattern to be "grep"ed.  This should NOT include a
            path
            --    since the path will automatically be prepended by the server.  This allows the server
            to make
            --    available only those files that are for "public" consumption.
            -- "output" will get all output (including error output) resulting from the command.
            -- "status" will contain the completion status of the command (follows the UNIX
            convention).
            -- FYI: The EOL character(s) in the "output" stream is server dependent.  Use
            GetEOLConvention to
            --    determine what this will be.

        GetEOLConvention: PROCEDURE
            RETURNS [eolConvention: STRING]
            REPORTS [ServiceError] = 2;
            -- Returns the server dependent line separation character(s).

--  Remote Errors

        ServiceError: ERROR [problem: ServiceProblem] = 100;
        ServiceProblem: TYPE = {
            cannotAuthenticate(0),      -- generally, an Authentication.CallProblem on the server
            serviceFull(1),             -- no more operations of that type can be accepted
            serviceUnavailable(2),      -- operations of that type are currently disabled
            notPublic(3)                -- the server decided that something in the way that the
            command was
                                        --   requested could compromise security (questionable
                                        arguments, for example)
            };

        TransferError: ERROR [problem: TransferProblem] = 101;
        TransferProblem: TYPE = {
            aborted(0)                  -- the transfer was aborted by the source or sink
            };

    END.
```

```
-- PublicCommands.mesa
-- Trow  4-Oct-89 13:11:55

DIRECTORY Stream, System;

PublicCommands: DEFINITIONS
 = {

  BindHandle: TYPE = LONG POINTER TO READONLY BindObject;
  BindObject: TYPE;

  Status: TYPE = INTEGER;

  Grep: PROCEDURE [
    bH: BindHandle, matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: LONG STRING, file: LONG STRING,
    output: Stream.Handle]
    RETURNS [status: Status];

  GetEOLConvention: PROCEDURE [
    bH: BindHandle]
    RETURNS [eolConvention: LONG STRING];

  FreeGetEOLConventionResults: PROCEDURE[bH: BindHandle, eolConvention: LONG STRING];

  ServiceError: ERROR [
    bH: BindHandle, problem: ServiceProblem];

  ServiceProblem: TYPE = MACHINE DEPENDENT {cannotAuthenticate(0), serviceFull(1),
  serviceUnavailable(2), notPublic(3), (CARDINAL.LAST - 1)};

  TransferError: ERROR [
    bH: BindHandle, problem: TransferProblem];

  TransferProblem: TYPE = MACHINE DEPENDENT {aborted(0), (CARDINAL.LAST - 1)};

  RemoteBind: PROCEDURE [
    host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
    RETURNS[bH: BindHandle];

  RemoteUnbind: PROCEDURE[bH: BindHandle] RETURNS [nil: BindHandle];



  }.
```

```
-- PublicCommandsClientImpl.mesa
-- Trow  4-Oct-89 13:22:18

DIRECTORY Courier, Heap, System, PublicCommands, PublicCommandsCourier, Stream, XStream;

PublicCommandsClientImpl: PROGRAM
  IMPORTS Heap, Courier, PublicCommandsCourier, XStream
  EXPORTS PublicCommands = {

  BindHandle: TYPE = LONG POINTER TO READONLY BindObject;
  BindObject: PUBLIC TYPE = Courier.Object;

  Grep: PUBLIC PROCEDURE [
    bH: BindHandle, matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: LONG STRING, file: LONG STRING,
    output: Stream.Handle]
    RETURNS [status: PublicCommands.Status]= {
    args: PublicCommandsCourier.GrepArgs ← [matchCase, matchWords, pattern, file,
    XStream.Make[[stream[sH: output]]]];
    res: PublicCommandsCourier.GrepRes;
    DoCourierCall[
      cH: bH, procedureNumber: PublicCommandsCourier.Grep,
      arguments: [@args, PublicCommandsCourier.DescribeGrepArgs],
      results: [@res, PublicCommandsCourier.DescribeGrepRes],
      streamCheckoutProc: XStream.UserCheckout!
        UNWIND => XStream.Destroy[args.output]];
    [status] ← res;
    XStream.Destroy[args.output];
    };

  GetEOLConvention: PUBLIC PROCEDURE [
    bH: BindHandle]
    RETURNS [eolConvention: LONG STRING]= {
    res: PublicCommandsCourier.GetEOLConventionRes;
    DoCourierCall[
      cH: bH, procedureNumber: PublicCommandsCourier.GetEOLConvention,
      results: [@res, PublicCommandsCourier.DescribeGetEOLConventionRes],
      streamCheckoutProc: XStream.UserCheckout!
        UNWIND => NULL];
    [eolConvention] ← res;
    };

  FreeGetEOLConventionResults: PUBLIC PROCEDURE[
    bH: BindHandle, eolConvention: LONG STRING] = {
    res: PublicCommandsCourier.GetEOLConventionRes;
    res.eolConvention ← eolConvention;
    Courier.Free[[@res, PublicCommandsCourier.DescribeGetEOLConventionRes], bH.zone];
    };

  ServiceError: PUBLIC ERROR [
    bH: BindHandle, problem: PublicCommands.ServiceProblem]= CODE;

  TransferError: PUBLIC ERROR [
    bH: BindHandle, problem: PublicCommands.TransferProblem]= CODE;


  RemoteBind: PUBLIC PROCEDURE[
    host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
    RETURNS[bH: BindHandle] = {
    IF zone = NIL THEN zone ← Heap.systemZone;
    bH ← Courier.Create[
      remote: host, programNumber: PublicCommandsCourier.programNumber,
    versionNumber: PublicCommandsCourier.version, zone: zone, classOfService: transactional];
    };

  RemoteUnbind: PUBLIC PROCEDURE[
    bH: BindHandle] RETURNS [nil: BindHandle] = {
    nil ← NIL;
    IF bH # NIL THEN Courier.Delete[bH];
    };


  DoCourierCall: PROCEDURE[
    cH: Courier.Handle, procedureNumber: CARDINAL,
    arguments: Courier.Parameters ← Courier.nullParameters,
    results: Courier.Parameters ← Courier.nullParameters,
    streamCheckoutProc: PROCEDURE [cH: Courier.Handle] ← NIL] = {
```

```
    ENABLE {
      Courier.RemoteErrorSignalled => {
        SELECT errorNumber FROM
          100 => DoServiceError[cH, arguments];
          101 => DoTransferError[cH, arguments];
          ENDCASE;
        };
      Courier.Error => NULL;
      };
    [] ← Courier.Call[
      cH: cH, procedureNumber: procedureNumber, arguments: arguments,
      results: results, streamCheckoutProc: streamCheckoutProc];
    };

  DoServiceError: PROCEDURE[
    bH: BindHandle, arguments: Courier.Arguments] = {
    args: PublicCommandsCourier.ServiceErrorArgs;
    arguments[[@args, PublicCommandsCourier.DescribeServiceErrorArgs]];
    ERROR ServiceError[bH, args.problem];
    };

  DoTransferError: PROCEDURE[
    bH: BindHandle, arguments: Courier.Arguments] = {
    args: PublicCommandsCourier.TransferErrorArgs;
    arguments[[@args, PublicCommandsCourier.DescribeTransferErrorArgs]];
    ERROR TransferError[bH, args.problem];
    };



  }.
```

```
-- PublicCommandsCourier.mesa
-- Trow  4-Oct-89 13:12:29

DIRECTORY Courier, PublicCommands, XStream;

PublicCommandsCourier: DEFINITIONS
 = {

  programNumber: LONG CARDINAL = 2220;
  version: CARDINAL = 1;
  DescribeStatus: Courier.Description;
  Grep: CARDINAL = 1;
  GetEOLConvention: CARDINAL = 2;
  ServiceError: CARDINAL = 100;
  DescribeServiceProblem: Courier.Description;
  TransferError: CARDINAL = 101;
  DescribeTransferProblem: Courier.Description;
  GrepArgs: TYPE = RECORD[matchCase: BOOLEAN, matchWords: BOOLEAN, pattern: LONG STRING, file: LONG
  STRING, output: XStream.Handle];
  DescribeGrepArgs: Courier.Description;

  GrepRes: TYPE = RECORD[status: PublicCommands.Status];
  DescribeGrepRes: Courier.Description;

  GetEOLConventionRes: TYPE = RECORD[eolConvention: LONG STRING];
  DescribeGetEOLConventionRes: Courier.Description;

  ServiceErrorArgs: TYPE = RECORD[problem: PublicCommands.ServiceProblem];
  DescribeServiceErrorArgs: Courier.Description;

  TransferErrorArgs: TYPE = RECORD[problem: PublicCommands.TransferProblem];
  DescribeTransferErrorArgs: Courier.Description;



  }.
```

```
-- PublicCommandsDescription.mesa
-- Trow  4-Oct-89 13:20:27

DIRECTORY Courier, PublicCommandsCourier, PublicCommands, XStream;

PublicCommandsDescription: PROGRAM
  IMPORTS XStream
  EXPORTS PublicCommandsCourier = PUBLIC {

  DescribeStatus: Courier.Description = {
    p: LONG POINTER TO PublicCommands.Status = notes.noteSize[
      SIZE[PublicCommands.Status]];
    };

  DescribeGrepArgs: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.GrepArgs = notes.noteSize[
      SIZE[PublicCommandsCourier.GrepArgs]];
    notes.noteString[@p.pattern];
    notes.noteString[@p.file];
    notes.noteParameters[@p.output, XStream.DescribeSink];
    };

  DescribeGrepRes: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.GrepRes = notes.noteSize[
      SIZE[PublicCommandsCourier.GrepRes]];
    notes.noteParameters[@p.status, DescribeStatus];
    };

  DescribeGetEOLConventionRes: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.GetEOLConventionRes = notes.noteSize[
      SIZE[PublicCommandsCourier.GetEOLConventionRes]];
    notes.noteString[@p.eolConvention];
    };

  DescribeServiceErrorArgs: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.ServiceErrorArgs = notes.noteSize[
      SIZE[PublicCommandsCourier.ServiceErrorArgs]];
    notes.noteParameters[@p.problem, DescribeServiceProblem];
    };

  DescribeServiceProblem: Courier.Description = {
    p: LONG POINTER TO PublicCommands.ServiceProblem = notes.noteSize[
      SIZE[PublicCommands.ServiceProblem]];
    };

  DescribeTransferErrorArgs: Courier.Description = {
    p: LONG POINTER TO PublicCommandsCourier.TransferErrorArgs = notes.noteSize[
      SIZE[PublicCommandsCourier.TransferErrorArgs]];
    notes.noteParameters[@p.problem, DescribeTransferProblem];
    };

  DescribeTransferProblem: Courier.Description = {
    p: LONG POINTER TO PublicCommands.TransferProblem = notes.noteSize[
      SIZE[PublicCommands.TransferProblem]];
    };



  }.
```

```
-- PublicCommandsServerImpl.mesa
-- Trow  4-Oct-89 13:24:44

DIRECTORY Courier, Heap, System, PublicCommands, PublicCommandsCourier, Stream, XStream;

PublicCommandsServerImpl: PROGRAM
  IMPORTS Heap, Courier, PublicCommands, PublicCommandsCourier, Stream, XStream
  EXPORTS PublicCommands = {

  BindHandle: TYPE = LONG POINTER TO READONLY BindObject;
  BindObject: PUBLIC TYPE = Courier.Object;


  Dispatcher: Courier.Dispatcher = {
    ENABLE {
      PublicCommands.ServiceError => GOTO error100;
      PublicCommands.TransferError => GOTO error101;
      };
    SELECT procedureNumber FROM
      1 => DoGrep[cH, arguments, results];
      2 => DoGetEOLConvention[cH, arguments, results];
      ENDCASE => ERROR Courier.NoSuchProcedureNumber;
    EXITS
      error100 => {
        args: PublicCommandsCourier.ServiceErrorArgs ← [problem];
        Courier.SignalRemoteError[100, [@args, PublicCommandsCourier.DescribeServiceErrorArgs]];
        };
      error101 => {
        args: PublicCommandsCourier.TransferErrorArgs ← [problem];
        Courier.SignalRemoteError[101, [@args, PublicCommandsCourier.DescribeTransferErrorArgs]];
        };
    };

  DoGrep: PROCEDURE[
    cH: Courier.Handle, arguments: Courier.Arguments, results: Courier.Results] = {
    args: PublicCommandsCourier.GrepArgs;
    res: PublicCommandsCourier.GrepRes;
    GrepBulkData: PROCEDURE[xH: XStream.Handle] = {
      xstream: Stream.Handle ← XStream.Create[xH];
      [res.status] ← PublicCommands.Grep[NIL, args.matchCase, args.matchWords, args.pattern, args.file,
      xstream!UNWIND => Stream.Delete[xstream]];
      Stream.Delete[xstream];
      };
    arguments[[@args, PublicCommandsCourier.DescribeGrepArgs]];
    XStream.ServerCheckout[cH, [proc[GrepBulkData]]];
    [] ← results[[@res, PublicCommandsCourier.DescribeGrepRes]];
    Courier.Free[[@args, PublicCommandsCourier.DescribeGrepArgs], cH.zone];
    };

  DoGetEOLConvention: PROCEDURE[
    cH: Courier.Handle, arguments: Courier.Arguments, results: Courier.Results] = {
    res: PublicCommandsCourier.GetEOLConventionRes;
    arguments[];
    [res.eolConvention] ← PublicCommands.GetEOLConvention[NIL];
    [] ← results[[@res, PublicCommandsCourier.DescribeGetEOLConventionRes]];
    PublicCommands.FreeGetEOLConventionResults[bH: NIL, eolConvention: res.eolConvention];
    };


  started: BOOLEAN ← FALSE;
  RemoteBind: PUBLIC PROCEDURE [
    host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
    RETURNS[bH: BindHandle] = {
    bH ← NIL;
    IF zone = NIL THEN zone ← Heap.systemZone;
    IF started THEN RETURN;
    Courier.ExportRemoteProgram[
      programNumber: PublicCommandsCourier.programNumber,
      versionRange: [PublicCommandsCourier.version,PublicCommandsCourier.version],
      dispatcher: Dispatcher, serviceName: "PublicCommands"L,
      zone: zone, classOfService: transactional];
    started ← TRUE;
  };

  RemoteUnbind: PUBLIC PROCEDURE[
    bH: BindHandle] RETURNS [nil: BindHandle] = {
```

```
    nil ← NIL;
    IF ~started THEN RETURN;
    Courier.UnexportRemoteProgram[
      programNumber: PublicCommandsCourier.programNumber,
      versionRange: [PublicCommandsCourier.version,PublicCommandsCourier.version]];
    started ← FALSE;
    };


}.
```

```
-- File: Speller.config - Last edited by:
-- Mark Hahn     21-Jan-87 16:35:53
-- D. MacKay     23-Sep-87 12:28:44

-- Copyright (C) 1986 Xerox Corporation. All rights reserved.

Speller: CONFIGURATION
  IMPORTS Atom, Attention, BackgroundProcess, Containee, Context, Display, DocInterchangeDefs, FormWindow, Heap, NSFile, NSFileStream,
  NSSegment, Process, Prototype, Selection, SimpleTextDisplay, Space, StarWindowShell, Stream, UserTerminal, XChar, XFormat, XMessage,
  XString, XToken

  CONTROL SpellerMsgImpl, SpellerImpl = {
    SpellerFormImpl;
    SpellerImpl;
    SpellerMsgImpl;
    SpellerVMImpl;
    }.
```

ideas for multi-character transliteration

```
-- SpellerDefs.mesa
-- Frank        13-Aug-86 14:44:38
-- Mark         16-Mar-87 11:44:01

-- Copyright (C) Xerox Corporation 1986. All rights reserved.

DIRECTORY
  NSFile USING [Handle, nullHandle],
  Space USING [Interval, nullInterval],
  Window USING [Handle],
  XChar USING [Character, null],
  XMessage USING [MsgKey],
  XString USING [Reader, ReaderBody, WriterBody];

SpellerDefs: DEFINITIONS = {

  z: UNCOUNTED ZONE;

  NoMoreRoom: SIGNAL;

  lastCardinal: CARDINAL = CARDINAL.LAST;
  Letters: TYPE = CARDINAL['a.ORD..'z.ORD];
  SpellerFileType: CARDINAL = 44693;

  ElementDesc: TYPE = LONG DESCRIPTOR FOR ARRAY OF Element;
  Data: TYPE = LONG POINTER TO DataObject;
  DataObject: TYPE = RECORD [
    busy: BOOLEAN ← FALSE,
    spaceBase: Space.Interval ← Space.nullInterval,
    handle: NSFile.Handle ← NSFile.nullHandle,
    nextFreeElement: CARDINAL ← 0,
    lastElement: CARDINAL ← 0,
    rootLetters: RootLetterArrayPtr ← NIL,
    tree: ElementDesc ← DESCRIPTOR[NIL,0]
  ];

  Element: TYPE = RECORD [
    ch: XChar.Character ← XChar.null,
    eow: BOOL ← FALSE,                    -- End Of Word
    child,
    sibling: CARDINAL ← lastCardinal
  ];

  RootElement: TYPE = RECORD [
    eow: BOOL ← FALSE,                    -- End Of Word
    child: CARDINAL ← lastCardinal
  ];

  RootLetterArrayPtr: TYPE = LONG POINTER TO RootLetterArray;
  RootLetterArray: TYPE = ARRAY Letters OF RootElement;

  -- PROCEDURES
  GetMessage: PROCEDURE[key: XMessage.MsgKey] RETURNS [msg: XString.ReaderBody];
  CheckOrInsertWord: PROC [data: Data, r: XString.Reader, checkSpelling: BOOL ← FALSE] RETURNS [correctlySpelled: BOOL ← FALSE];
  DeleteWord: PROC [data: Data, r: XString.Reader] RETURNS [found: BOOL ← FALSE];
  ExpandFile: PROCEDURE[data: Data];
  GetContext: PROCEDURE[body: Window.Handle] RETURNS[data: Data];
  List: PROC [data: Data, r: XString.Reader] RETURNS [XString.WriterBody];
  MakeFormWindow: PROC[wh: Window.Handle];

  MessageKey: TYPE = {spellerName, unknownAction, noWordSpecified, notFound, deleted, problemsWithDoc, couldntOpenFile, wrongType, word,
  read, delete, list, checkSpelling, feedback, noMatch, listWords, lessThan, done, insertionComplete, deletionComplete,
  checkingSpelling, noMoreRoom};

}...
```

```
-- SpellerFormImpl.mesa
-- Mark Hahn     16-Mar-87 11:56:43

-- Copyright (C) Xerox Corporation 1986. All rights reserved.

-- PROCEDURES:
-- AddToNotFoundList
-- CheckText
-- DeleteData
-- DisplayWordsNotFound
-- DoLayout
-- EnumerateDocument
-- GetSelectedFileAsText
-- GetSelectedText
-- GetSelectionAsReader
-- ListData
-- MakeFormItems
-- MakeFormWindow (public)
-- ReadData

DIRECTORY
  Atom USING [ATOM, GetProp, MakeAtom],
  Attention USING [Post],
  Auth USING [IdentityHandle],
  BackgroundProcess USING [CallBackProc, ManageMe],
  DocInterchangeDefs USING [Close, Doc, Enumerate, EnumProcsRecord, Open, OpenStatus, TextProc],
  FormWindow USING [AppendItem, AppendLine, CommandProc, Create, GetBooleanItemValue, GetTextItemValue, LayoutProc, Line,
  MakeBooleanItem, MakeCommandItem, MakeItemsProc, MakeTextItem, SetTextItemValue],
  NSFile USING [Close, GetType, Handle, Logoff, Logon, nullHandle, OpenByReference, Reference, Session, Type],
  NSFileStream USING [Create, GetLength],
  Process USING [Detach, priorityBackground, SetPriority],
  Selection USING [CanYouConvert, Convert, Enumerate, EnumerationProc, Free, Value],
  SpellerDefs USING [CheckOrInsertWord, Data, DeleteWord, GetContext, GetMessage, List, MessageKey, z],
  Stream USING [Delete, Handle],
  Window USING [Handle, Object],
  XFormat USING [Blanks, CR, Decimal, Handle, Object, Reader, WriterObject],
  XString USING [AppendStream, Compare, CopyToNewReaderBody, Empty, FreeReaderBytes, FreeWriterBytes, FromSTRING, NewWriterBody,
  nullReaderBody, Reader, ReaderBody, ReaderFromWriter, Relation, WriterBody],
  XToken USING [Alphabetic, Filtered, FreeReaderHandle, FreeTokenString, Handle, Object, ReaderToHandle];

SpellerFormImpl: MONITOR
  IMPORTS Atom, Attention, BackgroundProcess, DocInterchangeDefs, FormWindow, NSFile, NSFileStream, Process, Selection, SpellerDefs,
  Stream, XFormat, XString, XToken
  EXPORTS SpellerDefs = {
  OPEN Defs: SpellerDefs;

  -- TYPEs
  NotFoundPtr: TYPE = LONG POINTER TO NotFoundNode;
  NotFoundNode: TYPE = RECORD[
    word: XString.ReaderBody ← XString.nullReaderBody,
    numFound: CARDINAL ← 1,
    link: NotFoundPtr ← NIL];
  FormItems: TYPE = {word, read, checkSpelling, delete, list, feedback};

  -- CONSTANTs, SIGNALs, and ZONEs
  FileProblem: SIGNAL = CODE;
  FileIsDoc: SIGNAL = CODE;
  NoSelection: SIGNAL = CODE;
  docFileType: NSFile.Type = 4353;
  simpleTextDoc: NSFile.Type = 2;

  -- MONITOR ENTRY PROCEDURES
  -- data.busy is the monitor invariant. Refuse requests
  -- if another process is in progress.

  -- Delete the word specified in the form. If the word
  -- in 0 chars long post a msg. If the word is found post
  -- success message; otherwise, post not found
  DeleteData: ENTRY FormWindow.CommandProc = {
    data: Defs.Data ← Defs.GetContext[window];
    IF data.busy THEN RETURN;
    data.busy ← TRUE;
    Process.Detach[FORK DoBackgroundDelete[window]];
  };

  -- Called when user wishes to list contents of dictionary. The
  -- text in the "word" field of the form is used as a filter
  -- (e.g. if text = "to" then all words beginning with "to" would
  -- be listed).
  ListData: ENTRY FormWindow.CommandProc = {
    data: Defs.Data ← Defs.GetContext[window];
    IF data.busy THEN RETURN;
    data.busy ← TRUE;
    Process.Detach[FORK DoBackgroundList[window]];
  };

  -- Reset the monitor invariant
  MakeBusyFalse: ENTRY PROC[data: Defs.Data] = {
    data.busy ← FALSE;
  };

  -- Gets the currently selected document/simple doc/text/ or word and
  -- parses the information into valid words. Each word is passed off to
  -- a procedure to see if it exists in the dictionary; if not the word
  -- is saved. When all data has been processed, the list of words that
  -- were not found is displayed to the user.
  ReadData: ENTRY FormWindow.CommandProc = {
```

```
    data: Defs.Data ← Defs.GetContext[window];
    IF data.busy THEN RETURN;
    data.busy ← TRUE;
    Process.Detach[FORK DoBackgroundRead[window]];
  };

  -- END OF MONITOR


  -- Add the word to the linked list. If the word already exists, bump
  -- the counter for that word. Note that there is a dummy header node
  -- in the linked list that should not be displayed to the user.
  AddToNotFoundList: PROCEDURE[word: XString.Reader, curr: NotFoundPtr] = {
    rel: XString.Relation ← equal;
    WHILE curr.link # NIL DO
      rel ← XString.Compare[r1:@curr.link.word, r2: word];
      IF rel = equal THEN {      -- bump counter and release storage
        curr.numFound ← curr.numFound + 1;
        [] ← XToken.FreeTokenString[r: word];
        RETURN};
      IF rel = greater THEN {
        curr.link ← Defs.z.NEW[NotFoundNode ← [word: word↑, link: curr.link]];
        RETURN};
      curr ← curr.link
    ENDLOOP;
    curr.link ← Defs.z.NEW[NotFoundNode ← [word: word↑]];
  };

  -- Parse a reader and check to see if each word is contained in the
  -- dictionary. If the word is not found and we have spell checking
  -- enabled, then save the word in a linked list for later display.
  CheckText: PROC[data: Defs.Data, text: XString.Reader, check: BOOLEAN, head: NotFoundPtr] = {
    word: XString.ReaderBody;
    correct: BOOLEAN ← FALSE;
    tokenHandle: XToken.Handle ← XToken.ReaderToHandle[r: text];
    DO
      word ← XToken.Filtered[h: tokenHandle, data: NIL, filter: XToken.Alphabetic, skip: nonToken];
      IF XString.Empty[@word] THEN EXIT;
      correct ← Defs.CheckOrInsertWord[data: data, r: @word, checkSpelling: check];
      IF check THEN
        IF ~correct THEN AddToNotFoundList[@word, head]
        ELSE [] ← XToken.FreeTokenString[r: @word]
      ELSE [] ← XToken.FreeTokenString[r: @word];
    ENDLOOP;
    [] ← XToken.FreeReaderHandle[h: tokenHandle];
  };

  DoBackgroundDelete: PROCEDURE[window: Window.Handle] = {
    ENABLE UNWIND => { -- restore monitor invariant
      data: Defs.Data ← Defs.GetContext[window];
      MakeBusyFalse[data];};

    RealDelete: BackgroundProcess.CallBackProc = {
      data: Defs.Data ← Defs.GetContext[window];
      wb: XString.WriterBody ← XString.NewWriterBody[maxLength: 50, z: Defs.z];
      xfo: XFormat.Object ← XFormat.WriterObject[@wb];
      text: XString.ReaderBody ← FormWindow.GetTextItemValue[
        window: window,
        item: FormItems.word.ORD,
        zone: Defs.z];
      IF XString.Empty[@text] THEN {
        rb: XString.ReaderBody ← Defs.GetMessage [
          Defs.MessageKey.noWordSpecified.ORD];
        Attention.Post[@rb];
        MakeBusyFalse[data];
        RETURN[failure]};
       -- valid data so parse and delete each word
       BEGIN
         rb: XString.ReaderBody;
         word: XString.ReaderBody;
         tokenHandle: XToken.Handle ← XToken.ReaderToHandle[r: @text];
         DO
          word ← XToken.Filtered[h: tokenHandle,
            data: NIL, filter: XToken.Alphabetic, skip: nonToken];
          IF XString.Empty[@word] THEN EXIT;
          IF Defs.DeleteWord[data, @word] THEN {
            rb ← Defs.GetMessage [Defs.MessageKey.deleted.ORD];
            XFormat.Reader[@xfo, @word];
            XFormat.Reader[@xfo, @rb];
            XFormat.CR[@xfo];
          }
          ELSE {
            rb ← Defs.GetMessage [Defs.MessageKey.notFound.ORD];
            XFormat.Reader[@xfo, @word];
            XFormat.Reader[@xfo, @rb];
            XFormat.CR[@xfo];
          };
          [] ← XToken.FreeTokenString[r: @word];
        ENDLOOP;
        rb ← Defs.GetMessage [Defs.MessageKey.deletionComplete.ORD];
        XFormat.Reader[@xfo, @rb];
        FormWindow.SetTextItemValue[
          window: window,
          item: FormItems.feedback.ORD,
          newValue: XString.ReaderFromWriter[@wb]];
        [] ← XToken.FreeReaderHandle[h: tokenHandle];
        XString.FreeReaderBytes[@text, Defs.z];
```

```
        XString.FreeWriterBytes[@wb];
      END;
      MakeBusyFalse[data];
      RETURN[success];
    };

    name: XString.ReaderBody ← XString.FromSTRING["Delete"];
    Process.SetPriority[Process.priorityBackground];
    [] ← BackgroundProcess.ManageMe[name: @name, callBackProc: RealDelete];
  };

  -- Collects the words stored in a linked list, appends them to
  -- a writer, and displays them in the formwindow. Note the first
  -- node in the linked list is a dummy and should not be displayed.
  DisplayWordsNotFound: PROC[curr: NotFoundPtr, window: Window.Handle] = {
    wb: XString.WriterBody ← XString.NewWriterBody[maxLength: 2048, z: Defs.z];
    rb: XString.ReaderBody ← Defs.GetMessage [Defs.MessageKey.checkingSpelling.ORD];
    xfo: XFormat.Object ← XFormat.WriterObject[@wb];
    first: NotFoundPtr ← curr;
    last: NotFoundPtr;
    XFormat.Reader[h: @xfo, r: @rb];
    XFormat.CR[h: @xfo];
    curr ← curr.link;    -- skip first node
    WHILE curr # NIL DO
      XFormat.Reader[h: @xfo, r: @curr.word];
      XFormat.Blanks[h: @xfo, n: 5];
      XFormat.Decimal[h: @xfo, n: curr.numFound];
      XFormat.CR[h: @xfo];
      last ← curr;
      curr ← curr.link;
      [] ← XToken.FreeTokenString[@last.word];
      Defs.z.FREE[@last];        -- free first node
    ENDLOOP;
    rb ← Defs.GetMessage [Defs.MessageKey.done.ORD];
    XFormat.Reader[h: @xfo, r: @rb];
    XString.FreeReaderBytes[@first.word, Defs.z];        -- free dummy head node
    Defs.z.FREE[@first];
    FormWindow.SetTextItemValue[
      window: window,
      item: FormItems.feedback.ORD,
      newValue: XString.ReaderFromWriter[@wb]];
    XString.FreeWriterBytes[w: @wb];
  };

  -- LayoutProc for the form items
  DoLayout: PUBLIC FormWindow.LayoutProc = { OPEN FW: FormWindow;
    line: FW.Line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.word.ORD, line:line];
    line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.read.ORD, line:line];
    FW.AppendItem[window:window, item: FormItems.delete.ORD, line:line];
    FW.AppendItem[window:window, item: FormItems.list.ORD, line:line];
    line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.checkSpelling.ORD, line:line];
    line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.feedback.ORD, line:line];
  };

  -- open and enumerate the selected document. Check each word
  -- of text to see if it is in the  dictionary
  EnumerateDocument: PROC[data: Defs.Data, check: BOOLEAN, head: NotFoundPtr, window: Window.Handle] = {

    -- check each block of text found in the document
    TextProc: DocInterchangeDefs.TextProc = {
      CheckText[data, text, check, head]};

    element: Selection.Value ← Selection.Convert[file]; -- get reference to file
    ref: LONG POINTER TO NSFile.Reference ← element.value;
    enumProcs: DocInterchangeDefs.EnumProcsRecord ← [textProc:TextProc];
    status: DocInterchangeDefs.OpenStatus;
    doc: DocInterchangeDefs.Doc;
    user: Atom.ATOM ← Atom.MakeAtom["CurrentUser"L];
    prop: Atom.ATOM ← Atom.MakeAtom["IdentityHandle"L];
    identity: Auth.IdentityHandle = Atom.GetProp[user, prop]↑.value;
    session: NSFile.Session ← NSFile.Logon[identity];
    -- open source document and enumerate in a separate NSFile session
    [doc, status] ← DocInterchangeDefs.Open[
       docFileRef: ref↑, session: session];
    IF status # ok THEN GOTO Exit;
    [] ← DocInterchangeDefs.Enumerate[
      textContainer: [doc[h:doc]], procs: @enumProcs, clientData: head];
    DocInterchangeDefs.Close[docPtr:@doc];
    NSFile.Logoff[session];
    IF check THEN DisplayWordsNotFound[head, window]
    ELSE {
      rb: XString.ReaderBody ← Defs.GetMessage [Defs.MessageKey.insertionComplete.ORD];
      FormWindow.SetTextItemValue[
        window: window,
        item: FormItems.feedback.ORD,
        newValue: @rb]};
    EXITS Exit => {
      rb: XString.ReaderBody ← Defs.GetMessage [
        Defs.MessageKey.problemsWithDoc.ORD];
      Attention.Post[@rb]};
  };

  -- Return the contents of the selected file as a readerbody.
```

```
    -- If the file is a document raise a signal since docs are handled
    -- elsewhere. If the file is not a simple text doc raise the
    -- FileProblem signal and post a message. If the file cannot be
    -- opened, post a message and raise FileProblem.
    GetSelectedFileAsText: PROCEDURE[data: Defs.Data] RETURNS[rb: XString.ReaderBody] = {
      element: Selection.Value ← Selection.Convert[file]; -- get reference to file
      ref: LONG POINTER TO NSFile.Reference ← element.value;
      type: NSFile.Type;
      handle: NSFile.Handle ← NSFile.OpenByReference[reference: ref↑];
      IF handle = NSFile.nullHandle THEN {
        rb: XString.ReaderBody ← Defs.GetMessage [
          Defs.MessageKey.couldntOpenFile.ORD];
        Attention.Post[@rb];
        Selection.Free[@element];
        SIGNAL FileProblem};

      -- make sure file is of the correct type
      type ← NSFile.GetType[handle];
      IF type = docFileType THEN {
        NSFile.Close[handle];
        SIGNAL FileIsDoc};          -- handle docs elsewhere
      IF type # simpleTextDoc THEN {
        rb: XString.ReaderBody ← Defs.GetMessage [
          Defs.MessageKey.wrongType.ORD];
        Attention.Post[@rb];
        Selection.Free[@element];
        NSFile.Close[handle];
        SIGNAL FileProblem};
      BEGIN
        stream: Stream.Handle ← NSFileStream.Create[file: handle];
        length: CARDINAL ← MIN[
          LAST[CARDINAL],
          CARDINAL[NSFileStream.GetLength[[stream]]] ];
        wb: XString.WriterBody ← XString.NewWriterBody[
          maxLength: length, z: Defs.z];
        [] ← XString.AppendStream[
          to: @wb, from: stream, nBytes: length];
        rb ← XString.ReaderFromWriter[@wb]↑;
        Stream.Delete[stream];
        Selection.Free[@element];
      END;
    };

    -- get the currently selected text and return it as a readerbody
    GetSelectedText: PROC RETURNS[rb: XString.ReaderBody] = {
      NextString: Selection.EnumerationProc = {
        XFormat.Reader [@xfo, LOOPHOLE [element.value]];
        Selection.Free [@element];
      };
      wb: XString.WriterBody ← XString.NewWriterBody[maxLength:2048, z: Defs.z];
      xfo: XFormat.Object ← XFormat.WriterObject[w: @wb];
      [] ← Selection.Enumerate[NextString, string, NIL];
      rb ← XString.ReaderFromWriter[@wb]↑;
    };

    -- This proc will convert the current selection into a readerbody
    -- and return it to the caller. The selection can be a simple text
    -- doc, or a portion of selected text. The only restriction is that
    -- the total length of the selection must fit within a single reader.
    GetSelectionAsReader:PROCEDURE[data: Defs.Data]  RETURNS[rb: XString.ReaderBody ← XString.nullReaderBody] = {
      IF Selection.CanYouConvert[target: string, enumeration: TRUE] THEN
        RETURN[GetSelectedText[]]
      ELSE IF Selection.CanYouConvert[target: file, enumeration: FALSE] THEN
            RETURN[GetSelectedFileAsText[data]]
          ELSE SIGNAL NoSelection;        -- no selection so read from form
    };

    -- Performs the listing in the background
    DoBackgroundList: PROCEDURE[window: Window.Handle] = {
      ENABLE UNWIND => { -- restore monitor invariant
        data: Defs.Data ← Defs.GetContext[window];
        MakeBusyFalse[data]};

      RealList: BackgroundProcess.CallBackProc = {
        wb: XString.WriterBody;
        data: Defs.Data ← Defs.GetContext[window];
        text: XString.ReaderBody ← FormWindow.GetTextItemValue[
          window: window,
          item: FormItems.word.ORD,
          zone: Defs.z];
        tokenHandle: XToken.Handle ← XToken.ReaderToHandle[r: @text];
        word: XString.ReaderBody ← XToken.Filtered[h: tokenHandle,
          data: NIL, filter: XToken.Alphabetic, skip: nonToken];
        IF XString.Empty[@word] THEN {
          MakeBusyFalse[data];
          RETURN[failure]};
        wb ← Defs.List[data: data, r: @word];
        FormWindow.SetTextItemValue[
          window: window,
          item: FormItems.feedback.ORD,
          newValue: XString.ReaderFromWriter[@wb]];
        XString.FreeWriterBytes[@wb];
        [] ← XToken.FreeTokenString[@word];
        [] ← XToken.FreeReaderHandle[h: tokenHandle];
        MakeBusyFalse[data];
        RETURN[success];
      };
```

```
    name: XString.ReaderBody ← XString.FromSTRING["List"];
    Process.SetPriority[Process.priorityBackground];
    [] ← BackgroundProcess.ManageMe[name: @name, callBackProc: RealList];
  };

  -- Creates the form items in the formwindow
  MakeFormItems: FormWindow.MakeItemsProc = {
    rb: XString.ReaderBody ← Defs.GetMessage [Defs.MessageKey.word.ORD];
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.word.ORD,
      tag: @rb,
      width: 400];
    rb ← Defs.GetMessage [Defs.MessageKey.read.ORD];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: FormItems.read.ORD,
      commandProc: ReadData,
      commandName: @rb];
    rb ← Defs.GetMessage [Defs.MessageKey.delete.ORD];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: FormItems.delete.ORD,
      commandProc: DeleteData,
      commandName: @rb];
    rb ← Defs.GetMessage [Defs.MessageKey.list.ORD];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: FormItems.list.ORD,
      commandProc: ListData,
      commandName: @rb];
    rb ← Defs.GetMessage [Defs.MessageKey.checkSpelling.ORD];
    FormWindow.MakeBooleanItem [
      window: window,
      myKey: FormItems.checkSpelling.ORD,
      label: [string[rb]],
      initBoolean: FALSE];
    rb ← Defs.GetMessage [Defs.MessageKey.feedback.ORD];
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.feedback.ORD,
      tag: @rb,
      width: 400];
  };

  -- Called from MakeShell to layout the formwindow
  MakeFormWindow: PUBLIC PROC[wh: Window.Handle] = {
    FormWindow.Create[
      window: wh,
      makeItemsProc: MakeFormItems,
      layoutProc: DoLayout];
  };

  DoBackgroundRead: PROCEDURE[window: Window.Handle] = {
    ENABLE UNWIND => { -- restore monitor invariant
      data: Defs.Data ← Defs.GetContext[window];
      MakeBusyFalse[data]};

    RealRead: BackgroundProcess.CallBackProc = {
      data: Defs.Data ← Defs.GetContext[window];
      blank: XString.ReaderBody ← XString.FromSTRING[" "L];
      head: NotFoundPtr ← Defs.z.NEW[NotFoundNode ←
        [word: XString.CopyToNewReaderBody[r:@blank, z:Defs.z]]];
      check: BOOLEAN ← FormWindow.GetBooleanItemValue[
        window: window, item: FormItems.checkSpelling.ORD];
      text: XString.ReaderBody;
      finalStatus ← success;
      BEGIN
      ENABLE
      BEGIN
        FileProblem => {MakeBusyFalse[data]; GOTO Exit};
        NoSelection => {
         text ← FormWindow.GetTextItemValue[
           window: window,
           item: FormItems.word.ORD,
           zone: Defs.z];
         CONTINUE};
        FileIsDoc => {   -- docs are handled separately
         EnumerateDocument[data, check, head, window];
         MakeBusyFalse[data];
         GOTO Exit};
      END;
      text ← GetSelectionAsReader[data];
      END;
      CheckText[data, @text, check, head];
      XString.FreeReaderBytes[@text, Defs.z];    -- free the checked text
      IF check THEN DisplayWordsNotFound[head, window]
      ELSE {
        rb: XString.ReaderBody ← Defs.GetMessage [
          Defs.MessageKey.insertionComplete.ORD];
        FormWindow.SetTextItemValue[
          window: window,
          item: FormItems.feedback.ORD,
          newValue: @rb]};
      MakeBusyFalse[data];
      EXITS Exit => NULL;
```

```
        };

        name: XString.ReaderBody ← XString.FromSTRING["Read"];
        Process.SetPriority[Process.priorityBackground];
        [] ← BackgroundProcess.ManageMe[name: @name, callBackProc: RealRead];
      };


    }...
```

```
-- SpellerFormImplTemplate.mesa
-- Mark Hahn    16-Mar-87 13:48:48

-- Copyright (C) Xerox Corporation 1986. All rights reserved.

-- PROCEDURES:
-- AddToNotFoundList
-- CheckText
-- DeleteData
-- DisplayWordsNotFound
-- DoLayout
-- EnumerateDocument
-- GetSelectedFileAsText
-- GetSelectedText
-- GetSelectionAsReader
-- ListData
-- MakeFormItems
-- MakeFormWindow (public)
-- ReadData

DIRECTORY
  Attention USING [Post],
  DocInterchangeDefs USING [Close, Doc, Enumerate, EnumProcsRecord, Open, OpenStatus, TextProc],
  FormWindow USING [AppendItem, AppendLine, CommandProc, Create, GetBooleanItemValue, GetTextItemValue, LayoutProc, Line,
  MakeBooleanItem, MakeCommandItem, MakeItemsProc, MakeTextItem, SetTextItemValue],
  NSFile USING [Close, GetType, Handle, nullHandle, OpenByReference, Reference, Type],
  NSFileStream USING [Create, GetLength],
  Selection USING [CanYouConvert, Convert, Enumerate, EnumerationProc, Free, Value],
  SpellerDefs USING [CheckOrInsertWord, Data, DeleteWord, GetContext, GetMessage, List, MessageKey, z],
  Stream USING [Delete, Handle],
  Window USING [Handle, Object],
  XFormat USING [Blanks, CR, Decimal, Handle, Object, Reader, WriterObject],
  XString USING [AppendStream, Compare, CopyToNewReaderBody, Empty, FreeReaderBytes, FreeWriterBytes, FromSTRING, NewWriterBody,
  nullReaderBody, Reader, ReaderBody, ReaderFromWriter, Relation, WriterBody],
  XToken USING [Alphabetic, Filtered, FreeReaderHandle, FreeTokenString, Handle, ReaderToHandle];

SpellerFormImplTemplate: PROGRAM
  IMPORTS Attention, DocInterchangeDefs, FormWindow, NSFile, NSFileStream, Selection, SpellerDefs, Stream, XFormat, XString, XToken
  EXPORTS SpellerDefs = {
  OPEN Defs: SpellerDefs;

  -- TYPEs
  NotFoundPtr: TYPE = LONG POINTER TO NotFoundNode;
  NotFoundNode: TYPE = RECORD[
    word: XString.ReaderBody ← XString.nullReaderBody,
    numFound: CARDINAL ← 1,
    link: NotFoundPtr ← NIL];
  FormItems: TYPE = {word, read, checkSpelling, delete, list, feedback};

  -- CONSTANTs, SIGNALs, and ZONEs
  FileProblem: SIGNAL = CODE;
  FileIsDoc: SIGNAL = CODE;
  NoSelection: SIGNAL = CODE;
  docFileType: NSFile.Type = 4353;
  simpleTextDoc: NSFile.Type = 2;

  -- Add the word to the linked list. If the word already exists, bump
  -- the counter for that word. Note that there is a dummy header node
  -- in the linked list that should not be displayed to the user.
  AddToNotFoundList: PROCEDURE[word: XString.Reader, curr: NotFoundPtr] = {
    rel: XString.Relation ← equal;
    WHILE curr.link # NIL DO
      rel ← XString.Compare[r1:@curr.link.word, r2: word];
      IF rel = equal THEN {      -- bump counter and release storage
        curr.numFound ← curr.numFound + 1;
        [] ← XToken.FreeTokenString[r: word];
        RETURN};
      IF rel = greater THEN {
        curr.link ← Defs.z.NEW[NotFoundNode ← [word: word↑, link: curr.link]];
        RETURN};
      curr ← curr.link
    ENDLOOP;
    curr.link ← Defs.z.NEW[NotFoundNode ← [word: word↑]];
  };

  -- Parse a reader and check to see if each word is contained in the
  -- dictionary. If the word is not found and we have spell checking
  -- enabled, then save the word in a linked list for later display.
  CheckText: PROC[data: Defs.Data, text: XString.Reader, check: BOOLEAN, head: NotFoundPtr] = {
    word: XString.ReaderBody;
    correct: BOOLEAN ← FALSE;
    tokenHandle: XToken.Handle ← XToken.ReaderToHandle[r: text];
    DO
      word ← XToken.Filtered[h: tokenHandle, data: NIL, filter: XToken.Alphabetic, skip: nonToken];
      IF XString.Empty[@word] THEN EXIT;
      correct ← Defs.CheckOrInsertWord[data: data, r: @word, checkSpelling: check];
      IF check THEN
        IF ~correct THEN AddToNotFoundList[@word, head]
        ELSE [] ← XToken.FreeTokenString[r: @word]
      ELSE [] ← XToken.FreeTokenString[r: @word];
    ENDLOOP;
    [] ← XToken.FreeReaderHandle[h: tokenHandle];
  };

  DeleteData: FormWindow.CommandProc = {
    data: Defs.Data ← Defs.GetContext[window];
    wb: XString.WriterBody ← XString.NewWriterBody[maxLength: 50, z: Defs.z];
```

```
    xfo: XFormat.Object ← XFormat.WriterObject[@wb];
    text: XString.ReaderBody ← FormWindow.GetTextItemValue[
      window: window,
      item: FormItems.word.ORD,
      zone: Defs.z];
    IF XString.Empty[@text] THEN {
      rb: XString.ReaderBody ← Defs.GetMessage [
        Defs.MessageKey.noWordSpecified.ORD];
      Attention.Post[@rb];
      RETURN};
     -- valid data so parse and delete each word
     BEGIN
       rb: XString.ReaderBody;
       word: XString.ReaderBody;
       tokenHandle: XToken.Handle ← XToken.ReaderToHandle[r: @text];
       DO
        word ← XToken.Filtered[h: tokenHandle,
          data: NIL, filter: XToken.Alphabetic, skip: nonToken];
        IF XString.Empty[@word] THEN EXIT;
        IF Defs.DeleteWord[data, @word] THEN {
          rb ← Defs.GetMessage [Defs.MessageKey.deleted.ORD];
          XFormat.Reader[@xfo, @word];
          XFormat.Reader[@xfo, @rb];
          XFormat.CR[@xfo];
        }
        ELSE {
          rb ← Defs.GetMessage [Defs.MessageKey.notFound.ORD];
          XFormat.Reader[@xfo, @word];
          XFormat.Reader[@xfo, @rb];
          XFormat.CR[@xfo];
        };
        [] ← XToken.FreeTokenString[r: @word];
      ENDLOOP;
      rb ← Defs.GetMessage [Defs.MessageKey.deletionComplete.ORD];
      XFormat.Reader[@xfo, @rb];
      FormWindow.SetTextItemValue[
        window: window,
        item: FormItems.feedback.ORD,
        newValue: XString.ReaderFromWriter[@wb]];
      [] ← XToken.FreeReaderHandle[h: tokenHandle];
      XString.FreeReaderBytes[@text, Defs.z];
      XString.FreeWriterBytes[@wb];
    END;
  };


  -- Collects the words stored in a linked list, appends them to
  -- a writer, and displays them in the formwindow. Note the first
  -- node in the linked list is a dummy and should not be displayed.
  DisplayWordsNotFound: PROC[curr: NotFoundPtr, window: Window.Handle] = {
    wb: XString.WriterBody ← XString.NewWriterBody[maxLength: 2048, z: Defs.z];
    rb: XString.ReaderBody ← Defs.GetMessage [Defs.MessageKey.checkingSpelling.ORD];
    xfo: XFormat.Object ← XFormat.WriterObject[@wb];
    first: NotFoundPtr ← curr;
    last: NotFoundPtr;
    XFormat.Reader[h: @xfo, r: @rb];
    XFormat.CR[h: @xfo];
    curr ← curr.link;    -- skip first node
    WHILE curr # NIL DO
      XFormat.Reader[h: @xfo, r: @curr.word];
      XFormat.Blanks[h: @xfo, n: 5];
      XFormat.Decimal[h: @xfo, n: curr.numFound];
      XFormat.CR[h: @xfo];
      last ← curr;
      curr ← curr.link;
      [] ← XToken.FreeTokenString[@last.word];
      Defs.z.FREE[@last];       -- free first node
    ENDLOOP;
    rb ← Defs.GetMessage [Defs.MessageKey.done.ORD];
    XFormat.Reader[h: @xfo, r: @rb];
    XString.FreeReaderBytes[@first.word, Defs.z];        -- free dummy head node
    Defs.z.FREE[@first];
    FormWindow.SetTextItemValue[
      window: window,
      item: FormItems.feedback.ORD,
      newValue: XString.ReaderFromWriter[@wb]];
    XString.FreeWriterBytes[w: @wb];
  };

  -- LayoutProc for the form items
  DoLayout: PUBLIC FormWindow.LayoutProc = { OPEN FW: FormWindow;
    line: FW.Line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.word.ORD, line:line];
    line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.read.ORD, line:line];
    FW.AppendItem[window:window, item: FormItems.delete.ORD, line:line];
    FW.AppendItem[window:window, item: FormItems.list.ORD, line:line];
    line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.checkSpelling.ORD, line:line];
    line ← FW.AppendLine[window:window];
    FW.AppendItem[window:window, item: FormItems.feedback.ORD, line:line];
  };

  -- open and enumerate the selected document. Check each word
  -- of text to see if it is in the  dictionary
  EnumerateDocument: PROC[data: Defs.Data, check: BOOLEAN, head: NotFoundPtr, window: Window.Handle] = {
```

```
    -- check each block of text found in the document
    TextProc: DocInterchangeDefs.TextProc = {
      CheckText[data, text, check, head]};

    element: Selection.Value ← Selection.Convert[file]; -- get reference to file
    ref: LONG POINTER TO NSFile.Reference ← element.value;
    enumProcs: DocInterchangeDefs.EnumProcsRecord ← [textProc:TextProc];
    status: DocInterchangeDefs.OpenStatus;
    doc: DocInterchangeDefs.Doc;
    -- open source document and enumerate in a separate NSFile session
    [doc, status] ← DocInterchangeDefs.Open[docFileRef: ref↑];
    IF status # ok THEN GOTO Exit;
    [] ← DocInterchangeDefs.Enumerate[
      textContainer: [doc[h:doc]], procs: @enumProcs, clientData: head];
    DocInterchangeDefs.Close[docPtr:@doc];
    IF check THEN DisplayWordsNotFound[head, window]
    ELSE {
      rb: XString.ReaderBody ← Defs.GetMessage [Defs.MessageKey.insertionComplete.ORD];
      FormWindow.SetTextItemValue[
        window: window,
        item: FormItems.feedback.ORD,
        newValue: @rb]};
    EXITS Exit => {
      rb: XString.ReaderBody ← Defs.GetMessage [
        Defs.MessageKey.problemsWithDoc.ORD];
      Attention.Post[@rb]};
  };

  -- Return the contents of the selected file as a readerbody.
  -- If the file is a document raise a signal since docs are handled
  -- elsewhere. If the file is not a simple text doc raise the
  -- FileProblem signal and post a message. If the file cannot be
  -- opened, post a message and raise FileProblem.
  GetSelectedFileAsText: PROCEDURE[data: Defs.Data] RETURNS[rb: XString.ReaderBody] = {
    element: Selection.Value ← Selection.Convert[file]; -- get reference to file
    ref: LONG POINTER TO NSFile.Reference ← element.value;
    type: NSFile.Type;
    handle: NSFile.Handle ← NSFile.OpenByReference[reference: ref↑];
    IF handle = NSFile.nullHandle THEN {
      rb: XString.ReaderBody ← Defs.GetMessage [
        Defs.MessageKey.couldntOpenFile.ORD];
      Attention.Post[@rb];
      Selection.Free[@element];
      SIGNAL FileProblem};

    -- make sure file is of the correct type
    type ← NSFile.GetType[handle];
    IF type = docFileType THEN {
      NSFile.Close[handle];
      SIGNAL FileIsDoc};        -- handle docs elsewhere
    IF type # simpleTextDoc THEN {
      rb: XString.ReaderBody ← Defs.GetMessage [
        Defs.MessageKey.wrongType.ORD];
      Attention.Post[@rb];
      Selection.Free[@element];
      NSFile.Close[handle];
      SIGNAL FileProblem};
    BEGIN
      stream: Stream.Handle ← NSFileStream.Create[file: handle];
      length: CARDINAL ← MIN[
        LAST[CARDINAL],
        CARDINAL[NSFileStream.GetLength[[stream]]] ];
      wb: XString.WriterBody ← XString.NewWriterBody[
        maxLength: length, z: Defs.z];
      [] ← XString.AppendStream[
        to: @wb, from: stream, nBytes: length];
      rb ← XString.ReaderFromWriter[@wb]↑;
      Stream.Delete[stream];
      Selection.Free[@element];
    END;
  };

  -- get the currently selected text and return it as a readerbody
  GetSelectedText: PROC RETURNS[rb: XString.ReaderBody] = {
    NextString: Selection.EnumerationProc = {
      XFormat.Reader [@xfo, LOOPHOLE [element.value]];
      Selection.Free [@element];
    };
    wb: XString.WriterBody ← XString.NewWriterBody[maxLength:2048, z: Defs.z];
    xfo: XFormat.Object ← XFormat.WriterObject[w: @wb];
    [] ← Selection.Enumerate[NextString, string, NIL];
    rb ← XString.ReaderFromWriter[@wb]↑;
  };

  -- This proc will convert the current selection into a readerbody
  -- and return it to the caller. The selection can be a simple text
  -- doc, or a portion of selected text. The only restriction is that
  -- the total length of the selection must fit within a single reader.
  GetSelectionAsReader:PROCEDURE[data: Defs.Data]  RETURNS[rb: XString.ReaderBody ← XString.nullReaderBody] = {
    IF Selection.CanYouConvert[target: string, enumeration: TRUE] THEN
      RETURN[GetSelectedText[]]
    ELSE IF Selection.CanYouConvert[target: file, enumeration: FALSE] THEN
          RETURN[GetSelectedFileAsText[data]]
        ELSE SIGNAL NoSelection;        -- no selection so read from form
  };

  -- Called when user wishes to list contents of dictionary. The
```

```
  -- text in the "word" field of the form is used as a filter
  -- (e.g. if text = "to" then all words beginning with "to" would
  -- be listed).
  ListData: FormWindow.CommandProc = {
    wb: XString.WriterBody;
    data: Defs.Data ← Defs.GetContext[window];
    text: XString.ReaderBody ← FormWindow.GetTextItemValue[
      window: window,
      item: FormItems.word.ORD,
      zone: Defs.z];
    tokenHandle: XToken.Handle ← XToken.ReaderToHandle[r: @text];
    word: XString.ReaderBody ← XToken.Filtered[h: tokenHandle,
      data: NIL, filter: XToken.Alphabetic, skip: nonToken];
    IF XString.Empty[@word] THEN {
      RETURN};
    wb ← Defs.List[data: data, r: @word];
    FormWindow.SetTextItemValue[
      window: window,
      item: FormItems.feedback.ORD,
      newValue: XString.ReaderFromWriter[@wb]];
    XString.FreeWriterBytes[@wb];
    [] ← XToken.FreeTokenString[@word];
    [] ← XToken.FreeReaderHandle[h: tokenHandle];
  };

  -- Creates the form items in the formwindow
  MakeFormItems: FormWindow.MakeItemsProc = {
    rb: XString.ReaderBody ← Defs.GetMessage [Defs.MessageKey.word.ORD];
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.word.ORD,
      tag: @rb,
      width: 400];
    rb ← Defs.GetMessage [Defs.MessageKey.read.ORD];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: FormItems.read.ORD,
      commandProc: ReadData,
      commandName: @rb];
    rb ← Defs.GetMessage [Defs.MessageKey.delete.ORD];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: FormItems.delete.ORD,
      commandProc: DeleteData,
      commandName: @rb];
    rb ← Defs.GetMessage [Defs.MessageKey.list.ORD];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: FormItems.list.ORD,
      commandProc: ListData,
      commandName: @rb];
    rb ← Defs.GetMessage [Defs.MessageKey.checkSpelling.ORD];
    FormWindow.MakeBooleanItem [
      window: window,
      myKey: FormItems.checkSpelling.ORD,
      label: [string[rb]],
      initBoolean: FALSE];
    rb ← Defs.GetMessage [Defs.MessageKey.feedback.ORD];
    FormWindow.MakeTextItem [
      window: window,
      myKey: FormItems.feedback.ORD,
      tag: @rb,
      width: 400];
  };

  -- Called from MakeShell to layout the formwindow
  MakeFormWindow: PUBLIC PROC[wh: Window.Handle] = {
    FormWindow.Create[
      window: wh,
      makeItemsProc: MakeFormItems,
      layoutProc: DoLayout];
  };

  -- Gets the currently selected document/simple doc/text/ or word and
  -- parses the information into valid words. Each word is passed off to
  -- a procedure to see if it exists in the dictionary; if not the word
  -- is saved. When all data has been processed, the list of words that
  -- were not found is displayed to the user.
  ReadData: FormWindow.CommandProc = {
    data: Defs.Data ← Defs.GetContext[window];
    blank: XString.ReaderBody ← XString.FromSTRING[" "L];
    head: NotFoundPtr ← Defs.z.NEW[NotFoundNode ←
      [word: XString.CopyToNewReaderBody[r:@blank, z:Defs.z]]];
    check: BOOLEAN ← FormWindow.GetBooleanItemValue[
      window: window, item: FormItems.checkSpelling.ORD];
    text: XString.ReaderBody;
    BEGIN
    ENABLE
    BEGIN
      FileProblem => {GOTO Exit};
      NoSelection => {
       text ← FormWindow.GetTextItemValue[
         window: window,
         item: FormItems.word.ORD,
         zone: Defs.z];
       CONTINUE};
      FileIsDoc => {     -- docs are handled separately
```

```
        EnumerateDocument[data, check, head, window];
        GOTO Exit};
    END;
    text ← GetSelectionAsReader[data];
    END;
    CheckText[data, @text, check, head];
    XString.FreeReaderBytes[@text, Defs.z];      -- free the checked text
    IF check THEN DisplayWordsNotFound[head, window]
    ELSE {
      rb: XString.ReaderBody ← Defs.GetMessage [
        Defs.MessageKey.insertionComplete.ORD];
      FormWindow.SetTextItemValue[
        window: window,
        item: FormItems.feedback.ORD,
        newValue: @rb]};
    EXITS Exit => NULL;
  };

}...
```

```
-- SpellerImpl.mesa
-- Frank        13-Aug-86 17:33:01
-- Mark         16-Mar-87 11:44:56

-- Copyright (C) Xerox Corporation 1986. All rights reserved.

-- CanITake
-- DestroyContext
-- ExpandFile
-- FindOrCreateIconFile
-- GenericProc
-- Init
-- InitAtoms
-- InitBigPicture
-- MakeShell
-- OpenDictionary
-- PaintIconName
-- PictureProc
-- SetImplementation

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Containee USING [ChangeProc, Data, DataHandle, DefaultFileConvertProc, GenericProc, GetCachedName, GetImplementation, Implementation,
  PictureProc, ReturnTicket, SetImplementation, Ticket],
  Context USING [Create, Data, Find, Type, UniqueType],
  Display USING [Bitmap, Handle, Invert, replaceFlags, White],
  Environment USING [PageCount, wordsPerPage],
  Heap USING [Create],
  NSFile USING [Close, fullAccess, Handle, nullReference, OpenByReference, Type],
  NSSegment USING [GetSizeInPages, Map, PageCount, SetSizeInPages],
  Prototype USING [Create, Find],
  SimpleTextDisplay USING [StringIntoWindow, systemFontHeight],
  Space USING [ForceOut, Map, ScratchMap, Unmap],
  SpellerDefs USING [Data, DataObject, Element, GetMessage, lastCardinal, Letters, MakeFormWindow, MessageKey, RootLetterArray,
  SpellerFileType],
  StarWindowShell USING [Create, CreateBody, GetBody, Handle, IsCloseLegalProc, SetPreferredDims],
  UserTerminal USING [screenHeight, screenWidth],
  Window USING [Box, Dims, Handle],
  XString USING [Reader, ReaderBody];

SpellerImpl: MONITOR
  IMPORTS Atom, Containee, Context, Display, Heap, NSFile, NSSegment, Prototype, SimpleTextDisplay, Space, SpellerDefs, StarWindowShell,
  UserTerminal
  EXPORTS SpellerDefs = { OPEN Defs: SpellerDefs;

  FileHeaderRec: TYPE = RECORD [
    rootLetterArray: Defs.RootLetterArray,
    nextFreeNode: CARDINAL,
    lastNode: CARDINAL];
  FileHeaderRecPtr: TYPE = LONG POINTER TO FileHeaderRec;
  beginningPages: CARDINAL = 64;
  extensionSize: CARDINAL = 64;
  maxSize: CARDINAL = 1000;
  NoMoreRoom: PUBLIC SIGNAL = CODE;

  IconPictureBits: TYPE = ARRAY [0..256) OF WORD;
  normalIconPicture: LONG POINTER TO IconPictureBits ← NIL;
  open: Atom.ATOM ← Atom.null;
  oldImpl: LONG POINTER TO Containee.Implementation ← NIL;
  z: PUBLIC UNCOUNTED ZONE ← Heap.Create[initial: 32];
  true: BOOLEAN ← TRUE;
  false: BOOLEAN ← FALSE;
  shellDims: Window.Dims =                -- display size of tool
    [UserTerminal.screenWidth - 500, UserTerminal.screenHeight - 200];
  formWindowDims: Window.Dims ← [shellDims.w - 30, 30000];
  context: Context.Type ← Context.UniqueType[];

  -- MONITOR PROCs
  -- The monitor invariant is data.busy

  -- ensure that process is finished before destroying context data
  IsCloseLegal: ENTRY StarWindowShell.IsCloseLegalProc = {
    body: Window.Handle ← StarWindowShell.GetBody[sws];
    data: Defs.Data ← GetContext[body];
    IF data.busy THEN RETURN[FALSE];
    RETURN[TRUE];
  };

  -- END OF MONITOR

  -- save the current state of the dictionary and close the file.
  DestroyContext: PROC [mydata:Defs.Data, window:Window.Handle] = {
    headerPtr: FileHeaderRecPtr ← mydata.spaceBase.pointer;
    headerPtr.nextFreeNode ← mydata.nextFreeElement;
    headerPtr.lastNode ← mydata.lastElement;
    mydata.spaceBase.pointer ← Space.Unmap[mydata.spaceBase.pointer];
    NSFile.Close[mydata.handle];
    z.FREE[@mydata];
  };

  ExpandFile: PUBLIC PROCEDURE[data: Defs.Data] = {
    oldLength: CARDINAL ← CARDINAL[data.spaceBase.count];
    newLength: CARDINAL ← CARDINAL[oldLength + extensionSize];
    data.spaceBase.pointer ← Space.Unmap[data.spaceBase.pointer];
    IF newLength > maxSize THEN SIGNAL NoMoreRoom;
    NSSegment.SetSizeInPages[file: data.handle, pages: newLength];
    data.spaceBase ← NSSegment.Map[
```

```
      origin: [file: data.handle, base: 0, count: newLength],
      access: NSFile.fullAccess];
    data.lastElement ← ((newLength * Environment.wordsPerPage) -
        (SIZE[Defs.RootLetterArray] + 2)) / SIZE[Defs.Element];
    data.rootLetters ← data.spaceBase.pointer;
    data.tree ← DESCRIPTOR[
        data.spaceBase.pointer + SIZE[Defs.RootLetterArray] + 2,
        data.lastElement];
  };

  -- This procedure ensures that there is a file of the appropriate
  -- type in the prototype catalog.
  FindOrCreateIconFile: PROCEDURE = {
    name: XString.ReaderBody ← Defs.GetMessage[Defs.MessageKey.spellerName.ORD];
    version: CARDINAL = 1;
    IF (Prototype.Find [type: Defs.SpellerFileType, version: version] = NSFile.nullReference) THEN
      NSFile.Close [Prototype.Create [name: @name, type: Defs.SpellerFileType,
        version: version] ];
  };

  GenericProc: Containee.GenericProc = {
    SELECT atom FROM
      open => RETURN [MakeShell[data, changeProc, changeProcData]];
    ENDCASE => RETURN oldImpl.genericProc[atom, data, changeProc, changeProcData];
  };

  GetContext: PUBLIC PROCEDURE[body: Window.Handle] RETURNS[Defs.Data] = {
    RETURN[Context.Find[type: context, window: body]]};

  Init: PROCEDURE = {
    InitAtoms[];
    FindOrCreateIconFile[];
    InitBigPicture[];
    SetImplementation[];
  };

  InitAtoms: PROCEDURE = {
    open ← Atom.MakeAtom["Open"L];
  };

  InitBigPicture: PROCEDURE = {
    normalIconPicture ← Space.ScratchMap[1].pointer;
    normalIconPicture↑ ← [177777B, 177777B, 177777B, 177777B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 117760B, 12520B, 12520B, 17761B,
    110020B, 0B, 0B, 10021B, 111620B, 10020B, 10020B, 13621B, 113120B, 0B, 0B, 13121B, 113020B, 10020B, 10020B, 13121B, 113020B, 0B,
    100000B, 13121B, 111620B, 10022B, 50020B, 13621B, 110320B, 0B, 100000B, 13421B, 110320B, 10020B, 10020B, 13221B, 112320B, 0B, 0B,
    13121B, 111620B, 10020B, 10020B, 13121B, 110020B, 0B, 0B, 10021B, 117760B, 12520B, 12520B, 17761B, 100000B, 0B, 0B, 1B, 100400B,
    404B, 0B, 401B, 100400B, 1200B, 0B, 1601B, 100400B, 1B, 0B, 401B, 101600B, 400B, 40000B, 401B, 100400B, 0B, 140000B, 401B, 100000B,
    0B, 0B, 1B, 117760B, 12520B, 12520B, 17761B, 110020B, 0B, 0B, 10021B, 113620B, 10020B, 10020B, 13721B, 113120B, 0B, 0B, 13021B,
    113120B, 110020B, 10020B, 13021B, 113122B, 40000B, 0B, 13021B, 113620B, 110020B, 10020B, 13421B, 113020B, 0B, 0B, 13021B, 113020B,
    10020B, 10020B, 13021B, 113020B, 0B, 0B, 13021B, 113020B, 10020B, 10020B, 13721B, 110020B, 0B, 0B, 10021B, 117760B, 12520B, 12520B,
    17761B, 100000B, 0B, 0B, 1B, 100400B, 0B, 0B, 1B, 100400B, 0B, 0B, 140001B, 100400B, 0B, 0B, 140001B, 101600B, 0B, 1B, 1B, 100400B,
    0B, 2B, 1B, 100000B, 0B, 4B, 1B, 117760B, 17760B, 17760B, 12521B, 110020B, 10020B, 10020B, 1B, 113720B, 13020B, 13020B, 10021B,
    113020B, 13020B, 13020B, 1B, 113020B, 13020B, 13020B, 10021B, 113020B, 113020B, 113020B, 100001B, 113427B, 153027B, 153022B, 50021B,
    113020B, 113020B, 113020B, 100001B, 113020B, 13020B, 13020B, 10021B, 113020B, 13020B, 13020B, 1B, 113720B, 13720B, 13720B, 10021B,
    110020B, 10020B, 10020B, 1B, 117760B, 17760B, 17760B, 12521B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B,
    100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 177777B, 177777B, 177777B, 177777B];
  };

  MakeShell: PROCEDURE [data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL, changeProcData: LONG POINTER ← NIL]
    RETURNS [shell: StarWindowShell.Handle] = {
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    formWindow: Window.Handle;
    mydata: Defs.Data ← NIL;
    [name, ticket] ← Containee.GetCachedName [data];
    shell ← StarWindowShell.Create [name: @name, isCloseLegalProc: IsCloseLegal];
    Containee.ReturnTicket [ticket];
    formWindow ← StarWindowShell.CreateBody [
      sws: shell, box: [[0,0], formWindowDims]];
    Defs.MakeFormWindow[wh: formWindow];
    StarWindowShell.SetPreferredDims [shell, shellDims];
    Context.Create[
      type: context,
      data: mydata ← z.NEW[Defs.DataObject ← []],
      window: formWindow,
      proc: DestroyContext];
    OpenDictionary[data, mydata];
  };

  -- open the dictionary and set the data object pointers and indexes
  -- to the corresponding dictionary values. If the dictionary has not been
  -- opened before, extend its length and initialize its values
  OpenDictionary: PROC [data: Containee.DataHandle, mydata: Defs.Data] = {
    length: NSSegment.PageCount;
    headerPtr: FileHeaderRecPtr;
    mydata.handle ← NSFile.OpenByReference[reference: data.reference];
    length ← NSSegment.GetSizeInPages[file: mydata.handle];
    IF length < 10 THEN {
      NSSegment.SetSizeInPages[file: mydata.handle, pages: beginningPages];
      mydata.nextFreeElement ← 0;
      mydata.lastElement ←
        ((beginningPages * Environment.wordsPerPage) -
         (SIZE[Defs.RootLetterArray] + 2)) / SIZE[Defs.Element];
      mydata.spaceBase ← NSSegment.Map[
        origin: [file: mydata.handle, base: 0, count: beginningPages],
        access: NSFile.fullAccess];
```

```
      mydata.rootLetters ← mydata.spaceBase.pointer;
      FOR i: CARDINAL IN Defs.Letters DO
        mydata.rootLetters[i] ← [FALSE, Defs.lastCardinal];
      ENDLOOP;
      -- write info to file in case of a premature crash
      BEGIN
        headerPtr: FileHeaderRecPtr ← mydata.spaceBase.pointer;
        headerPtr.nextFreeNode ← mydata.nextFreeElement;
        headerPtr.lastNode ← mydata.lastElement;
        Space.ForceOut[mydata.spaceBase];
      END;
    }
    ELSE {
      mydata.spaceBase ← NSSegment.Map[
        origin: [file: mydata.handle, base: 0, count: length],
        access: NSFile.fullAccess];
      headerPtr ← mydata.spaceBase.pointer;
      mydata.nextFreeElement ← headerPtr.nextFreeNode;
      mydata.lastElement ← headerPtr.lastNode;
      mydata.rootLetters ← mydata.spaceBase.pointer;
    };
    mydata.tree ← DESCRIPTOR[
        mydata.spaceBase.pointer + SIZE[Defs.RootLetterArray] + 2,
        mydata.lastElement];
  };

  PaintIconName: PROCEDURE [window: Window.Handle, iconBox, textBox: Window.Box, name: XString.Reader] = {
    -- textBox is relative to iconBox
    -- iconBox is relative to window
    lineToLineDeltaY: CARDINAL ← SimpleTextDisplay.systemFontHeight-3;
    textBox.place ← [
      x: iconBox.place.x+textBox.place.x,
      y: iconBox.place.y+textBox.place.y];
    [] ← SimpleTextDisplay.StringIntoWindow [
      string: name,
      window: window,
      place: textBox.place,
      lineWidth: textBox.dims.w,
      maxNumberOfLines: textBox.dims.h/(lineToLineDeltaY-1),
      lineToLineDeltaY: lineToLineDeltaY,
      wordBreak: TRUE];
  };

  PictureProc: Containee.PictureProc = {
    <<[data: Containee.DataHandle, window: Window.Handle, box: Window.Box,
       old, new: Containee.PictureState]>>
    textBox: Window.Box ← [ [x: 10, y: 20], [w: 50, h: 20] ];
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    IF new=garbage THEN RETURN;
    box.dims ← [64,64];
    [name, ticket] ← Containee.GetCachedName [data];
    SELECT old FROM
      garbage, ghost => {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [normalIconPicture, 0, 0],
          flags: Display.replaceFlags]};
      highlighted =>  {
        Display.Invert [
          window: window,
          box: box]};
      ENDCASE;
    SELECT new FROM
      highlighted =>  {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [normalIconPicture, 0, 0],
          flags: Display.replaceFlags];
        Display.Invert [
          window: window,
          box: box]};
      ghost => {
        Display.White [window, box];
        PaintIconName [window, box, textBox, @name]};
      ENDCASE => {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [normalIconPicture, 0, 0],
          flags: Display.replaceFlags]};
    Containee.ReturnTicket [ticket];
  };

  SetImplementation: PROCEDURE = {
    newImpl: Containee.Implementation ← Containee.GetImplementation [Defs.SpellerFileType];
    oldImpl ← z.NEW[Containee.Implementation ← newImpl];
    newImpl.convertProc ← Containee.DefaultFileConvertProc;
    newImpl.genericProc ← GenericProc;
    newImpl.pictureProc ← PictureProc;
    newImpl.name ← Defs.GetMessage[
      Defs.MessageKey.spellerName.ORD];
```

```
    [] ← Containee.SetImplementation [Defs.SpellerFileType, newImpl];
  };

  -- Mainline code

  Init[]; -- Note that the message impl must be started first!

}...
```

```
-- SpellerImplTemplate.mesa
-- Frank        13-Aug-86 17:33:01
-- Mark         16-Mar-87 13:46:09

-- Copyright (C) Xerox Corporation 1986. All rights reserved.

-- CanITake
-- DestroyContext
-- ExpandFile
-- FindOrCreateIconFile
-- GenericProc
-- Init
-- InitAtoms
-- InitBigPicture
-- MakeShell
-- OpenDictionary
-- PaintIconName
-- PictureProc
-- SetImplementation

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Containee USING [ChangeProc, Data, DataHandle, DefaultFileConvertProc, GenericProc, GetCachedName, GetImplementation, Implementation,
  PictureProc, ReturnTicket, SetImplementation, Ticket],
  Context USING [Create, Data, Find, Type, UniqueType],
  Display USING [Bitmap, Handle, Invert, replaceFlags, White],
  Environment USING [PageCount, wordsPerPage],
  Heap USING [Create],
  NSFile USING [Close, fullAccess, Handle, nullReference, OpenByReference, Type],
  NSSegment USING [GetSizeInPages, Map, PageCount, SetSizeInPages],
  Prototype USING [Create, Find],
  SimpleTextDisplay USING [StringIntoWindow, systemFontHeight],
  Space USING [ForceOut, Map, ScratchMap, Unmap],
  SpellerDefs USING [Data, DataObject, Element, GetMessage, lastCardinal, Letters, MakeFormWindow, MessageKey, RootLetterArray,
  SpellerFileType],
  StarWindowShell USING [Create, CreateBody, Handle, SetPreferredDims],
  UserTerminal USING [screenHeight, screenWidth],
  Window USING [Box, Dims, Handle],
  XString USING [Reader, ReaderBody];

SpellerImplTemplate: PROGRAM
  IMPORTS Atom, Containee, Context, Display, Heap, NSFile, NSSegment, Prototype, SimpleTextDisplay, Space, SpellerDefs, StarWindowShell,
  UserTerminal
  EXPORTS SpellerDefs = { OPEN Defs: SpellerDefs;

  FileHeaderRec: TYPE = RECORD [
    rootLetterArray: Defs.RootLetterArray,
    nextFreeNode: CARDINAL,
    lastNode: CARDINAL];
  FileHeaderRecPtr: TYPE = LONG POINTER TO FileHeaderRec;
  beginningPages: CARDINAL = 64;
  extensionSize: CARDINAL = 64;
  maxSize: CARDINAL = 1000;
  NoMoreRoom: PUBLIC SIGNAL = CODE;

  IconPictureBits: TYPE = ARRAY [0..256) OF WORD;
  normalIconPicture: LONG POINTER TO IconPictureBits ← NIL;
  open: Atom.ATOM ← Atom.null;
  oldImpl: LONG POINTER TO Containee.Implementation ← NIL;
  z: PUBLIC UNCOUNTED ZONE ← Heap.Create[initial: 32];
  true: BOOLEAN ← TRUE;
  false: BOOLEAN ← FALSE;
  shellDims: Window.Dims =                -- display size of tool
    [UserTerminal.screenWidth - 500, UserTerminal.screenHeight - 200];
  formWindowDims: Window.Dims ← [shellDims.w - 30, 30000];
  context: Context.Type ← Context.UniqueType[];

  -- save the current state of the dictionary and close the file.
  DestroyContext: PROC [mydata:Defs.Data, window:Window.Handle] = {
    headerPtr: FileHeaderRecPtr ← mydata.spaceBase.pointer;
    headerPtr.nextFreeNode ← mydata.nextFreeElement;
    headerPtr.lastNode ← mydata.lastElement;
    mydata.spaceBase.pointer ← Space.Unmap[mydata.spaceBase.pointer];
    NSFile.Close[mydata.handle];
    z.FREE[@mydata];
  };

  ExpandFile: PUBLIC PROCEDURE[data: Defs.Data] = {
    oldLength: CARDINAL ← CARDINAL[data.spaceBase.count];
    newLength: CARDINAL ← CARDINAL[oldLength + extensionSize];
    data.spaceBase.pointer ← Space.Unmap[data.spaceBase.pointer];
    IF newLength > maxSize THEN SIGNAL NoMoreRoom;
    NSSegment.SetSizeInPages[file: data.handle, pages: newLength];
    data.spaceBase ← NSSegment.Map[
      origin: [file: data.handle, base: 0, count: newLength],
      access: NSFile.fullAccess];
    data.lastElement ← ((newLength * Environment.wordsPerPage) -
        (SIZE[Defs.RootLetterArray] + 2)) / SIZE[Defs.Element];
    data.rootLetters ← data.spaceBase.pointer;
    data.tree ← DESCRIPTOR[
        data.spaceBase.pointer + SIZE[Defs.RootLetterArray] + 2,
        data.lastElement];
  };

  -- This procedure ensures that there is a file of the appropriate
  -- type in the prototype catalog.
  FindOrCreateIconFile: PROCEDURE = {
```

```
    name: XString.ReaderBody ← Defs.GetMessage[Defs.MessageKey.spellerName.ORD];
    version: CARDINAL = 1;
    IF (Prototype.Find [type: Defs.SpellerFileType, version: version] = NSFile.nullReference) THEN
      NSFile.Close [Prototype.Create [name: @name, type: Defs.SpellerFileType,
        version: version] ];
  };

  GenericProc: Containee.GenericProc = {
    SELECT atom FROM
      open => RETURN [MakeShell[data, changeProc, changeProcData]];
    ENDCASE => RETURN oldImpl.genericProc[atom, data, changeProc, changeProcData];
  };

  GetContext: PUBLIC PROCEDURE[body: Window.Handle] RETURNS[Defs.Data] = {
    RETURN[Context.Find[type: context, window: body]]};

  Init: PROCEDURE = {
    InitAtoms[];
    FindOrCreateIconFile[];
    InitBigPicture[];
    SetImplementation[];
  };

  InitAtoms: PROCEDURE = {
    open ← Atom.MakeAtom["Open"L];
  };

  InitBigPicture: PROCEDURE = {
    normalIconPicture ← Space.ScratchMap[1].pointer;
    normalIconPicture↑ ← [177777B, 177777B, 177777B, 177777B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 117760B, 12520B, 12520B, 17761B,
    110020B, 0B, 0B, 10021B, 111620B, 10020B, 10020B, 13621B, 113120B, 0B, 0B, 13121B, 113020B, 10020B, 10020B, 13121B, 113020B, 0B,
    100000B, 13121B, 111620B, 10022B, 50020B, 13621B, 110320B, 0B, 100000B, 13421B, 110320B, 10020B, 10020B, 13221B, 112320B, 0B, 0B,
    13121B, 111620B, 10020B, 10020B, 13121B, 110020B, 0B, 0B, 10021B, 117760B, 12520B, 12520B, 17761B, 100000B, 0B, 0B, 1B, 100400B,
    404B, 0B, 401B, 100400B, 1200B, 0B, 1601B, 100400B, 1B, 0B, 401B, 101600B, 400B, 40000B, 401B, 100400B, 0B, 140000B, 401B, 100000B,
    0B, 0B, 1B, 117760B, 12520B, 12520B, 17761B, 110020B, 0B, 0B, 10021B, 113620B, 10020B, 10020B, 13721B, 113120B, 0B, 0B, 13021B,
    113120B, 110020B, 10020B, 13021B, 113122B, 40000B, 0B, 13021B, 113620B, 110020B, 10020B, 13421B, 113020B, 0B, 0B, 13021B, 113020B,
    10020B, 10020B, 13021B, 113020B, 0B, 0B, 13021B, 113020B, 10020B, 10020B, 13721B, 110020B, 0B, 0B, 10021B, 117760B, 12520B, 12520B,
    17761B, 100000B, 0B, 0B, 1B, 100400B, 0B, 0B, 1B, 100400B, 0B, 0B, 140001B, 100400B, 0B, 0B, 140001B, 101600B, 0B, 1B, 1B, 100400B,
    0B, 2B, 1B, 100000B, 0B, 4B, 1B, 117760B, 17760B, 17760B, 12521B, 110020B, 10020B, 10020B, 1B, 113720B, 13020B, 13020B, 10021B,
    113020B, 13020B, 13020B, 1B, 113020B, 13020B, 13020B, 10021B, 113020B, 113020B, 113020B, 100001B, 113427B, 153027B, 153022B, 50021B,
    113020B, 113020B, 113020B, 100001B, 113020B, 13020B, 13020B, 10021B, 113020B, 13020B, 13020B, 1B, 113720B, 13720B, 13720B, 10021B,
    110020B, 10020B, 10020B, 1B, 117760B, 17760B, 17760B, 12521B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B,
    100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 100000B, 0B, 0B, 1B, 177777B, 177777B, 177777B, 177777B];
  };

  MakeShell: PROCEDURE [data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL, changeProcData: LONG POINTER ← NIL]
    RETURNS [shell: StarWindowShell.Handle] = {
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    formWindow: Window.Handle;
    mydata: Defs.Data ← NIL;
    [name, ticket] ← Containee.GetCachedName [data];
    shell ← StarWindowShell.Create [name: @name];
    Containee.ReturnTicket [ticket];
    formWindow ← StarWindowShell.CreateBody [
      sws: shell, box: [[0,0], formWindowDims]];
    Defs.MakeFormWindow[wh: formWindow];
    StarWindowShell.SetPreferredDims [shell, shellDims];
    Context.Create[
      type: context,
      data: mydata ← z.NEW[Defs.DataObject ← []],
      window: formWindow,
      proc: DestroyContext];
    OpenDictionary[data, mydata];
  };

  -- open the dictionary and set the data object pointers and indexes
  -- to the corresponding dictionary values. If the dictionary has not been
  -- opened before, extend its length and initialize its values
  OpenDictionary: PROC [data: Containee.DataHandle, mydata: Defs.Data] = {
    length: NSSegment.PageCount;
    headerPtr: FileHeaderRecPtr;
    mydata.handle ← NSFile.OpenByReference[reference: data.reference];
    length ← NSSegment.GetSizeInPages[file: mydata.handle];
    IF length < 10 THEN {
      NSSegment.SetSizeInPages[file: mydata.handle, pages: beginningPages];
      mydata.nextFreeElement ← 0;
      mydata.lastElement ←
        ((beginningPages * Environment.wordsPerPage) -
         (SIZE[Defs.RootLetterArray] + 2)) / SIZE[Defs.Element];
      mydata.spaceBase ← NSSegment.Map[
        origin: [file: mydata.handle, base: 0, count: beginningPages],
        access: NSFile.fullAccess];
      mydata.rootLetters ← mydata.spaceBase.pointer;
      FOR i: CARDINAL IN Defs.Letters DO
        mydata.rootLetters[i] ← [FALSE, Defs.lastCardinal];
      ENDLOOP;
      -- write info to file in case of a premature crash
      BEGIN
        headerPtr: FileHeaderRecPtr ← mydata.spaceBase.pointer;
        headerPtr.nextFreeNode ← mydata.nextFreeElement;
        headerPtr.lastNode ← mydata.lastElement;
        Space.ForceOut[mydata.spaceBase];
      END;
    }
    ELSE {
```

```
      mydata.spaceBase ← NSSegment.Map[
        origin: [file: mydata.handle, base: 0, count: length],
        access: NSFile.fullAccess];
      headerPtr ← mydata.spaceBase.pointer;
      mydata.nextFreeElement ← headerPtr.nextFreeNode;
      mydata.lastElement ← headerPtr.lastNode;
      mydata.rootLetters ← mydata.spaceBase.pointer;
    };
    mydata.tree ← DESCRIPTOR[
        mydata.spaceBase.pointer + SIZE[Defs.RootLetterArray] + 2,
        mydata.lastElement];
  };

  PaintIconName: PROCEDURE [window: Window.Handle, iconBox, textBox: Window.Box, name: XString.Reader] = {
    -- textBox is relative to iconBox
    -- iconBox is relative to window
    lineToLineDeltaY: CARDINAL ← SimpleTextDisplay.systemFontHeight-3;
    textBox.place ← [
      x: iconBox.place.x+textBox.place.x,
      y: iconBox.place.y+textBox.place.y];
    [] ← SimpleTextDisplay.StringIntoWindow [
      string: name,
      window: window,
      place: textBox.place,
      lineWidth: textBox.dims.w,
      maxNumberOfLines: textBox.dims.h/(lineToLineDeltaY-1),
      lineToLineDeltaY: lineToLineDeltaY,
      wordBreak: TRUE];
  };

  PictureProc: Containee.PictureProc = {
    <<[data: Containee.DataHandle, window: Window.Handle, box: Window.Box,
       old, new: Containee.PictureState]>>
    textBox: Window.Box ← [ [x: 10, y: 20], [w: 50, h: 20] ];
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    IF new=garbage THEN RETURN;
    box.dims ← [64,64];
    [name, ticket] ← Containee.GetCachedName [data];
    SELECT old FROM
      garbage, ghost => {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [normalIconPicture, 0, 0],
          flags: Display.replaceFlags]};
      highlighted =>  {
        Display.Invert [
          window: window,
          box: box]};
      ENDCASE;
    SELECT new FROM
      highlighted =>  {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [normalIconPicture, 0, 0],
          flags: Display.replaceFlags];
        Display.Invert [
          window: window,
          box: box]};
      ghost => {
        Display.White [window, box];
        PaintIconName [window, box, textBox, @name]};
      ENDCASE => {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [normalIconPicture, 0, 0],
          flags: Display.replaceFlags]};
    Containee.ReturnTicket [ticket];
  };

  SetImplementation: PROCEDURE = {
    newImpl: Containee.Implementation ← Containee.GetImplementation [Defs.SpellerFileType];
    oldImpl ← z.NEW[Containee.Implementation ← newImpl];
    newImpl.convertProc ← Containee.DefaultFileConvertProc;
    newImpl.genericProc ← GenericProc;
    newImpl.pictureProc ← PictureProc;
    newImpl.name ← Defs.GetMessage[
      Defs.MessageKey.spellerName.ORD];
    [] ← Containee.SetImplementation [Defs.SpellerFileType, newImpl];
  };


  -- Mainline code

  Init[]; -- Note that the message impl must be started first!

}...
```

```
-- File: SpellerMsgImpl.mesa - last edit:
-- mark          16-Mar-87 11:46:04
-- Frank         13-Aug-86 14:52:05

-- Copyright (C) Xerox Corporation 1985. All rights reserved.

DIRECTORY
  SpellerDefs USING [MessageKey],
  XMessage USING [AllocateMessages, ClientData, Get, Handle, Messages, MsgEntry, MsgKey, RegisterMessages],
  XString USING [FromSTRING, ReaderBody];

SpellerMsgImpl: PROGRAM
  IMPORTS XMessage, XString
  EXPORTS SpellerDefs = {OPEN XS: XString, Defs: SpellerDefs;

  h: XMessage.Handle ← NIL;

  DeleteMessages: PROCEDURE [clientData: XMessage.ClientData] = {};

  GetMessage: PUBLIC PROCEDURE[key: XMessage.MsgKey] RETURNS [msg: XString.ReaderBody] = {RETURN[h.Get[key]]};

  Init: PROCEDURE = {
    msgArray: ARRAY Defs.MessageKey OF XMessage.MsgEntry ← [
      spellerName: [
        msgKey: Defs.MessageKey.spellerName.ORD,
        msg: XS.FromSTRING["Speller"L],
        id: 1],
      unknownAction: [
        msgKey: Defs.MessageKey.unknownAction.ORD,
        msg: XS.FromSTRING["Unknown user action."L],
        id: 2],
      noWordSpecified: [
        msgKey: Defs.MessageKey.noWordSpecified.ORD,
        msg: XS.FromSTRING["No word specified."L],
        id: 3],
      notFound: [
        msgKey: Defs.MessageKey.notFound.ORD,
        msg: XS.FromSTRING[" not found in dictionary."L],
        id: 4],
      deleted: [
        msgKey: Defs.MessageKey.deleted.ORD,
        msg: XS.FromSTRING[" deleted."L],
        id: 5],
      problemsWithDoc: [
        msgKey: Defs.MessageKey.problemsWithDoc.ORD,
        msg: XS.FromSTRING["Problem opening document."L],
        id: 6],
      couldntOpenFile: [
        msgKey: Defs.MessageKey.couldntOpenFile.ORD,
        msg: XS.FromSTRING["Could not open specified file."L],
        id: 7],
      wrongType: [
        msgKey: Defs.MessageKey.wrongType.ORD,
        msg: XS.FromSTRING["Wrong file type for this operation."L],
        id: 8],
      word: [
        msgKey: Defs.MessageKey.word.ORD,
        msg: XS.FromSTRING["Word "L],
        id: 9],
      read: [
        msgKey: Defs.MessageKey.read.ORD,
        msg: XS.FromSTRING["Read"L],
        id: 10],
      delete: [
        msgKey: Defs.MessageKey.delete.ORD,
        msg: XS.FromSTRING["Delete"L],
        id: 11],
      list: [
        msgKey: Defs.MessageKey.list.ORD,
        msg: XS.FromSTRING["List"L],
        id: 12],
      checkSpelling: [
        msgKey: Defs.MessageKey.checkSpelling.ORD,
        msg: XS.FromSTRING["Check Spelling"L],
        id: 13],
      feedback: [
        msgKey: Defs.MessageKey.feedback.ORD,
        msg: XS.FromSTRING["Feedback"L],
        id: 14],
      noMatch: [
        msgKey: Defs.MessageKey.noMatch.ORD,
        msg: XS.FromSTRING["No Matches"L],
        id: 15],
      listWords: [
        msgKey: Defs.MessageKey.listWords.ORD,
        msg: XS.FromSTRING["List of words that begin with >"L],
        id: 16],
      lessThan: [
        msgKey: Defs.MessageKey.lessThan.ORD,
        msg: XS.FromSTRING["<"L],
        id: 17],
      done: [
        msgKey: Defs.MessageKey.done.ORD,
        msg: XS.FromSTRING["Done."L],
        id: 18],
      insertionComplete: [
        msgKey: Defs.MessageKey.insertionComplete.ORD,
```

```
        msg: XS.FromSTRING["Insertion completed."L],
        id: 19],
      deletionComplete: [
        msgKey: Defs.MessageKey.deletionComplete.ORD,
        msg: XS.FromSTRING["Deletion completed."L],
        id: 20],
      checkingSpelling: [
        msgKey: Defs.MessageKey.checkingSpelling.ORD,
        msg: XS.FromSTRING["Checking the spelling..."L],
        id: 21],
      noMoreRoom: [
        msgKey: Defs.MessageKey.noMoreRoom.ORD,
        msg: XS.FromSTRING["No more room to store words."L],
        id: 22]
     ];

    messages: XMessage.Messages ← DESCRIPTOR [
      LOOPHOLE [
        msgArray,
        ARRAY [0..Defs.MessageKey.LAST.ORD] OF XMessage.MsgEntry] ];
    h ← XMessage.AllocateMessages [
      applicationName: "Speller"L,
      maxMessages: Defs.MessageKey.LAST.ORD + 1,
      clientData: NIL,
      proc: DeleteMessages ];
    XMessage.RegisterMessages[
      h: h,
      messages: messages,
      stringBodiesAreReal: FALSE];
    };

  -- Mainline code

  Init[];

  }...
```

```
-- SpellerVMImpl.mesa
-- Frank Yien    13-Aug-86 17:45:46
-- Mark Hahn     16-Mar-87 11:46:36

-- Copyright (C) Xerox Corporation 1986. All rights reserved.

-- CheckChildren
-- CheckOrInsertWord *          (* means public proc)
-- CheckSiblings
-- ChildExists
-- DeleteWord *
-- FindPattern
-- FindWord
-- IncrementNextFreeElementCounter
-- Insert
-- List *
-- ListWords
-- SiblingExists
-- TestWord
-- WriteInfo

DIRECTORY
  Attention USING [Post],
  SpellerDefs USING [Data, ExpandFile, GetMessage, lastCardinal, MessageKey, NoMoreRoom, z],
  XChar USING [Character, LowerCase],
  XFormat USING [CR, Handle, Object, Reader, WriterObject],
  XString USING [AppendChar, AppendReader, Character, CharacterLength, CopyToNewReaderBody, FreeReaderBytes, FreeWriterBytes, Lop,
  NewWriterBody, Reader, ReaderBody, ReaderFromWriter, WriterBody];

SpellerVMImpl: PROGRAM
  IMPORTS Attention, SpellerDefs, XChar, XFormat, XString
  EXPORTS SpellerDefs = { OPEN Defs: SpellerDefs;


  CheckChildren: PROC [data:Defs.Data, r:XString.Reader,
  current:CARDINAL] = {
    length: CARDINAL ← XString.CharacterLength[r];
    IF length = 0 THEN {                                                    -- If no more string then done
      data.tree[current].eow ← TRUE;
      RETURN;
    };
    IF data.tree[current].child = Defs.lastCardinal THEN {                  -- If no child & not done then create one
      data.tree[current].child ← data.nextFreeElement;
      data.tree[data.nextFreeElement] ← [XChar.LowerCase[XString.Lop[r]], FALSE,    --   Assign next char to new element
        Defs.lastCardinal, Defs.lastCardinal];
      IncrementNextFreeElementCounter[data ! Defs.NoMoreRoom => GOTO bad];  --   Bump the index to the next free element
      CheckChildren[data, r, data.tree[current].child];                     --   Do the next char
    } ELSE {                                                                -- Else if there's a child...
      c: XChar.Character ← XChar.LowerCase[XString.Lop[r]];                 --   get the next char
      IF data.tree[data.tree[current].child].ch = c THEN                    --   if the child matches then check children
        CheckChildren[data, r, data.tree[current].child]
      ELSE CheckSiblings[data, r, c, data.tree[current].child];             --    else check siblings
    };
    EXITS bad => NULL;
  };

  CheckOrInsertWord: PUBLIC PROC [data:Defs.Data, r:XString.Reader,
  checkSpelling:BOOL ← FALSE] RETURNS [correctlySpelled: BOOL ← FALSE] = {
    localRB: XString.ReaderBody ← XString.CopyToNewReaderBody[r:r, z:Defs.z];
    IF checkSpelling THEN correctlySpelled ← FindWord[data, @localRB].found
    ELSE Insert[data, @localRB];
    XString.FreeReaderBytes[@localRB, Defs.z];
    RETURN[correctlySpelled];
  };

  CheckSiblings: PROC [data:Defs.Data, r:XString.Reader, c: XChar.Character,
  current:CARDINAL] = {
    IF data.tree[current].sibling = Defs.lastCardinal THEN {                -- If no sibling then add sibling
      data.tree[current].sibling ← data.nextFreeElement;                    --   Use next free element for sibling
      data.tree[data.nextFreeElement] ←
        [c, FALSE, Defs.lastCardinal, Defs.lastCardinal];                   --   Set the sibling to the char c
      IncrementNextFreeElementCounter[data ! Defs.NoMoreRoom => GOTO bad];  --   Bump the index to the next free element
      CheckChildren[data, r, data.tree[current].sibling];                   --   Check children
    } ELSE {                                                                -- Else if there's a sibling...
      IF data.tree[data.tree[current].sibling].ch = c THEN                  --   if the sibling matches then check children
        CheckChildren[data, r, data.tree[current].sibling]
      ELSE CheckSiblings[data, r, c, data.tree[current].sibling]            --    else keep checking siblings
    };
    EXITS bad => NULL;
  };

  ChildExists: PROC[a:CARDINAL] RETURNS[BOOL] = {
    RETURN [a # Defs.lastCardinal];
  };

  DeleteWord: PUBLIC PROC [data:Defs.Data, r:XString.Reader] RETURNS [found:BOOL ← FALSE] = {
    lastCharIndex: CARDINAL;
    rootElement: BOOL;
    localRB: XString.ReaderBody ← XString.CopyToNewReaderBody[r:r, z:Defs.z];
    [found, rootElement, lastCharIndex] ← FindWord[data, @localRB];
    IF found THEN
      IF rootElement THEN data.rootLetters[lastCharIndex].eow ← FALSE
      ELSE data.tree[lastCharIndex].eow ← FALSE;
    XString.FreeReaderBytes[@localRB, Defs.z];
  };

  -- FindPattern is similar to FindWord except that it ignores eow; it simply
```

```
-- determines if a string pattern exists in the database, regardless of
-- whether that pattern is a well-formed word or not. This proc is called from
-- the List procedure.
FindPattern: PROC [data:Defs.Data, r:XString.Reader] RETURNS [found:BOOL ← FALSE,
rootElement:BOOL ← FALSE, lastCharIndex:CARDINAL ← Defs.lastCardinal] = {
  length: CARDINAL ← XString.CharacterLength[r];
  c: XChar.Character ← XChar.LowerCase[XString.Lop[r]];
  IF length = 1 THEN RETURN[TRUE,TRUE,c.ORD]                          -- Return true if a 1-letter reader is passed in
  ELSE IF ~ChildExists[data.rootLetters[c.ORD].child] THEN RETURN     -- Return false if reader has > 1 letter & root
  has no child
  ELSE {
    [found, lastCharIndex] ← TestWord[data:data,
      currentElement:data.rootLetters[c.ORD].child, r:r,
      c:XChar.LowerCase[XString.Lop[r]], needWord:FALSE];
    RETURN[found, FALSE, lastCharIndex]};
};


-- FindWord searches the database for r & returns:
-- 1. was it found?
-- 2. if found, did the word terminate in the root element (i.e., was it a
--    one-letter word)?
-- 3. if found, what was the index in the DB of the last character of that word?
-- Note: Whenever you see a RETURN in this proc with no arguments, it's just a
-- logical RETURN[found:FALSE].
FindWord: PROC [data:Defs.Data, r:XString.Reader] RETURNS [found:BOOL ← FALSE,
rootElement:BOOL ← FALSE, lastCharIndex:CARDINAL ← Defs.lastCardinal] = {
  length: CARDINAL ← XString.CharacterLength[r];
  c: XChar.Character ← XChar.LowerCase[XString.Lop[r]];
  IF length = 1 AND data.rootLetters[c.ORD].eow THEN RETURN[TRUE,TRUE,c.ORD]   -- Return true if the 1-letter reader passed in
  is a word
  ELSE IF length = 1 AND ~data.rootLetters[c.ORD].eow THEN RETURN              -- Return false if the 1-letter reader passed in
  isn't a word
  ELSE IF ~ChildExists[data.rootLetters[c.ORD].child] THEN RETURN              -- Return false if reader has > 1 letter & root
  has no child
  ELSE {
    [found, lastCharIndex] ← TestWord[data:data,
      currentElement:data.rootLetters[c.ORD].child, r:r,
      c:XChar.LowerCase[XString.Lop[r]]];
    RETURN[found, FALSE, lastCharIndex]};
};

-- IncrementNextFreeElementCounter, in the normal case, increments the
-- field data.nextFreeElement, which is the array index of the next free
-- element. If data.nextFreeElement is at the end of the array, then
-- the array and the backing file must be expanded. If there's a problem
-- in expanding, then allow error NoMoreRoom to propagate.
IncrementNextFreeElementCounter: PROC [data:Defs.Data] = {
  IF data.nextFreeElement < CARDINAL[data.tree.LENGTH - 1] THEN                -- If we're not past the end of
  the array
    data.nextFreeElement ← data.nextFreeElement + 1                            --   bump counter
  ELSE {                                                                       -- Else
    Defs.ExpandFile[data ! Defs.NoMoreRoom => {                                --   make more room; file is not allowed to grow
    larger
      rb: XString.ReaderBody ← Defs.GetMessage[Defs.MessageKey.noMoreRoom.ORD];--   expansion, then don't expand, post a msg,
      and allow
      Attention.Post[s:@rb, beep:TRUE]}];                                      --   error to propagate
    data.nextFreeElement ← data.nextFreeElement + 1;                           --   if ExpandFile worked fine, then bump counter
  };
};

Insert: PROC [data:Defs.Data, r:XString.Reader] = {
  length: CARDINAL ← XString.CharacterLength[r];                               -- Get length since there are special cases
  rootCh: XChar.Character;                                                     -- First char is matched in the root elements
  c: XChar.Character;                                                          -- the char to be tested
  IF length = 0 THEN RETURN;                                                   -- No string: return
  rootCh ← XChar.LowerCase[XString.Lop[r]];                                    -- Get root char
  IF length = 1 THEN data.rootLetters[rootCh.ORD].eow ← TRUE                   -- Simple case: 1-letter word
  ELSE {
    c ← XChar.LowerCase[XString.Lop[r]];
    IF data.rootLetters[rootCh.ORD].child = Defs.lastCardinal THEN {           -- If no child for root element, create one
      data.rootLetters[rootCh.ORD].child ← data.nextFreeElement;               --   Use next free element for child
      data.tree[data.nextFreeElement] ←
        [c, FALSE, Defs.lastCardinal, Defs.lastCardinal];                      --   Set the child to the char c
      data.nextFreeElement ← data.nextFreeElement + 1;                         --   Bump the index to the next free element
      CheckChildren[data, r, data.rootLetters[rootCh.ORD].child];              --   Check children
    } ELSE {                                                                   -- Else if there's a child...
      IF data.tree[data.rootLetters[rootCh.ORD].child].ch = c THEN
        CheckChildren[data, r, data.rootLetters[rootCh.ORD].child]             --   if the child matches then check children
      ELSE CheckSiblings[data, r, c, data.rootLetters[rootCh.ORD].child]};     --   else check siblings
  };
};

-- List is a public proc that's called when the user wants to find all
-- instances that start with a given string ("r"). List first checks the
-- database to see if r exists; if it does, then ListWords is called to
-- enumerate all valid entries in the database.
List: PUBLIC PROC [data:Defs.Data, r:XString.Reader]
RETURNS[textWB: XString.WriterBody] = {
  localRB: XString.ReaderBody ← XString.CopyToNewReaderBody[r:r, z:Defs.z];    -- Make a local copy of r since it may be lopped
  textXFO: XFormat.Object ← XFormat.WriterObject[w:@textWB];                   -- textXFO is format object for the list of
  matches
  found: BOOL ← FALSE;
  rootElement: BOOL ← FALSE;
  lastCharIndex: CARDINAL ← Defs.lastCardinal;
  [found, rootElement, lastCharIndex] ← FindPattern[data:data, r:@localRB];    -- See if the local copy (logically r) is found
  in database
```

```
    textWB ← XString.NewWriterBody[maxLength:300, z:Defs.z];                    -- As matches are found, they'll be appended to
    textWB
    XString.FreeReaderBytes[@localRB, Defs.z];                                  -- Free the local copy
    WriteInfo[xfh:@textXFO, msgKey:Defs.MessageKey.listWords.ORD, wantCR:FALSE];
    WriteInfo[xfh:@textXFO, r:r, wantCR:FALSE];
    WriteInfo[@textXFO, , Defs.MessageKey.lessThan.ORD];
    IF ~found THEN WriteInfo[@textXFO, , Defs.MessageKey.noMatch.ORD]                                   -- If not found, tell user
    ELSE {
      wb: XString.WriterBody ← XString.NewWriterBody[maxLength:100, z:Defs.z];  -- Allocate new writer to build single matches
      oldWB: XString.WriterBody;                                                -- oldWB will be a copy of wb to reflect state
      before AppendChar later
      XString.AppendReader[to:@wb, from:r];                                     -- Append r to wb
      oldWB ← wb;                                                               -- Save a copy of wb to reflect state before
      AppendChar later
      IF rootElement THEN {                                                     -- If r was 1 letter long
        IF data.rootLetters[lastCharIndex].eow THEN WriteInfo[@textXFO, r];     --   if the 1 letter is a word then print it
        IF ~ChildExists[data.rootLetters[lastCharIndex].child] THEN {           --   if no more children then done
          XString.FreeWriterBytes[@wb];
          RETURN};
        XString.AppendChar[to:@wb,                                              -- Append the child to wb
          c:data.tree[data.rootLetters[lastCharIndex].child].ch];
        ListWords[data, wb, oldWB, data.rootLetters[lastCharIndex].child, @textXFO];  -- Call ListWords to find all words that start
        with r
      } ELSE {                                                                  -- Else if r was > 1 letter
        IF data.tree[lastCharIndex].eow THEN WriteInfo[@textXFO, r];            --   if r is a word then print it
        IF ~ChildExists[data.tree[lastCharIndex].child] THEN {                  --   if no more children then done
          XString.FreeWriterBytes[@wb];
          RETURN};
        XString.AppendChar[to:@wb, c:data.tree[data.tree[lastCharIndex].child].ch];  -- Append the child to wb
        ListWords[data, wb, oldWB, data.tree[lastCharIndex].child, @textXFO];   -- Call ListWords to find all words that start
        with r
      };
      XString.FreeWriterBytes[@wb];                                             -- Free the writer bytes for the one used to
      build single matches
    };
  };


  -- ListWords is a recursive proc that's called originally from List. The
  -- basic algorithm is:
  -- 1. If the current writer is a valid word, print it.
  -- 2. If the current letter in the database has a child, then append the
  --    child to the writer, call List, and then remove the child
  --    from the writer.
  -- 3. If the current letter in the database has a sibling, then replace
  --    the current letter with the sibling and call List.
  -- wb is the WriterBody for the current text passed in, while oldWB is the
  -- WriterBody for the text before the last character was appended to the text.
  -- oldWB is needed when checking the siblings because the last letter of the
  -- current text at that point will be replaced by the sibling. Since oldWB
  -- describes the bytes before that last character was appended, we simply do
  -- an XString.AppendChar to the oldWB.
  ListWords: PROC[data:Defs.Data, wb, oldWB: XString.WriterBody,
  currentIndex: CARDINAL, textXFH:XFormat.Handle] = {
    localWB: XString.WriterBody;
    IF data.tree[currentIndex].eow THEN                                         -- If wb is a word
      WriteInfo[textXFH, XString.ReaderFromWriter[@wb]];                        --   write it to textXFH
    IF ChildExists[data.tree[currentIndex].child] THEN {                        -- If there're more letters
      localWB ← wb;
      XString.AppendChar[to:@wb, c:data.tree[data.tree[currentIndex].child].ch];  --   append the child to wb
      ListWords[data, wb, localWB, data.tree[currentIndex].child, textXFH];     --   call ListWords recursively with this new wb
    };
    IF SiblingExists[data.tree[currentIndex].sibling] THEN {                    -- If sibling exists then
      localWB ← oldWB;
      XString.AppendChar[to:@oldWB, c:data.tree[data.tree[currentIndex].sibling].ch];  --   REPLACE the child in wb w/ the sibling by
      using oldWB
      ListWords[data, oldWB, localWB, data.tree[currentIndex].sibling, textXFH];  --   call ListWords recursively with this new wb
    };
  };

  SiblingExists: PROC[a:CARDINAL] RETURNS[BOOL] = {
    RETURN [a # Defs.lastCardinal];
  };

  -- TestWord is a recursive proc called from FindWord. It checks the current
  -- element against the char passed in. If they match and that's the last
  -- character to test, return true; if they match but there're more chars to
  -- test, call TestWord with the child element and the lopped string, if a child
  -- exists. If the current element & the char passed in don't match, call
  -- TestWord with the sibling element and the SAME string(DON'T lop the string),
  -- if a sibling exists. The parameter needWord is true when searching for a
  -- word (as opposed to just a pattern); i.e., if true, then the pattern in the
  -- database must have an eow on the last character.
  TestWord: PROC[data: Defs.Data, currentElement:CARDINAL, r:XString.Reader,
  c:XChar.Character, needWord:BOOL ← TRUE] RETURNS[found:BOOL,
  index:CARDINAL ← Defs.lastCardinal] = {
    length: CARDINAL ← XString.CharacterLength[r];                              -- Get length so you know if this is the last
    element to test
    IF data.tree[currentElement].ch = c THEN {                                  -- If current element matches our char...
      IF length = 0 THEN {                                                      --   if this is the last char...
        IF needWord THEN {                                                      --     and if we're searching for words as
        opposed to patterns...
          IF data.tree[currentElement].eow THEN RETURN[TRUE, currentElement]    --       and if element is marked as end of word,
          return true
          ELSE RETURN[FALSE, currentElement]                                    --       else return false
        }
        ELSE RETURN[TRUE, currentElement];                                      --     else if we're searching for patterns,
```

```
        return true
      }
      ELSE {                                                          --   if this is not the last char...
        IF data.tree[currentElement].child = Defs.lastCardinal THEN   --     if there're no more children, return false
          RETURN[FALSE, currentElement]
        ELSE {                                                        --     else recursively call TestWord, passing
        the child,
          [found, index] ← TestWord[data:data,                        --     the lopped string, and the char that was
          lopped
            currentElement:data.tree[currentElement].child, r:r,      --     from the string
            c:XChar.LowerCase[XString.Lop[r]], needWord:needWord];
          RETURN[found, index]}}}                                     --     return the values from the recursive call
    ELSE {                                                            -- If current element doesn't match our char...
      IF data.tree[currentElement].sibling = Defs.lastCardinal THEN   --   if no siblings, return false
        RETURN[FALSE, currentElement]
      ELSE {                                                          --   else recursively call TestWord, passing the
      sibling
        [found, index] ← TestWord[data,  a.tree[currentElement].sibling,  --   and the same reader and char to be tested
           r, c, needWord];
        RETURN[found, index]}};                                       --   return the values from the recursive call
};

-- WriteInfo appends readers to a writer via XFormat. If a msg key is passed in,
-- the proc gets the reader from the msg; else, it uses the reader passed in.
WriteInfo: PROC [xfh:XFormat.Handle, r:XString.Reader ← NIL,
msgKey:CARDINAL ← Defs.lastCardinal, wantCR:BOOL ← TRUE] = {
  rb: XString.ReaderBody;
  IF msgKey # Defs.lastCardinal THEN {
    rb ← Defs.GetMessage[msgKey];
    XFormat.Reader[h:xfh, r:@rb];
  } ELSE XFormat.Reader[h:xfh, r:r];
  IF wantCR THEN XFormat.CR[xfh];
};

}...
```

ideas for transliteration table tool

-- File: TableWindows.doc - last edit:
-- Breisacher.ES 18-Dec-84 15:17:05

This doc describes two interfaces that support the creation and manipulation of a simple table in a window.

THESE INTERFACES ARE ONLY PARTIALLY IMPLEMENTED. There IS enough working to make them useful. A complete list of what's implemented and what isn't is given below.

Clients are encouraged to use the interfaces and are especially encouraged to comment on them. PLEASE let me know if you use these interfaces.

Here is a brief overview of each interface. There is a little more detail in the mesa file for each interface, including a list of potential features that could be added.

> XStringTableWindow

XStringTableWindow supports a table of "cells" each of which is backed by an XString. The storage for the strings is provided by the client. When the user edits a string, the string is copied and the edited copy is maintained by XStringTableWindow. The client can then call XStringTableWindow later to obtain the edited strings.

This interface is intended to support List Management (alias RP) tabular views, Bar Chart and Pie Chart data entry in their property sheets, and spreadsheets. Also display of JStar user dictionaries. It is NOT intended to support Star document tables (i.e. there's no attributed text, no divided columns, all ruling lines are the same, etc.).

The client calls XStringTableWindow.Create supplying a Window.Handle, the initial number of rows and columns, and a procedure that will provide a string for each cell.

XStringTableWindow takes care of all display, notification, editing, etc. It grows and shrinks rows automatically as the user edits. The client can specify row-wise or column-wise operation of the NEXT key.

The client can determine the value of a specific cell, can enumerate by row or column, and can set the value of a cell. The client can also enumerate just the changed cells. The client can reset cells to be unchanged after retrieving the changed values.

The client can add rows or columns. The client can determine the current number of rows and columns. The user can add rows or columns by "nexting" out of the last cell in the table.

> TableWindow

TableWindow is a much lower level interface than XStringTableWindow. TableWindow supports a table of cells, but the content, display, and notification handling for each cell must be provided by the client. XStringTableWindow uses TableWindow.

XStringTableWindow is intended to mostly hide the operations in TableWindow. If an operation appears in TableWindow and in XStringTableWindow, the client should ALWAYS use XStringTableWindow. There are some operations in TableWindow that the client of XStringTableWindow may properly use, such as NumberofRowsAndColumns, CellBox, GetColumnWidth.

> Installation/Usage instructions

TableWindows.df points to all the files needed.

Simply get the interfaces onto your machine, compile against them and load TableWindows.bcd before you load your application.

Here's a brief description of each file:

TableWindows.doc - this file

TableWindow.bcd/.mesa - interface

XStringTableWindow.bcd/.mesa - interface

TableWindowImpl.bcd/.mesa - impl for TableWindow

XStringTableWindowImpl.bcd/.mesa - impl for XStringTableWindow

TableWindows.bcd/.config - TableWindowImpl and XStringTableWindowImpl.

XStringTableTestTool - A sample client of XStringTableWindow.

XStringTableFile* - A couple defs and an impl which
support a simple backing file for an XStringTableWindow.
This was done just for testing.

XStringTableIconImpl - Another sample client.
Uses XStringTableFile and XStringTableWindow.

> Restrictions

CAUTION!!!!!!

MANY, MANY FEATURE OF THE INTERFACES ARE NOT IMPLEMENTED YET!!!!

Also, there has not been thorough and extensive testing. I'm only slightly interested in bug reports.

> Here is what HAS been implemented:

XStringTableWindow

-- Create, except:
-- options are ignored, except nextKeyDirection.
-- canCellChange is ignored.
-- menuItems are not returned.
-- minDimsChangeProc is never called.
-- minColumnWidth, maxColumnWidth are ignored.
-- Destroy
-- IsIt
-- HasAnyBeenChanged
-- HasBeenChanged
-- ResetChanged

-- ResetAllChanged
-- EnumerateCells
-- EnumerateChangedCells
-- LookAtCell
-- GetSelection, but it only returns selectionObject = cellContent.
-- AppendRows
-- AppendColumns

TableWindow

-- Create, except:
-- options are ignored.
-- minRowHeight, minColumnWidth, maxRowHeight, maxColumnWidth are ignored.
-- Destroy
-- IsIt
-- NumberofRowsAndColumns
-- ResolveToCell
-- CellBox
-- GetColumnWidth
-- GetRowHeight
-- SetColumnWidth
-- SetRowHeight
-- GetMinMaxInfo
-- SetMinRowHeight
-- SetMaxRowHeight
-- SetMinColumnWidth
-- SetMaxColumnWidth
-- AppendRows
-- AppendColumns

> Here is what HAS NOT been implemented:

XStringTableWindow

-- SetSelection
-- SetCellSelection
-- SetInputFocus
-- DeleteRows
-- DeleteColumns
-- InsertRows
-- InsertColumns
-- SetNextKeyDirection
-- GetOptions
-- Nothing except the text in the cells can be selected.
In particular, you can't select rows or columns or areas.
-- The user can not change column widths.
-- The user cannot copy entire rows or columns.

TableWindow

-- DeleteRows
-- DeleteColumns
-- InsertRows
-- InsertColumns
-- Invert
-- Invert* (Area, Row, Column, Cell)
-- RedisplayArea

```
-- File: TableWindows.df - last edit:
-- Breisacher.es       19-Dec-84 11:02:39

Exports [Alt:OSBU North:Xerox]<BWSHacks>4.0>DF>     ReleaseAs [Not]<Yet>

  TableWindows.df                          19-Dec-84 11:06:53 PST

Exports [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>     ReleaseAs [Not]<Yet>

  +TableWindow.bcd!1                       21-Nov-84 15:07:01 PST
  TableWindow.mesa!1                       21-Nov-84 14:46:03 PST

  +XStringTableWindow.bcd!1                21-Nov-84 15:07:50 PST
  XStringTableWindow.mesa!1                 9-Nov-84 16:26:16 PST

-- TableWindows contains TableWindowImpl and XStringTableWindowImpl.
  +TableWindows.bcd!1                      18-Dec-84 10:08:13 PST

Exports [Alt:OSBU North:Xerox]<BWSHacks>4.0>Doc>     ReleaseAs [Not]<Yet>

  +TableWindows.doc!1                      18-Dec-84 15:17:05 PST

Directory [Alt:OSBU North:Xerox]<BWSHacks>4.0>Source>TableWindows>     ReleaseAs [Not]<Yet>

  TableWindows.config!1                    13-Dec-84 13:19:34 PST

  +TableWindowImpl.bcd!1                   18-Dec-84 10:05:25 PST
  TableWindowImpl.mesa!1                   18-Dec-84 10:04:32 PST
  +XStringTableWindowImpl.bcd!1            14-Dec-84 14:44:53 PST
  XStringTableWindowImpl.mesa!1            14-Dec-84 14:44:37 PST

-- XStringTableTestTool is a simple tool invoked through the Attention window menu used for testing XStringTableWindow and TableWindow.
  +XStringTableTestTool.bcd!1              14-Dec-84 15:02:16 PST
  XStringTableTestTool.mesa!1              14-Dec-84 14:57:14 PST

-- XStringTableFile* is support for a simple backing file for an XStringTableWindow.
  +XStringTableFileFormat.bcd!1            20-Nov-84 19:35:44 PST
  XStringTableFileFormat.mesa!1             5-Oct-84 10:27:42 PDT
  +XStringTableFile.bcd!1                  20-Nov-84 19:35:58 PST
  XStringTableFile.mesa!1                  27-Sep-84 10:05:14 PDT
  +XStringTableFileImpl.bcd!1              18-Dec-84 10:08:36 PST
  XStringTableFileImpl.mesa!1              20-Nov-84 19:40:23 PST

-- XStringTableIconImpl is an icon application that uses XStringTableFile and XStringTableWindow. It creates a dummy table backing file
in the system catalog.
  +XStringTableIconImpl.bcd!1              18-Dec-84 10:09:10 PST
  XStringTableIconImpl.mesa!1              18-Dec-84 10:08:13 PST
```

```
-- File: TableWindows.config - last edit:
-- Breisacher.ES       13-Dec-84 13:19:34

TableWindows: CONFIGURATION LINKS: CODE
  IMPORTS Atom, Context, Display, Heap, SimpleTextDisplay, SimpleTextEdit, SpecialSimpleText, TIP, TIPStar, Window, XString

  EXPORTS ALL = BEGIN

  TableWindowImpl;
  XStringTableWindowImpl;
  END.
```

```
-- File: TableWindow.mesa - last edit:
-- Breisacher.ES       21-Nov-84 14:46:03

-- Copyright (C) 1984 by Xerox Corporation

DIRECTORY
  TIP USING [Results],
  Window USING [Box, Handle, Place];

TableWindow: DEFINITIONS = BEGIN

<<

This interface provides a simple tabular display in a window. When a cell needs to be displayed, TableWindow calls the CellDisplayProc.
Likewise, TableWindow passes notifications to the NotifyProc.

TableWindow handles all display and notifications for the ruling lines between rows and columns, including allowing column widths and row
heights to be adjusted by the user (if the client allows this).

TableWindow owns all the storage for the column widths and row heights, but the client can change these values at any time. Min and max
column widths and row heights are also supported.

>>

<< Other potential features: fixed scrolling rows and columns (spreadsheets), invisibility of rows and columns, scrolling units (e.g.,
row at a time)...>>

  -- Create and destroy, etc.

  Create: PROCEDURE [
    window: Window.Handle,
    columns: CARDINAL,
    rows: CARDINAL,
    cellDisplayProc: CellDisplayProc,
    notifyProc: NotifyProc,
    columnWidths: FixedOrVarying ← fixed,
    initialColumnWidths: CARDINAL ← 100,
    obtainColumnWidthProc: ObtainColumnWidthProc ← NIL,
    rowHeights: FixedOrVarying ← fixed,
    initialRowHeights: CARDINAL ← 30,
    obtainRowHeightProc: ObtainRowHeightProc ← NIL,
    rulingLinesThickness: [0..10] ← 2,
    offset: Window.Place ← [0,0],
    options: Options ← defaultOptions,
    minRowHeight, minColumnWidth: CARDINAL ← 0,
    maxRowHeight, maxColumnWidth: CARDINAL ← CARDINAL.LAST];
    -- rowHeights, columnWidths, and rulingLinesThickness are in screen dots.

  CellDisplayProc: TYPE = PROCEDURE [
    window: Window.Handle,
    box: Window.Box,
    cell: Cell];
    -- A CellDisplayProc should display the cell in box.

  NotifyProc: TYPE = PROCEDURE [
    window: Window.Handle,
    cell: Cell, -- nullCell if there's no COORDS result
    results: TIP.Results];
    -- A CellNotifyProc should respond to results as desired.

  Cell: TYPE = RECORD [row, column: CARDINAL];

  Area: TYPE = RECORD [upperLeft, lowerRight: Cell];

  FixedOrVarying: TYPE = {fixed, varying};

  ObtainColumnWidthProc: TYPE = PROCEDURE [window: Window.Handle,
    column: CARDINAL] RETURNS [width: CARDINAL];

  ObtainRowHeightProc: TYPE = PROCEDURE [window: Window.Handle,
    row: CARDINAL] RETURNS [height: CARDINAL];
    -- This proc should only be provided if row heights are
    --  varying. It will be called once (ever) for each row.
    -- TableWindow will cache the row heights returned from
    --  this procedure, so the client should not cache the
    --  row heights.
    -- If the height of a row changes, the client should call
    --  SetRowHeight.

  Options: TYPE = RECORD [
    userCanAdjustRows: BOOLEAN,
    userCanAdjustColumns: BOOLEAN];

  defaultOptions: Options = [
    userCanAdjustRows: FALSE,
    userCanAdjustColumns: TRUE];

  -- Some constants
  nullCell: Cell = [CARDINAL.LAST, CARDINAL.LAST];
  firstRow: CARDINAL = 0;
  firstColumn: CARDINAL = 0;
  lastRow: CARDINAL = CARDINAL.LAST-1;
  lastColumn:CARDINAL = CARDINAL.LAST-1;
  firstCell: Cell = [firstRow,firstColumn];
  lastCell: Cell = [lastRow, lastColumn];
  nullArea: Area = [nullCell, nullCell];
```

```
Destroy: PROCEDURE [window: Window.Handle];

IsIt: PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN];

NumberofRowsAndColumns: PROCEDURE [window: Window.Handle]
  RETURNS [rows, columns: CARDINAL];

ResolveToCell: PROCEDURE [window: Window.Handle, place: Window.Place]
  RETURNS [cell: Cell];

CellBox: PROCEDURE [window: Window.Handle, cell: Cell]
  RETURNS [box: Window.Box];

-- Row heights and column widths

GetColumnWidth: PROCEDURE [window: Window.Handle, column: CARDINAL]
  RETURNS [width: CARDINAL];

GetRowHeight: PROCEDURE [window: Window.Handle, row: CARDINAL]
  RETURNS [height: CARDINAL];

SetColumnWidth: PROCEDURE [window: Window.Handle, column: CARDINAL,
  width: CARDINAL];
  -- only valid if columnWidths are varying.

SetRowHeight: PROCEDURE [window: Window.Handle, row: CARDINAL,
  height: CARDINAL];
  -- only valid if rowHeights are varying.

GetMinMaxInfo: PROCEDURE [window: Window.Handle]
  RETURNS [minRowHeight, minColumnWidth, maxRowHeight, maxColumnWidth: CARDINAL];

SetMinRowHeight: PROCEDURE [window: Window.Handle, row: CARDINAL,
  minHeight: CARDINAL];
SetMaxRowHeight: PROCEDURE [window: Window.Handle, row: CARDINAL,
  maxHeight: CARDINAL];
SetMinColumnWidth: PROCEDURE [window: Window.Handle, column: CARDINAL,
  minWidth: CARDINAL];
SetMaxColumnWidth: PROCEDURE [window: Window.Handle, column: CARDINAL,
  maxWidth: CARDINAL];

-- Highlighting stuff

InvertObject: TYPE = {cell, row, col, area, nil};

Invert: PROCEDURE [window: Window.Handle, invertObject: InvertObject,
  upperLeft, lowerRight: Cell];

-- INLINEs for convenience

InvertArea: PROCEDURE [window: Window.Handle, area: Area] = INLINE {
    Invert [window, area, area.upperLeft, area.lowerRight]};

InvertRow: PROCEDURE [window: Window.Handle, firstRow, lastRow: CARDINAL] =
  INLINE {Invert [window, row, [firstRow,0], [lastRow,lastColumn]]};

InvertColumn: PROCEDURE [window: Window.Handle, firstColumn, lastColumn: CARDINAL] = INLINE {
  Invert [window, col, [0,firstColumn], [lastRow,lastColumn]]};

InvertCell: PROCEDURE [window: Window.Handle, cell: Cell] = INLINE {
  Invert [window, cell, cell, cell]};


<<Do we want this?:
GetInvert: PROCEDURE [window: Window.Handle]
  RETURNS [upperLeft, lowerRight: Cell, invertObject: InvertObject];
>>

-- Editing

DeleteRows: PROCEDURE [window: Window.Handle, firstRow, nRows: CARDINAL];
DeleteColumns: PROCEDURE [window: Window.Handle, firstColumn, nColumns: CARDINAL];

InsertRows: PROCEDURE [window: Window.Handle, beforeRow,
  nRows: CARDINAL];

InsertColumns: PROCEDURE [window: Window.Handle, beforeColumn,
  nColumns: CARDINAL];

AppendRows: PROCEDURE [window: Window.Handle, nRows: CARDINAL];

AppendColumns: PROCEDURE [window: Window.Handle, nColumns: CARDINAL];

-- Redisplay

RedisplayArea: PROCEDURE [window: Window.Handle, upperLeft, lowerRight: Cell];

-- Signals and errors

Error: ERROR [code: ErrorCode];

ErrorCode: TYPE = {notATableWindow, inconsistentRowColumnSizes};

END...
```

```
-- File: TableWindowImpl.mesa - last edit:
-- Breisacher.ES       18-Dec-84 10:04:32

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Context USING [Create, Data, Destroy, Find, Type, UniqueType],
  Display USING [Black],
  Heap USING [systemZone],
  TableWindow USING [Area, Cell, CellDisplayProc, defaultOptions, ErrorCode, FixedOrVarying, InvertObject, NotifyProc, nullArea,
  nullCell, ObtainColumnWidthProc, ObtainRowHeightProc, Options],
  TIP USING [NotifyProc, Results, SetTableAndNotifyProc],
  TIPStar USING [NormalTable],
  Window USING [Box, EnumerateInvalidBoxes, Handle, IntersectBoxes, InvalidateBox, nullBox, Place, SetDisplayProc, TrimBoxStickouts,
  Validate];

TableWindowImpl: MONITOR
  IMPORTS Atom, Context, Display, Heap, TIP, TIPStar, Window
  EXPORTS TableWindow = BEGIN OPEN TableWindow;

  -- TYPEs

  Data: TYPE = LONG POINTER TO DataObject;

  DataObject: TYPE = RECORD [
    window: Window.Handle ← NIL,
    rulingLinesThickness: [0..10] ← 2,
    offset: Window.Place ← [0,0],
    initialColumnWidths: CARDINAL ← 100,
    initialRowHeights: CARDINAL ← 30,
    widths: Sizes ← [fixed[size: 100]],
    heights: Sizes ← [fixed[size: 30]],
    columns, rows: CARDINAL ← 0,
    inverted: InvertData ← [nullCell,nullCell, nil],
    cellDisplayProc: CellDisplayProc ← NIL,
    notifyProc: NotifyProc ← NIL,
    obtainColumnWidthProc: ObtainColumnWidthProc ← NIL,
    obtainRowHeightProc: ObtainRowHeightProc ← NIL,
    minRowHeight, minColumnWidth: CARDINAL ← 0,
    maxRowHeight, maxColumnWidth: CARDINAL ← CARDINAL.LAST,
    options: Options ← defaultOptions];

  Sizes: TYPE = RECORD [
    variant: SELECT type: FixedOrVarying FROM
      fixed => [size: CARDINAL],
      varying => [
        sizes: LONG POINTER TO SizeSeq,
        valid: LONG POINTER TO BoolSeq]
    ENDCASE];

  SizeSeq: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF CARDINAL];

  BoolSeq: TYPE = RECORD [PACKED SEQUENCE COMPUTED CARDINAL OF BOOLEAN];

  << THIS IS BOGUS!!>>
  InvertData: TYPE = RECORD [
    upperLeft, lowerRight: Cell,
    invertObject: InvertObject];

  ResolveType: TYPE = {cell, verticalLine, horizontalLine};

  -- Constants and data

  borderThickness: INTEGER = 2; -- space between contents and ruling lines.

  context: Context.Type ← Context.UniqueType[];
  sysZ: UNCOUNTED ZONE ← Heap.systemZone;

  Error: PUBLIC ERROR [code: ErrorCode] = CODE;

  pointUp, pointDown, pointMotion: Atom.ATOM ← Atom.null;

  -- Procedures

  AppendColumns: PUBLIC PROCEDURE [window: Window.Handle, nColumns: CARDINAL] = {
    data: Data = GetContext [window];
    x,y,w,h: INTEGER ← 0;
    AppendColumnWidths [data, nColumns];
    [x,y] ← data.offset;
    w ← (2*borderThickness);
    h ← SumOfRowHeights [data] + (data.rows*2*borderThickness) + ((data.rows+1)*data.rulingLinesThickness);
    FOR c: CARDINAL IN [0..data.columns) DO
      x ← x + GetColumnWidthInternal [data, c] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    FOR c: CARDINAL IN [data.columns..data.columns + nColumns) DO
      w ← w + GetColumnWidthInternal [data, c] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    data.columns ← data.columns + nColumns;
    Window.InvalidateBox [ window, [[x,y],[w,h]] ];
    Window.Validate [window];
    };

  AppendColumnWidths: PROCEDURE [data: Data, nColumns: CARDINAL] = {
    WITH widths: data.widths SELECT FROM
      varying => {
        temp: Sizes.varying;
        temp.sizes ← sysZ.NEW [ SizeSeq [data.columns + nColumns] ];
        temp.valid ← sysZ.NEW [ BoolSeq [data.columns + nColumns] ];
```

```
        FOR i: CARDINAL IN [0..data.columns) DO
          temp.sizes[i] ← widths.sizes[i];
          temp.valid[i] ← widths.valid[i];
          ENDLOOP;
        FOR i: CARDINAL IN [data.columns..data.columns + nColumns) DO
          temp.sizes[i] ← data.initialColumnWidths + (2*borderThickness);
          temp.valid[i] ← FALSE;
          ENDLOOP;
        sysZ.FREE [@widths.sizes];
        sysZ.FREE [@widths.valid];
        widths ← temp;
        };
      fixed => NULL;
      ENDCASE;
    };

  AppendRows: PUBLIC PROCEDURE [window: Window.Handle, nRows: CARDINAL] = {
    data: Data = GetContext [window];
    x,y,w,h: INTEGER ← 0;
    AppendRowHeights [data, nRows];
    [x,y] ← data.offset;
    h ← (2*borderThickness);
    w ← SumOfColumnWidths [data] + (data.columns*2*borderThickness) + ((data.columns+1)*data.rulingLinesThickness);
    FOR r: CARDINAL IN [0..data.rows) DO
      y ← y + GetRowHeightInternal [data, r] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    FOR r: CARDINAL IN [data.rows..data.rows + nRows) DO
      h ← h + GetRowHeightInternal [data, r] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    data.rows ← data.rows + nRows;
    Window.InvalidateBox [ window, [[x,y],[w,h]] ];
    Window.Validate [window];
    };

  AppendRowHeights: PROCEDURE [data: Data, nRows: CARDINAL] = {
    WITH heights: data.heights SELECT FROM
      varying => {
        temp: Sizes.varying;
        temp.sizes ← sysZ.NEW [ SizeSeq [data.rows + nRows] ];
        temp.valid ← sysZ.NEW [ BoolSeq [data.rows + nRows] ];
        FOR i: CARDINAL IN [0..data.rows) DO
          temp.sizes[i] ← heights.sizes[i];
          temp.valid[i] ← heights.valid[i];
          ENDLOOP;
        FOR i: CARDINAL IN [data.rows..data.rows + nRows) DO
          temp.sizes[i] ← data.initialRowHeights + (2*borderThickness);
          temp.valid[i] ← FALSE;
          ENDLOOP;
        sysZ.FREE [@heights.sizes];
        sysZ.FREE [@heights.valid];
        heights ← temp;
        };
      fixed => NULL;
      ENDCASE;
    };

  CellBox: PUBLIC PROCEDURE [window: Window.Handle, cell: Cell]
    RETURNS [box: Window.Box] = {
    data: Data = GetContext [window];
    x,y,w,h: INTEGER ← 0;
    h ← GetRowHeightInternal [data, cell.row];
    w ← GetColumnWidthInternal [data, cell.column];

    y ← data.rulingLinesThickness + borderThickness + data.offset.y;
    x ← data.rulingLinesThickness + borderThickness + data.offset.x;
    FOR r: CARDINAL IN [0..cell.row) DO
      y ← y + GetRowHeightInternal [data, r] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    FOR c: CARDINAL IN [0..cell.column) DO
      x ← x + GetColumnWidthInternal [data,c] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;

    RETURN [ [[x,y], [w,h]] ];
    };

  Create: PUBLIC PROCEDURE [
    window: Window.Handle,
    columns: CARDINAL,
    rows: CARDINAL,
    cellDisplayProc: CellDisplayProc,
    notifyProc: NotifyProc,
    columnWidths: FixedOrVarying ← fixed,
    initialColumnWidths: CARDINAL ← 100,
    obtainColumnWidthProc: ObtainColumnWidthProc ← NIL,
    rowHeights: FixedOrVarying ← fixed,
    initialRowHeights: CARDINAL ← 30,
    obtainRowHeightProc: ObtainRowHeightProc ← NIL,
    rulingLinesThickness: [0..10] ← 2,
    offset: Window.Place ← [0,0],
    options: Options ← defaultOptions,
    minRowHeight, minColumnWidth: CARDINAL ← 0,
    maxRowHeight, maxColumnWidth: CARDINAL ← CARDINAL.LAST] = {

    data: Data ← sysZ.NEW [DataObject ← [
      window: window,
      columns: columns,
      rows: rows,
```

```
      cellDisplayProc: cellDisplayProc,
      notifyProc: notifyProc,
      obtainColumnWidthProc: obtainColumnWidthProc,
      obtainRowHeightProc: obtainRowHeightProc,
      offset: offset,
      rulingLinesThickness: rulingLinesThickness,
      initialColumnWidths: initialColumnWidths,
      initialRowHeights: initialRowHeights,
      minRowHeight: minRowHeight,
      minColumnWidth: minColumnWidth,
      maxRowHeight: maxRowHeight,
      maxColumnWidth: maxColumnWidth,
      options: options ]];

    SELECT columnWidths FROM
      fixed => data.widths ← [fixed [initialColumnWidths + (2*borderThickness)]];
      varying => {
        data.widths ← [varying[sizes: sysZ.NEW [ SizeSeq [columns] ],
          valid: sysZ.NEW [ BoolSeq [columns] ] ]];
        FOR i: CARDINAL IN [0..columns) DO
          WITH widths: data.widths SELECT FROM
            varying => {
              widths.sizes[i] ← initialColumnWidths + (2*borderThickness);
              widths.valid[i] ← FALSE};
            ENDCASE;
          ENDLOOP;
        };
      ENDCASE;

    SELECT rowHeights FROM
      fixed => data.heights ← [fixed [initialRowHeights + (2*borderThickness)]];
      varying => {
        data.heights ← [varying[sizes: sysZ.NEW [ SizeSeq [rows] ],
          valid: sysZ.NEW [ BoolSeq [rows] ] ]];
        FOR i: CARDINAL IN [0..rows) DO
          WITH heights: data.heights SELECT FROM
            varying => {
              heights.sizes[i] ← initialRowHeights + (2*borderThickness);
              heights.valid[i] ← FALSE};
            ENDCASE;
          ENDLOOP;
        };
      ENDCASE;

    Context.Create [context, data, DestroyContext, window];
    [] ← Window.SetDisplayProc [window, Repaint];
    TIP.SetTableAndNotifyProc [window: window, table: TIPStar.NormalTable[], notify: Notify]
    };

  Destroy: PUBLIC PROCEDURE [window: Window.Handle] = {
    Context.Destroy [context, window];
    [] ← Window.SetDisplayProc [window, NIL];
    };

  DestroyContext: PROC [data: Data, window: Window.Handle] = {
    WITH dcw: data.widths SELECT FROM
      varying => {sysZ.FREE [@dcw.sizes]; sysZ.FREE [@dcw.valid]};
      ENDCASE;
    WITH drh: data.heights SELECT FROM
      varying => {sysZ.FREE [@drh.sizes]; sysZ.FREE [@drh.valid]};
      ENDCASE;
    sysZ.FREE [@data];
    };

  GetColumnWidth: PUBLIC PROCEDURE [window: Window.Handle, column: CARDINAL]
    RETURNS [width: CARDINAL] = {
    data: Data = GetContext [window];
    RETURN [ GetColumnWidthInternal [data, column] - (2*borderThickness) ];
    };

  GetColumnWidthInternal: PROCEDURE [data: Data, column: CARDINAL]
    RETURNS [width: CARDINAL] = {
    WITH dcw: data.widths SELECT FROM
      varying => IF dcw.valid[column] OR data.obtainColumnWidthProc = NIL
        THEN width ← dcw.sizes[column]
        ELSE {
          width ← dcw.sizes[column] ← data.obtainColumnWidthProc [
            data.window, column];
          dcw.valid[column] ← TRUE};
      fixed => width ← dcw.size;
      ENDCASE;
    };

  GetContext: PROC [body: Window.Handle] RETURNS [data: Data] = {
    data ← Context.Find[context, body];
    IF data = NIL THEN ERROR Error [notATableWindow];
    };

  GetMinMaxInfo: PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [minRowHeight, minColumnWidth, maxRowHeight, maxColumnWidth: CARDINAL] = {
    data: Data = GetContext [window];
    RETURN [data.minRowHeight, data.minColumnWidth, data.maxRowHeight, data.maxColumnWidth];
    };

  GetRowHeight: PUBLIC PROCEDURE [window: Window.Handle, row: CARDINAL]
    RETURNS [height: CARDINAL] = {
    data: Data = GetContext [window];
```

```
    RETURN [ GetRowHeightInternal [data, row] ];
    };

  GetRowHeightInternal: PROCEDURE [data: Data, row: CARDINAL]
    RETURNS [height: CARDINAL] = {
    WITH drh: data.heights SELECT FROM
      varying => IF drh.valid[row] OR data.obtainRowHeightProc = NIL
        THEN height ← drh.sizes[row]
        ELSE {
          height ← drh.sizes[row] ← data.obtainRowHeightProc [
            data.window, row];
          drh.valid[row] ← TRUE};
      fixed => height ← drh.size;
      ENDCASE;
    };

  InitAtoms: PROC = {
    pointUp ← Atom.MakeAtom["PointUp"L];
    pointDown ← Atom.MakeAtom["PointDown"L];
    pointMotion ← Atom.MakeAtom["PointMotion"L];
    };

  IsIt: PUBLIC PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN] =
    {RETURN [ Context.Find[context, window] # NIL ]};

  Notify: TIP.NotifyProc = {
    <<[window: Window.Handle, results: TIP.Results]>>
    -- Nothing in here about ruling lines yet.
    noPlace: Window.Place = [-1,-1];
    data: Data = GetContext [window];
    place: Window.Place ← noPlace;
    cell: Cell;
    resolveType: ResolveType ← cell;
    FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
      WITH z: input SELECT FROM
        coords => place ← z.place;
        atom => {
          IF place = noPlace THEN cell ← nullCell
          ELSE [cell, resolveType] ← Resolve[window, place, data];
          SELECT resolveType FROM
            cell => data.notifyProc [window, cell, results];
            verticalLine => NULL; << move the line??? >>
            horizontalLine => NULL; << move the line??? >>
            ENDCASE;
          EXIT;
          };
        string => {
          cell ← nullCell;
          data.notifyProc [window, cell, results];
          EXIT;
          };
        ENDCASE;
      ENDLOOP;
    };

  NumberofRowsAndColumns: PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [rows, columns: CARDINAL] = {
    data: Data = GetContext [window];
    RETURN [data.rows, data.columns];
    };

  Repaint: PROCEDURE [window: Window.Handle] = {
    data: Data = GetContext [window];

    x,y,w,h: INTEGER ← 0;

    upperLeft: Window.Place ← [INTEGER.LAST, INTEGER.LAST];
    lowerRight: Window.Place ← [0, 0];
    boxToPaint: Window.Box ← Window.nullBox;
    cellHasBeenPainted: BOOLEAN ← FALSE;
    areaPainted: Area ← nullArea;
    boxPainted: Window.Box ← Window.nullBox;

    EachInvalidBox: PROCEDURE [w: Window.Handle, box: Window.Box] = {
      upperLeft.x ← MIN [upperLeft.x, box.place.x];
      upperLeft.y ← MIN [upperLeft.y, box.place.y];
      lowerRight.x ← MAX [lowerRight.x, box.place.x + box.dims.w];
      lowerRight.y ← MAX [lowerRight.y, box.place.y + box.dims.h];
      };

    DrawLineIfValid: PROCEDURE [line: Window.Box] = {
      IF Window.IntersectBoxes [ line, boxToPaint ].dims # [0,0] THEN
        Display.Black [window, line];
      };

    -- Start of code for Repaint

    Window.EnumerateInvalidBoxes [window, EachInvalidBox];

    boxToPaint ← [upperLeft, [lowerRight.x-upperLeft.x, lowerRight.y-upperLeft.y]];
    boxToPaint ← Window.TrimBoxStickouts [window, boxToPaint];

    -- This still isn't as smart as it could be.
    --  We go through the whole table, we just don't
    --  actually paint into the cells that aren't invalid.

    -- Repaint contents
```

```
    y ← data.rulingLinesThickness + borderThickness + data.offset.y;
    FOR r: CARDINAL IN [0..data.rows) DO
      x ← data.rulingLinesThickness + borderThickness + data.offset.x;
      h ← GetRowHeightInternal [data, r];
      FOR c: CARDINAL IN [0..data.columns) DO
        w ← GetColumnWidthInternal [data, c];
        IF Window.IntersectBoxes [ [[x,y],[w,h]], boxToPaint ].dims # [0,0] THEN {
          IF ~cellHasBeenPainted THEN {
            areaPainted.upperLeft ← [r,c];
            boxPainted.place ← [
              x - data.rulingLinesThickness - borderThickness,
              y - data.rulingLinesThickness - borderThickness]};
          cellHasBeenPainted ← TRUE;
          data.cellDisplayProc [ window, [[x,y],[w,h]], [r,c] ];
          areaPainted.lowerRight ← [r,c]};
        x ← x + w + (2*borderThickness) + data.rulingLinesThickness;
        IF x > lowerRight.x THEN EXIT;
        ENDLOOP;
      boxPainted.dims.w ← x - (borderThickness + data.offset.x);
      y ← y + h + (2*borderThickness) + data.rulingLinesThickness;
      IF y > lowerRight.y THEN EXIT;
      ENDLOOP;
    boxPainted.dims.h ← y - (borderThickness + data.offset.y);

    -- Repaint ruling lines
    IF areaPainted = nullArea THEN RETURN;
    -- Horizontal
    [x,y] ← boxPainted.place;
    w ← boxPainted.dims.w;
    h ← data.rulingLinesThickness;
    FOR i: CARDINAL IN [areaPainted.upperLeft.row..areaPainted.lowerRight.row] DO
      DrawLineIfValid [ [[x,y],[w,h]] ];
      y ← y + GetRowHeightInternal [data, i] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    DrawLineIfValid [ [[x,y],[w,h]] ]; -- last line.
    -- Vertical
    [x,y] ← boxPainted.place;
    h ← boxPainted.dims.h;
    w ← data.rulingLinesThickness;
    FOR i: CARDINAL IN [areaPainted.upperLeft.column..areaPainted.lowerRight.column] DO
      DrawLineIfValid [ [[x,y],[w,h]] ];
      x ← x + GetColumnWidthInternal [data, i] + (2*borderThickness) + data.rulingLinesThickness;
      ENDLOOP;
    DrawLineIfValid [ [[x,y],[w,h]] ]; -- last line.
    };

  Resolve: PROCEDURE [window: Window.Handle, place: Window.Place,
    data: Data]
    RETURNS [cell: Cell, resolveType: ResolveType ← cell] = {
    <<Notice there's nothing here about ruling lines yet.>>
    x,y: INTEGER ← 0;
    x ← data.offset.x;
    y ← data.offset.y;
    cell ← nullCell;
    FOR i: CARDINAL IN [0..data.rows) DO
      y ← y + data.rulingLinesThickness;
      IF y > place.y THEN {
        cell.row ← i;
        resolveType ← horizontalLine;
        EXIT};
      y ← y + GetRowHeightInternal [data, i] + (2*borderThickness);
      IF y > place.y THEN {cell.row ← i;EXIT};
      ENDLOOP;
    FOR i: CARDINAL IN [0..data.columns) DO
      x ← x + data.rulingLinesThickness;
      IF x > place.x THEN {
        cell.column ← i;
        resolveType ← verticalLine;
        EXIT};
      x ← x + GetColumnWidthInternal [data, i] + (2*borderThickness);
      IF x > place.x THEN {cell.column ← i;EXIT};
      ENDLOOP;
    IF cell.row = nullCell.row OR cell.column = nullCell.column THEN cell ← nullCell;
    };

  ResolveToCell: PUBLIC PROCEDURE [window: Window.Handle, place: Window.Place]
    RETURNS [cell: Cell] = {
    data: Data = GetContext [window];
    RETURN [ Resolve [window, place, data].cell ];
    };

  SetColumnWidth: PUBLIC PROCEDURE [window: Window.Handle, column: CARDINAL,
    width: CARDINAL] = {
    data: Data = GetContext [window];
    WITH dcw: data.widths SELECT FROM
      varying => dcw.sizes[column] ← width + (2*borderThickness);
      fixed => ERROR Error [inconsistentRowColumnSizes];
      ENDCASE;
    };

  SetMaxColumnWidth: PUBLIC PROCEDURE [window: Window.Handle, column: CARDINAL,
    maxWidth: CARDINAL] = {
    data: Data = GetContext [window];
    data.maxColumnWidth ← maxWidth;
    };

  SetMaxRowHeight: PUBLIC PROCEDURE [window: Window.Handle, row: CARDINAL,
```

```
    maxHeight: CARDINAL] = {
    data: Data = GetContext [window];
    data.maxRowHeight ← maxHeight;
    };

  SetMinColumnWidth: PUBLIC PROCEDURE [window: Window.Handle, column: CARDINAL,
    minWidth: CARDINAL] = {
    data: Data = GetContext [window];
    data.minColumnWidth ← minWidth;
    };

  SetMinRowHeight: PUBLIC PROCEDURE [window: Window.Handle, row: CARDINAL,
    minHeight: CARDINAL] = {
    data: Data = GetContext [window];
    data.minRowHeight ← minHeight;
    };

  SetRowHeight: PUBLIC PROCEDURE [window: Window.Handle, row: CARDINAL,
    height: CARDINAL] = {
    data: Data = GetContext [window];
    WITH drh: data.heights SELECT FROM
      varying => drh.sizes[row] ← height;
      fixed => ERROR Error [inconsistentRowColumnSizes];
      ENDCASE;
    };

  SumOfColumnWidths: PROCEDURE [data: Data] RETURNS [width: INTEGER ← 0] = {
    WITH dcw: data.widths SELECT FROM
      varying => FOR i: CARDINAL IN [0..data.columns) DO
        width ← width + dcw.sizes[i];
        ENDLOOP;
      fixed => width ← dcw.size * data.columns;
      ENDCASE;
    };

  SumOfRowHeights: PROCEDURE [data: Data] RETURNS [height: INTEGER ← 0] = {
    WITH drh: data.heights SELECT FROM
      varying => FOR i: CARDINAL IN [0..data.rows) DO
        height ← height + drh.sizes[i];
        ENDLOOP;
      fixed => height ← drh.size * data.rows;
      ENDCASE;
    };

  -- Main line code

  InitAtoms[];

  END.
```

```
-- File: XStringTableFile.mesa - last edit:
-- Breisacher.ES       27-Sep-84 10:05:14

DIRECTORY
  NSFile USING [Handle, Reference],
  XStringTableFileFormat USING [Table];

XStringTableFile: DEFINITIONS = BEGIN

  Open: PROCEDURE [file: NSFile.Reference]
    RETURNS [table: XStringTableFileFormat.Table, fh: NSFile.Handle];

  Create: PROCEDURE RETURNS [table: XStringTableFileFormat.Table, fh: NSFile.Handle];

  END...
```

```
-- File: XStringTableFileFormat.mesa - last edit:
-- Breisacher.ES        5-Oct-84 10:27:42

DIRECTORY
  NSFile USING [Type],
  XString USING [ByteSequence];

XStringTableFileFormat: DEFINITIONS = BEGIN

  Table: TYPE = LONG BASE POINTER TO TableHeader;

  TableHeader: TYPE = RECORD [
    rowCt: CARDINAL,
    colCt: CARDINAL,
    elements: Elements,
    text: Text];

  Elements: TYPE = Table RELATIVE POINTER[0..177777B] TO ARRAY [0..0) OF CARDINAL;
  -- These are offsets to the END+1 of the string.
  -- The beginning of the string is the offset of the previous string.
  nullElements: Elements = LOOPHOLE[0];

<<
  The string for element [row, col] is:
    index: CARDINAL = row * table.colCt + col;
    offset: CARDINAL =
      IF index = 0 THEN 0 ELSE table[table.elements][index-1];
    limit: CARDINAL = table[table.elements][index];
    block: Environment.Block ← [LOOPHOLE [@table[table.text]], offset, limit];
>>


  Text: TYPE = Table RELATIVE POINTER[0..177777B] TO XString.ByteSequence;
  nullText: Text = LOOPHOLE[0];

  xStringTableFileType: NSFile.Type = 21346;

  END...
```

```
-- File: XStringTableFileImpl.mesa - last edit:
-- Breisacher.ES       20-Nov-84 19:40:23

DIRECTORY
  Environment,
  NSFile,
  NSSegment,
  Prototype,
  Space,
  XString,
  XStringTableFile,
  XStringTableFileFormat;

XStringTableFileImpl: PROGRAM
  IMPORTS NSFile, NSSegment, Prototype, Space, XString
  EXPORTS XStringTableFile = BEGIN

  -- TYPEs

  -- Procedures

  Open: PUBLIC PROCEDURE [file: NSFile.Reference]
    RETURNS [table: XStringTableFileFormat.Table, fh: NSFile.Handle] = {
    fileType: NSFile.Type;
    fh ← NSFile.OpenByReference [file];
    fileType ← NSFile.GetType [fh];
    IF fileType # XStringTableFileFormat.xStringTableFileType THEN {
      NSFile.Close [fh];
      RETURN [NIL, NSFile.nullHandle];
      };
    table ← NSSegment.Map [origin:[
      file: fh, base: 0, count: NSSegment.GetSizeInPages [fh]] ].pointer;
    };

  Create: PUBLIC PROCEDURE RETURNS [table: XStringTableFileFormat.Table, fh: NSFile.Handle] = {
    name: XString.ReaderBody ← XString.FromSTRING["XString Table Icon"L];
    version: CARDINAL = 11204;
    ref: NSFile.Reference;
    IF (ref ← Prototype.Find [
        type: XStringTableFileFormat.xStringTableFileType,
        version: version]) # NSFile.nullReference THEN
      fh ← NSFile.OpenByReference [ref]
    ELSE
      fh ← Prototype.Create [name: @name,
        type: XStringTableFileFormat.xStringTableFileType,
        version: version, size: filePages*Environment.bytesPerPage];
    table ← MakeDummyTable[fh];
    };

  filePages: CARDINAL = 10;

  MakeDummyTable: PROCEDURE [fh: NSFile.Handle] RETURNS [table: XStringTableFileFormat.Table] = {
    OPEN XStringTableFileFormat;
    rowCt: CARDINAL = 50;
    colCt: CARDINAL = 10;
    elementsOffset: Elements ← Elements.FIRST + SIZE [TableHeader];
    textOffset: Text ← Text.FIRST + SIZE [TableHeader] + (rowCt * colCt);

    table ← NSSegment.Map [origin: [
      file: fh, base: 0, count: NSSegment.GetSizeInPages [fh]],
      access:  NSFile.fullAccess].pointer;
    table↑ ← [
      rowCt: rowCt,
      colCt: colCt,
      elements: elementsOffset,
      text: textOffset ];

    BEGIN
    block: Environment.Block ← [LOOPHOLE [@table[textOffset]], 0, Environment.bytesPerPage * filePages];
    wb: XString.WriterBody ← XString.WriterBodyFromBlock [block];
    byte: CARDINAL ← 0;
    FOR i: CARDINAL IN [0..rowCt*colCt) DO
      byte ← byte + AppendSTRING [to: @wb, from: "Test"L];
      table[table.elements][i] ← byte;
      ENDLOOP;
    END;

    Space.ForceOut [[table, filePages]];
    };

    AppendSTRING: PROCEDURE [to: XString.Writer, from: LONG STRING]
      RETURNS [bytes: CARDINAL] = {
      XString.AppendSTRING [to: to, from: from];
      RETURN [from.length];
      };

  -- Mainline code

  END...


  table[table.elements][i]
```

```
-- File: XStringTableIconImpl.mesa - last edit:
-- Breisacher.ES       18-Dec-84 10:08:13

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Attention USING [Post],
  Containee USING [ChangeProc, DataHandle, DefaultFileConvertProc, GenericProc, GetCachedName, GetImplementation, Implementation,
  PictureProc, ReturnTicket, SetImplementation, SmallPictureProc, Ticket],
  Context USING [Create, Data, Find, Type, UniqueType],
  Courier USING [Error],
  Display USING [Bitmap, Handle, Invert, replaceFlags, White],
  Environment USING [Block],
  Heap USING [systemZone],
  MenuData,
  NSFile USING [Attribute, AttributesRecord, ChangeAttributes, ClearAttributes, Close, Error, GetAttributes, Handle, nullHandle,
  OpenByReference, Reference, Type],
  NSFileStream USING [Create, GetLength, Handle, SetLength],
  NSSegment USING [SetSizeInBytes],
  Selection USING [CanYouConvert, Convert, ConvertProc, Enumerate, EnumerationProc, Free],
  SimpleTextDisplay USING [StringIntoWindow, systemFontHeight],
  SimpleTextFont USING [AddClientDefinedCharacter],
  Space USING [ScratchMap],
  Stream USING [Delete, GetPosition, SetPosition],
  StarWindowShell,
  Window,
  XChar USING [null],
  XFormat USING [NSString, Object, Reader, StreamObject],
  XString USING [Character, FromBlock, FromSTRING, nullReaderBody, Reader, ReaderBody],
  XStringTableFile,
  XStringTableFileFormat,
  XStringTableWindow,
  XToken USING [Filtered, FilterProcType, FreeTokenString, FreeStreamHandle, Handle, StreamToHandle];

XStringTableIconImpl: PROGRAM
  IMPORTS Atom, Attention, Containee, Context, Courier, Display, Heap, MenuData, NSFile, NSFileStream, NSSegment, Selection,
  SimpleTextDisplay, SimpleTextFont, Space, StarWindowShell, Stream, Window, XFormat, XString, XStringTableFile, XStringTableWindow,
  XToken = BEGIN

  -- TYPEs

  IconPictureBits: TYPE = ARRAY [0..256) OF WORD;

  -- Constants and data

  context: Context.Type = Context.UniqueType[];

  oldImpl, newImpl: Containee.Implementation ← [];

  sampleIconPicture: LONG POINTER TO IconPictureBits ← NIL;

  open,
  props,
  canYouTakeSelection,
  takeSelection,
  takeSelectionCopy,
  starLogon: Atom.ATOM ← Atom.null;

  true: BOOLEAN ← TRUE;
  false: BOOLEAN ← FALSE;

  sysZ: UNCOUNTED ZONE ← Heap.systemZone;

  bodyWindowDims: Window.Dims = [30000, 30000]; -- arbitrary

  -- Procedures

  CanITake: PROCEDURE RETURNS [yes: BOOLEAN] = {RETURN[FALSE]};

  CellContent: XStringTableWindow.CellContentProc = {
    <<[window: Window.Handle, cell: XStringTableWindow.Cell,
       clientData: LONG POINTER, callBack: PROCEDURE [XString.Reader]]>>
    table: XStringTableFileFormat.Table = clientData;
    index: CARDINAL = cell.row * table.colCt + cell.column;
    block: Environment.Block ← [
      blockPointer: LOOPHOLE [@table[table.text]],
      startIndex: IF index = 0 THEN 0 ELSE table[table.elements][index-1],
      stopIndexPlusOne: table[table.elements][index] ];
    rb: XString.ReaderBody ← XString.FromBlock [block: block];
    IF cell.column < table.colCt AND cell.row < table.rowCt THEN
      callBack [@rb];
    };

  DestroyContext: PROCEDURE [fileHandle: LONG POINTER, window: Window.Handle] = {
    NSFile.Close [LOOPHOLE [fileHandle]];
    };

  FindOrCreateIconFile: PROCEDURE = {
    NSFile.Close [XStringTableFile.Create [].fh];
    };

  GenericProc: Containee.GenericProc = {
    SELECT atom FROM
      canYouTakeSelection => RETURN [
        IF CanITake[] THEN @true ELSE @false];
      takeSelection, -- we treat MOVE and COPY the same
      takeSelectionCopy => {RETURN [@false]};
      open => RETURN [ MakeShell[data, changeProc, changeProcData] ];
```

```
      ENDCASE => RETURN oldImpl.genericProc [atom, data, changeProc, changeProcData];
    };

  InitAtoms: PROCEDURE = {
    -- This gets all the strings out of the global frame.
    open ← Atom.MakeAtom["Open"L];
    props ← Atom.MakeAtom["Props"L];
    canYouTakeSelection ← Atom.MakeAtom["CanYouTakeSelection"L];
    takeSelection ← Atom.MakeAtom["TakeSelection"L];
    takeSelectionCopy ← Atom.MakeAtom["TakeSelectionCopy"L];
    starLogon ← Atom.MakeAtom["StarLogon"L];
    };

  InitBigPicture: PROCEDURE = {
    sampleIconPicture ← Space.ScratchMap[1].pointer;
    sampleIconPicture↑ ← [
      177777B, 177777B, 177777B, 177777B, 177777B, 177777B, 177777B,
      177777B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 177777B, 177777B, 177777B, 177777B, 177777B,
      177777B, 177777B, 177777B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 146000B, 000000B, 000000B, 000063B, 146000B, 000000B,
      000000B, 000063B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 146000B, 000000B, 000000B, 000063B,
      146000B, 000000B, 000000B, 000063B, 146000B, 000000B, 000000B,
      000063B, 177777B, 177777B, 177777B, 177777B, 177777B, 177777B,
      177777B, 177777B, 146000B, 000000B, 000000B, 000063B, 146000B,
      000000B, 000000B, 000063B, 177777B, 177777B, 177777B, 177777B,
      177777B, 177777B, 177777B, 177777B];
    };

  InitSmallPicture: PROCEDURE RETURNS [XString.Character] = {
    bits: ARRAY [0..13) OF WORD ← [
      177770B, 120050B, 177770B, 120050B, 120050B, 120050B,
      120050B, 120050B, 120050B, 120050B, 177770B, 120050B, 177770B];
    RETURN [ SimpleTextFont.AddClientDefinedCharacter [
      width: 13, height: 13, bitsPerLine: 16, bits: @bits] ];
    };

  MakeShell: PROCEDURE [
    data: Containee.DataHandle,
    changeProc: Containee.ChangeProc ← NIL,
    changeProcData: LONG POINTER ← NIL]
    RETURNS [shell: StarWindowShell.Handle] = {
    body: Window.Handle ← NIL;
    name: XString.ReaderBody;
    ticket: Containee.Ticket;

    table: XStringTableFileFormat.Table;
    fh: NSFile.Handle;
    [table, fh] ← XStringTableFile.Open [data.reference];

    [name, ticket] ← Containee.GetCachedName [data]; -- This retrieves the file name from the file.

    shell ← StarWindowShell.Create [name: @name, scrollData: [
      displayHorizontal: TRUE, displayVertical: TRUE] ];

    Containee.ReturnTicket [ticket];

    body ← StarWindowShell.CreateBody [sws: shell,
      box: [ place: [0,0], dims: bodyWindowDims ] ];

    [] ← XStringTableWindow.Create [window: body, columns: table.colCt,
      rows: table.rowCt, cellContent: CellContent, clientData: table];

    Context.Create [context, LOOPHOLE[fh], DestroyContext, body];

    BEGIN
    z: UNCOUNTED ZONE ← StarWindowShell.GetZone [shell];
    repaint: XString.ReaderBody ← XString.FromSTRING["Repaint"L];
    items: ARRAY [0..1) OF MenuData.ItemHandle ← [
      MenuData.CreateItem [zone: z, name: @repaint, proc: RepaintMenuProc]];
    myMenu: MenuData.MenuHandle = MenuData.CreateMenu [
```

```
      zone: z,
      title: NIL,
      array: DESCRIPTOR [items]];
    StarWindowShell.SetRegularCommands [sws: shell, commands: myMenu];
    END;

    };

  PaintIconName: PROCEDURE [window: Window.Handle, iconBox, textBox: Window.Box, name: XString.Reader] = {
    -- textBox is relative to iconBox
    -- iconBox is relative to window
    lineToLineDeltaY: CARDINAL ← SimpleTextDisplay.systemFontHeight-3;
    textBox.place ← [
      x: iconBox.place.x+textBox.place.x,
      y: iconBox.place.y+textBox.place.y];
    [] ← SimpleTextDisplay.StringIntoWindow [
      string: name,
      window: window,
      place: textBox.place,
      lineWidth: textBox.dims.w,
      maxNumberOfLines: textBox.dims.h/(lineToLineDeltaY-1),
      lineToLineDeltaY: lineToLineDeltaY,
      wordBreak: TRUE];
    };

  PictureProc: Containee.PictureProc = {
    <<[data: Containee.DataHandle, window: Window.Handle, box: Window.Box,
       old, new: Containee.PictureState]>>
    textBox: Window.Box ← [ [x: 7, y: 10], [w: 55, h: 36] ];
    name: XString.ReaderBody;
    ticket: Containee.Ticket;
    IF new=garbage THEN RETURN;
    box.dims ← [64,64];
    [name, ticket] ← Containee.GetCachedName [data];
    SELECT old FROM
      garbage, ghost => {
        Display.Bitmap [
          window: window,
          box: box,
          bitmapBitWidth: 64,
          address: [sampleIconPicture, 0, 0],
          flags: Display.replaceFlags];
        PaintIconName [window, box, textBox, @name]};
      highlighted => Display.Invert[window, box];
      ENDCASE;
    SELECT new FROM
      highlighted => Display.Invert[window, box];
      ghost => {
        Display.White [window, box];
        PaintIconName [window, box, textBox, @name]};
      ENDCASE;
    Containee.ReturnTicket [ticket];
    };

  RepaintMenuProc: MenuData.MenuProc = {
    body: Window.Handle = StarWindowShell.GetBody[[window]];
    Window.InvalidateBox[body, [[0, 0], [30000, 30000] ]];
    Window.Validate[body];
    };

  SetImplementation: PROCEDURE = {
    oldImpl ← newImpl ← Containee.GetImplementation [ XStringTableFileFormat.xStringTableFileType ];
    newImpl.convertProc ← Containee.DefaultFileConvertProc;
    newImpl.genericProc ← GenericProc;
    newImpl.pictureProc ← PictureProc;
    newImpl.smallPictureProc ← SmallPictureProc;
    [] ← Containee.SetImplementation [ XStringTableFileFormat.xStringTableFileType, newImpl ];
    };

  SmallPictureProc: Containee.SmallPictureProc = {
    <<[data: Containee.DataHandle,
    normalOrReference: Containee.PictureState]
    RETURNS [smallPicture: XString.Character]>>
    RETURN [smallIconPicture]};

  -- Main line code

  smallIconPicture: XString.Character ← InitSmallPicture[];

  InitAtoms[];

  FindOrCreateIconFile[];

  InitBigPicture[];

  SetImplementation[];

  END.
```

```
-- File: XStringTableTestTool.mesa - last edit:
-- Breisacher.ES       14-Dec-84 14:57:14
-- Created by FormWindowLayoutTool on October 6, 84 17:04:14

DIRECTORY
  Attention,
  Context,
  FormWindow,
  Heap,
  MenuData,
  MessageWindow,
  StarWindowShell,
  TIP,
  Window,
  XString,
  XStringTableWindow;

XStringTableTestTool: PROGRAM
  IMPORTS Attention, Context, FormWindow, Heap, MenuData, MessageWindow, StarWindowShell, <<TIP, >>Window, XString, XStringTableWindow =
  {OPEN FW: FormWindow, XS: XString;

  -- TYPEs

  Items: TYPE = {msg, enumerateChangedCells, enumerateAllCells, appendRow, nRows, appendColumn, nColumns, hasAnyChanged, hasChanged, row,
  column, answer, resetAllChanged, getSelectedCell, repaint, tableWindow};

  Data: TYPE = LONG POINTER TO DataObject;
  DataObject: TYPE = RECORD [
    -- Record structure for the tool data
    -- Fill in with the variables used by tool
  ];


  -- Constants and Data

  formWindowDims: Window.Dims ← [1200, 1100];
  shellDims: Window.Dims = [650, 750];
  sampleString: XString.ReaderBody ← XString.FromSTRING["XStringTableWindow Test"L];

  columns: CARDINAL = 4;
  rows: CARDINAL = 5;

  tabStopInterval: CARDINAL = 50;

  context: Context.Type ← Context.UniqueType[];

  special1: XStringTableWindow.Cell = [0,1];
  special2: XStringTableWindow.Cell = [0,2];

  -- Procedures

  AppendARow: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    XStringTableWindow.AppendRows [
      window: FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD],
      nRows: CARDINAL[FormWindow.GetIntegerItemValue [window, Items.nRows.ORD]] ];
    };

  AppendAColumn: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    XStringTableWindow.AppendColumns [
      window: FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD],
      nColumns: CARDINAL[FormWindow.GetIntegerItemValue [window, Items.nColumns.ORD]] ];
    };

  CellContent: XStringTableWindow.CellContentProc = {
    <<[window: Window.Handle, cell: XStringTableWindow.Cell,
    clientData: LONG POINTER, callBack: PROCEDURE [XString.Reader]]>>
    testString: XString.ReaderBody ← XString.FromSTRING ["Test"L];
    rb2: XString.ReaderBody ← XString.FromSTRING ["Special1"L];
    rb3: XString.ReaderBody ← XString.FromSTRING ["Special2"L];
    nullrb: XString.ReaderBody ← XString.nullReaderBody;
    callBack [
      SELECT TRUE FROM
        cell.row = cell.column => @testString,
        cell = special1 => @rb2,
        cell = special2 => @rb3,
        ENDCASE => @nullrb];
    };

  CellNotify: XStringTableWindow.CellNotifyProc = {
    <<[window: Window.Handle, cell: XStringTableWindow.Cell,
       results: TIP.Results, clientData: LONG POINTER]
       RETURNS [processedResults: BOOLEAN ← FALSE]>>
    msgSW: Window.Handle ← FormWindow.GetWindowItemValue [window.GetParent, Items.msg.ORD];
    rb1: XString.ReaderBody ← XString.FromSTRING ["Cell notified and processed."L];
    rb2: XString.ReaderBody ← XString.FromSTRING ["Cell notified but NOT processed."L];
    SELECT cell FROM
      special1 => MessageWindow.Post [msgSW, @rb1];
      special2 => MessageWindow.Post [msgSW, @rb2];
      ENDCASE;
    RETURN [cell=special1];
    };

  EnumerateAllCells: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
```

```
    msgSW: Window.Handle ← FormWindow.GetWindowItemValue [window, Items.msg.ORD];
    spaces: XString.ReaderBody ← XString.FromSTRING ["  "L];

    EachCell: XStringTableWindow.EnumProc = {
      <<[window: Window.Handle, cell: XStringTableWindow.Cell,
      cellContent: XString.Reader] RETURNS [stop: BOOLEAN ← FALSE]>>
      MessageWindow.Post [msgSW, cellContent];
      MessageWindow.Post [msgSW, @spaces];
      };

    XStringTableWindow.EnumerateCells [
      window: FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD],
      callBack: EachCell];
    };

  EnumerateChangedCells: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>

    msgSW: Window.Handle ← FormWindow.GetWindowItemValue [window, Items.msg.ORD];
    spaces: XString.ReaderBody ← XString.FromSTRING ["  "L];

    EachChangedCell: XStringTableWindow.EnumProc = {
      <<[window: Window.Handle, cell: XStringTableWindow.Cell,
      cellContent: XString.Reader] RETURNS [stop: BOOLEAN ← FALSE]>>
      MessageWindow.Post [msgSW, cellContent, FALSE];
      MessageWindow.Post [msgSW, @spaces, FALSE];
      };

    MessageWindow.Clear [msgSW];
    XStringTableWindow.EnumerateChangedCells [
      window: FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD],
      callBack: EachChangedCell];
    };

  GetSelectedCell: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    rb: XString.ReaderBody ← XString.FromSTRING ["Something's wrong!"L];
    upperLeft, lowerRight: XStringTableWindow.Cell;
    selectionObject: XStringTableWindow.SelectionObject;
    msgSW: Window.Handle ← FormWindow.GetWindowItemValue [window, Items.msg.ORD];
    [upperLeft, lowerRight, selectionObject] ← XStringTableWindow.GetSelection [FormWindow.GetWindowItemValue [window,
    Items.tableWindow.ORD]];
    IF upperLeft # lowerRight THEN MessageWindow.Post [msgSW, @rb];
    FormWindow.SetIntegerItemValue [window, Items.row.ORD, upperLeft.row];
    FormWindow.SetIntegerItemValue [window, Items.column.ORD, upperLeft.column];
    };

  HasAnyBeenChanged: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    FormWindow.SetBooleanItemValue [
      window: window, item: Items.answer.ORD,
      newValue: XStringTableWindow.HasAnyBeenChanged [
        window: FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD]]]};

  HasBeenChanged: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    FormWindow.SetBooleanItemValue [
      window: window, item: Items.answer.ORD,
      newValue: XStringTableWindow.HasBeenChanged [
        window: FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD],
        cell: [
          row: CARDINAL[FormWindow.GetIntegerItemValue [window, Items.row.ORD]],
          column: CARDINAL[FormWindow.GetIntegerItemValue [window, Items.column.ORD]]]]]};

  Init: PROCEDURE = {
    Attention.AddMenuItem [
      MenuData.CreateItem [
        zone: Heap.systemZone,
        name: @sampleString,
        proc: MakeShell] ];
    };

  MakeShell: MenuData.MenuProc = {
    shell: StarWindowShell.Handle = StarWindowShell.Create [
      name: @sampleString,
      scrollData: [displayHorizontal: TRUE, displayVertical: TRUE]];
    formWindow: Window.Handle ← StarWindowShell.CreateBody [
      sws: shell,
      box: [[0,0], formWindowDims] ];
    FormWindow.Create [
      window: formWindow,
      makeItemsProc: MakeItems,
      layoutProc: DoLayout];
    StarWindowShell.SetPreferredDims [shell, shellDims];
    StarWindowShell.Push [shell];
    };

  MakeItems: FormWindow.MakeItemsProc = {
    fwz: UNCOUNTED ZONE = FW.GetZone[window];

    BEGIN
    wh: Window.Handle ← FormWindow.MakeWindowItem [
      window: window,
      myKey: Items.msg.ORD,
      size: [600, 100]];
    MessageWindow.Create [window: wh, zone: fwz, lines: 6];
    END;
```

```
BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Enumerate Changed Cells"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.enumerateChangedCells.ORD,
  commandProc: EnumerateChangedCells,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Enumerate All Cells"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.enumerateAllCells.ORD,
  commandProc: EnumerateAllCells,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Append A Row"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.appendRow.ORD,
  commandProc: AppendARow,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Number Of Rows:"L];
FormWindow.MakeIntegerItem [
  window: window,
  myKey: Items.nRows.ORD,
  tag: @rb,
  width: 100,
  initInteger: 1];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Append A Column"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.appendColumn.ORD,
  commandProc: AppendAColumn,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Number Of Columns:"L];
FormWindow.MakeIntegerItem [
  window: window,
  myKey: Items.nColumns.ORD,
  tag: @rb,
  width: 100,
  initInteger: 1];
END;

BEGIN
wh: Window.Handle ← FormWindow.MakeWindowItem [
  window: window,
  myKey: Items.tableWindow.ORD,
  size: [600, 500]];
[] ← XStringTableWindow.Create [
  window: wh, columns: columns, rows: rows,
  offset: [10,10], cellContent: CellContent,
  cellNotify: CellNotify, clientData: NIL];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["HasAnyBeenChanged"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.hasAnyChanged.ORD,
  commandProc: HasAnyBeenChanged,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["HasBeenChanged"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.hasChanged.ORD,
  commandProc: HasBeenChanged,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["ResetAllChanged"L];
FormWindow.MakeCommandItem [
  window: window,
  myKey: Items.resetAllChanged.ORD,
  commandProc: ResetAllChanged,
  commandName: @rb];
END;

BEGIN
rb: XS.ReaderBody ← XS.FromSTRING["Row:"L];
```

```
    FormWindow.MakeIntegerItem [
      window: window,
      myKey: Items.row.ORD,
      tag: @rb,
      width: 100,
      initInteger: 0];
    END;

    BEGIN
    rb: XS.ReaderBody ← XS.FromSTRING["Column:"L];
    FormWindow.MakeIntegerItem [
      window: window,
      myKey: Items.column.ORD,
      tag: @rb,
      width: 100,
      initInteger: 0];
    END;

    BEGIN
    rb: XS.ReaderBody ← XS.FromSTRING["Changed"L];
    FormWindow.MakeBooleanItem [
      window: window,
      myKey: Items.answer.ORD,
      label: [string [rb] ],
      initBoolean: FALSE];
    END;

    BEGIN
    rb: XS.ReaderBody ← XS.FromSTRING["GetSelectedCell"L];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: Items.getSelectedCell.ORD,
      commandProc: GetSelectedCell,
      commandName: @rb];
    END;

    BEGIN
    rb: XS.ReaderBody ← XS.FromSTRING["Repaint"L];
    FormWindow.MakeCommandItem [
      window: window,
      myKey: Items.repaint.ORD,
      commandProc: Repaint,
      commandName: @rb];
    END;

    };

  DoLayout: FormWindow.LayoutProc = {
    spaceAboveLine: CARDINAL = 10;
    leadingMargin: CARDINAL = 5;
    line: FormWindow.Line;

        -- set the tabs for FormWindow
    tabChoice: fixed FormWindow.TabStops = [fixed[ tabStopInterval]];
    FormWindow.SetTabStops[window: window, tabStops: tabChoice];
    -- Line 1
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.msg.ORD,
      line: line];
    -- Line 2
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.enumerateChangedCells.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.enumerateAllCells.ORD,
      line: line];
    -- Line 3
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.appendRow.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.nRows.ORD,
      line: line];
    -- Line 4
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.appendColumn.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.nColumns.ORD,
      line: line];
    -- Line 5
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.hasAnyChanged.ORD,
      line: line];
```

```
    FormWindow.AppendItem [
      window: window,
      item: Items.resetAllChanged.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.answer.ORD,
      line: line];
    -- Line 6
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.hasChanged.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.row.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.column.ORD,
      line: line];
    -- Line 7
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.getSelectedCell.ORD,
      line: line];
    FormWindow.AppendItem [
      window: window,
      item: Items.repaint.ORD,
      line: line];
    -- Line 8
    line ← FormWindow.AppendLine [window, spaceAboveLine];
    FormWindow.AppendItem [
      window: window,
      item: Items.tableWindow.ORD,
      line: line];
    };

  LocalFind: PROC [fw: Window.Handle] RETURNS [mydata: Data] = {
    mydata ← Context.Find[context, fw];
    IF mydata = NIL THEN ERROR;
    RETURN [mydata];
    };

  Repaint: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    tw: Window.Handle = FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD];
    Window.InvalidateBox [tw, Window.EntireBox [tw]];
    Window.Validate [tw];
    };

  ResetAllChanged: FormWindow.CommandProc = {
    <<[window: Window.Handle, item: FormWindow.ItemKey]>>
    XStringTableWindow.ResetAllChanged [FormWindow.GetWindowItemValue [window, Items.tableWindow.ORD]];
    };

  -- Mainline code

  Init[];

  }...
```

```
-- File: XStringTableWindow.mesa - last edit:
-- Breisacher.ES        9-Nov-84 16:26:16

-- Copyright (C) 1984 by Xerox Corporation

DIRECTORY
  MenuData USING [ArrayHandle],
  TableWindow USING [Area, Cell, FixedOrVarying],
  TIP USING [Results],
  Window USING [Dims, Handle, Place],
  XString USING [Reader];

XStringTableWindow: DEFINITIONS = BEGIN

<<

This interface provides a simple tabular display of strings in a window.

The client calls Create with the initial number of rows and columns and a procedure that will provide a string for each cell. All storage
for the strings is managed by the client, except for cells that are edited by the user. When the user edits a cell, XStringTableWindow
copies the client's string. The edited string can be retrieved later by the client by calling EnumerateChangedCells or LookAtCell. The
client can then reset the changed cell and XStringTableWindow destroys its copy of the string.

XStringTableWindow takes care of all display, notification, editing, etc. It grows and shrinks rows automatically as the user edits. It
allows the user to change the column widths.

The client can determine the value of a specific cell, can enumerate by row or column, and can set the value of a cell.

The user can add rows or columns. The client can determine the current number of rows and columns.

XStringTableWindow is intended to mostly hide the operations in TableWindow. If an operation appears in TableWindow and in
XStringTableWindow, the client should ALWAYS use XStringTableWindow. There are some operations in TableWindow that the client of
XStringTableWindow may properly use, such as NumberofRowsAndColumns, CellBox, GetColumnWidth, SetColumnWidth.

>>

<< Other potential future features: fixed scrolling rows and columns (spreadsheets), invisibility of rows and columns, scrolling units
(e.g., row at a time),...>>

  -- Create and destroy, etc.

  Create: PROCEDURE [
    window: Window.Handle,
    columns: CARDINAL,
    rows: CARDINAL,
    cellContent: CellContentProc,
    columnWidths: TableWindow.FixedOrVarying ← fixed,
    initialColumnWidths: CARDINAL ← 100,
    rulingLinesThickness: [0..10] ← 2,
    offset: Window.Place ← [0,0],
    options: Options ← defaultOptions,
    canCellChange: CanCellChangeProc ← NIL,
    cellNotify: CellNotifyProc ← NIL,
    minDimsChangeProc: MinDimsChangeProc ← NIL,
    minColumnWidth: CARDINAL ← 0,
    maxColumnWidth: CARDINAL ← CARDINAL.LAST,
    clientData: LONG POINTER ← NIL]
    RETURNS [menuItems: MenuData.ArrayHandle];
    -- menuItems should be installed in containing StarWindowShell.
    --   menuItems will include such things as Select Row, Append Row, etc.
    -- If cellContent is NIL, the table will be empty.
    -- See explanation of cellContent, copyStrings, etc. below.
    -- columnWidths rulingLinesThickness are in screen dots.

  CellContentProc: TYPE = PROCEDURE [window: Window.Handle,
    cell: Cell, clientData: LONG POINTER, callBack: PROCEDURE [XString.Reader]];
    -- This procedure will be called by TableWindowXString to obtain the
    --   contents of a cell. callBack should be called with the value
    --   of cell.
    -- If the user tries to edit the cell, XStringTableWindow WILL
    --   copy the string.

  Cell: TYPE = TableWindow.Cell; -- RECORD [row, column: CARDINAL]

  Area: TYPE = TableWindow.Area; -- RECORD [upperLeft, lowerRight: Cell];

  CanCellChangeProc: TYPE = PROCEDURE [window: Window.Handle,
    cell: Cell, clientData: LONG POINTER]
    RETURNS [cellCanChange: BOOLEAN];
    -- This is called when a cell is about to be edited by the user.
    -- If it returns FALSE, the edit is dissallowed.

  CellNotifyProc: TYPE = PROCEDURE [window: Window.Handle,
    cell: Cell, results: TIP.Results, clientData: LONG POINTER]
    RETURNS [processedResults: BOOLEAN ← FALSE];
    -- This optional client-supplied procedure will be called
    --   whenever a notification arrives for a cell.
    -- Most clients will not need this.
    -- This is useful if the client wants to do something other than
    --   the "normal" thing. (The normal thing to do with notifications
    --   is to have them operate on the contents of the cell as a
    --   simple text string.)
    -- If processedResults is TRUE, XStringTableWindow will ignore
    --   the results. Otherwise, it will go ahead and process the
    --   results as usual, as if the CellNotifyProc hadn't been called.
    << Ed. note: what I really had in mind here is spreadsheets where many of the cells are ordinary text, but some of the cells (formula
    cells) are not. I figured the spreadsheet guy that creates this window could supply one of these procedures, check to see if the cell
```

```
    is a formula cell and handle the notification itself. Otherwise, it just returns processedResults:FALSE and lets XStringTableWindow
    handle it as an ordinary text cell. >>

  MinDimsChangeProc: TYPE = PROCEDURE [window: Window.Handle,
    old, new: Window.Dims];
    -- If this proc is provided, it will be called each time
    --   the dimensions of the table change.

  Options: TYPE = RECORD [
    nextKeyDirection: RowOrColumn,
    nextKeyAtEndAddsARowOrColumn: BOOLEAN,
    userCanAddRows: BOOLEAN,
    userCanAddColumns: BOOLEAN,
    userCanAdjustColumns: BOOLEAN,
    userCanSelectRow: BOOLEAN,
    userCanSelectColumn: BOOLEAN];

  RowOrColumn: TYPE = {row, column};

  defaultOptions: Options = [
    nextKeyDirection: row,
    nextKeyAtEndAddsARowOrColumn: TRUE,
    userCanAddRows: TRUE,
    userCanAddColumns: TRUE,
    userCanAdjustColumns: TRUE,
    userCanSelectRow: TRUE,
    userCanSelectColumn: TRUE];

  Destroy: PROCEDURE [Window.Handle];

  IsIt: PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN];

  -- Changed bit for each cell.

  HasAnyBeenChanged: PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN];

  HasBeenChanged: PROCEDURE [window: Window.Handle, cell: Cell]
    RETURNS [yes: BOOLEAN];

  ResetChanged: PROCEDURE [window: Window.Handle, cell: Cell];

  ResetAllChanged: PROCEDURE [window: Window.Handle];

  EnumerateCells: PROCEDURE [window: Window.Handle, callBack: EnumProc, byRowOrColumn: RowOrColumn ← row];
    -- This will call the client's CellContentProc for
    --   the cells which have not been edited.
    -- byRowOrColumn=row means enumerate left to right, top to bottom.
    -- byRowOrColumn=column means enumerate top to bottom, left to right.

  EnumerateChangedCells: PROCEDURE [window: Window.Handle, callBack: EnumProc];
    -- Random order.

  EnumProc: TYPE = PROCEDURE [window: Window.Handle, cell: Cell,
    cellContent: XString.Reader] RETURNS [stop: BOOLEAN ← FALSE];
    -- The Reader is only valid during the call-back.

  -- Get specific cell

  LookAtCell: PROCEDURE [window: Window.Handle, cell: Cell,
    callBack: EnumProc];

  -- Selection

  SelectionObject: TYPE = {cellContent, cell, row, col, area, nil};

  SetSelection: PROCEDURE [window: Window.Handle,
    selectionObject: SelectionObject, upperLeft, lowerRight: Cell];

  GetSelection: PROCEDURE [window: Window.Handle]
    RETURNS [upperLeft, lowerRight: Cell, selectionObject: SelectionObject];

  -- Selection and input focus for a single cell

  SetCellSelection: PROCEDURE [
    window: Window.Handle,
    cell: Cell,
    firstChar: CARDINAL ← 0,
    lastChar: CARDINAL ← CARDINAL.LAST];

  SetInputFocus: PROCEDURE [
    window: Window.Handle,
    cell: Cell,
    beforeChar: CARDINAL ← CARDINAL.LAST];

  -- Editing

  DeleteRows: PROCEDURE [window: Window.Handle, firstRow, nRows: CARDINAL];
  DeleteColumns: PROCEDURE [window: Window.Handle, firstColumn, nColumns: CARDINAL];

  InsertRows: PROCEDURE [window: Window.Handle, beforeRow, nRows: CARDINAL];
  InsertColumns: PROCEDURE [window: Window.Handle, beforeColumn, nColumns: CARDINAL];

  AppendRows: PROCEDURE [window: Window.Handle, nRows: CARDINAL];
  AppendColumns: PROCEDURE [window: Window.Handle, nColumns: CARDINAL];

  -- Misc.
```

```
SetNextKeyDirection: PROCEDURE [window: Window.Handle, nextKeyDirection: RowOrColumn] RETURNS [old: RowOrColumn];
GetOptions: PROCEDURE [window: Window.Handle] RETURNS [options: Options];
-- Signals and errors
Error: ERROR [code: ErrorCode];
ErrorCode: TYPE = {notATableWindow};
END.
```

```
-- File: XStringTableWindowImpl.mesa - last edit:
-- Breisacher.ES       14-Dec-84 14:44:37

DIRECTORY
  Atom USING [ATOM, MakeAtom, null],
  Context USING [Create, Data, Destroy, Find, Type, UniqueType],
  Display USING [Handle, Shift, White],
  Heap USING [systemZone],
  MenuData USING [ArrayHandle],
  SimpleTextDisplay USING [MeasureString, Result, StringIntoWindow, systemFontHeight],
  SimpleTextEdit USING [ChangeSizeProc, CreateField, CreateFieldContext, DestroyField, DestroyFieldContext, Field, FieldContext, GetBox,
  GetClientData, GetValue, RepaintField, SetInputFocus, SetPlace, SetSelection, TIPResults],
  SpecialSimpleText USING [GetSelectedField],
  TableWindow USING [AppendColumns, AppendRows, Cell, CellBox, CellDisplayProc, Create, Destroy, FixedOrVarying, GetColumnWidth,
  GetRowHeight, NotifyProc, nullCell, NumberofRowsAndColumns, ObtainRowHeightProc, SetRowHeight],
  TIP USING [Results],
  Window USING [Box, GetBox, Handle, InvalidateBox, nullBox, Place, Validate],
  XString USING [nullReaderBody, Reader, ReaderBody],
  XStringTableWindow USING [CanCellChangeProc, Cell, CellContentProc, CellNotifyProc, defaultOptions, EnumProc, MinDimsChangeProc,
  Options, RowOrColumn, SelectionObject];

XStringTableWindowImpl: PROGRAM
  IMPORTS Atom, Context, Display, Heap, SimpleTextDisplay, SimpleTextEdit, SpecialSimpleText, TableWindow, TIP, Window, XString
  EXPORTS XStringTableWindow = BEGIN OPEN XStringTableWindow;

  -- TYPEs

  Data: TYPE = LONG POINTER TO DataObject;

  DataObject: TYPE = RECORD [
    window: Window.Handle ← NIL,
    inputFocus: Cell ← TableWindow.nullCell,
    steContext: SimpleTextEdit.FieldContext ← NIL,
    fields: FieldInfoHandle ← NIL,
    anyChanged: BOOLEAN ← FALSE,
    cellContent: CellContentProc ← NIL,
    canCellChange: CanCellChangeProc ← NIL,
    options: Options ← defaultOptions,
    cellNotify: CellNotifyProc ← NIL,
    minDimsChangeProc: MinDimsChangeProc ← NIL,
    clientData: LONG POINTER ← NIL];

  FieldInfo: TYPE = RECORD [
    cell: Cell ← TableWindow.nullCell,
    field: SimpleTextEdit.Field ← NIL,
    next: FieldInfoHandle,
    changed: BOOLEAN ← FALSE];

  FieldInfoHandle: TYPE = LONG POINTER TO FieldInfo;

  -- Constants and data

  context: Context.Type = Context.UniqueType[];

  sysZ: UNCOUNTED ZONE = Heap.systemZone;

  nextDown: Atom.ATOM ← Atom.null;

  defaultRowHeight: CARDINAL = 30;

  -- Procedures

  AddFieldToList: PROCEDURE [data: Data, cell: Cell]
    RETURNS [FieldInfoHandle] = {
    f: FieldInfoHandle ← NIL;
    IF data.fields = NIL THEN
      RETURN [ data.fields ← CreateFieldInfo [data, cell] ];
    FOR f ← data.fields, f.next UNTIL f.next = NIL DO ENDLOOP; -- find last one
    RETURN [ f.next ← CreateFieldInfo [data, cell] ];
    };

  AppendRows: PUBLIC PROCEDURE [window: Window.Handle, nRows: CARDINAL] = {
    data: Data = GetContext [window];
    rows: CARDINAL = TableWindow.NumberofRowsAndColumns[window].rows;
    TableWindow.AppendRows [window, nRows];
    FOR i: CARDINAL IN [rows..rows + nRows) DO
      TableWindow.SetRowHeight [ window, i, RecalculateRowHeight [data, i] ];
      ENDLOOP;
    };

  AppendColumns: PUBLIC PROCEDURE [window: Window.Handle, nColumns: CARDINAL] = {
    TableWindow.AppendColumns [window, nColumns];
    };

  CellDisplay: TableWindow.CellDisplayProc = {
    <<[window: Window.Handle, box: Window.Box, cell: Cell]>>
    data: Data = GetContext [window];

    displayString: PROCEDURE [r: XString.Reader] = {
      [] ← SimpleTextDisplay.StringIntoWindow [
        string: r,
        window: window,
        place: box.place,
        lineWidth: box.dims.w];
      };

    fi: FieldInfoHandle ← NIL;
```

```
    IF (fi ← FindField [data, cell]) # NIL THEN {
      SimpleTextEdit.SetPlace [f: fi.field, place: box.place];
      SimpleTextEdit.RepaintField [fi.field]}
    ELSE {
      IF data.cellContent = NIL THEN RETURN;
      data.cellContent [window, cell, data.clientData, displayString];
      };
    };

  CellHeight: PROCEDURE [data: Data, cell: Cell] RETURNS [height: CARDINAL ← SimpleTextDisplay.systemFontHeight] = {
    fi: FieldInfoHandle ← NIL;
    IF (fi ← FindField [data, cell]) # NIL THEN
      RETURN [SimpleTextEdit.GetBox [fi.field].dims.h];
    IF data.cellContent = NIL THEN
      RETURN [SimpleTextDisplay.systemFontHeight];
    BEGIN
    measureString: PROCEDURE [r: XString.Reader] = {
      height ← SimpleTextDisplay.systemFontHeight *
        LinesInString [r, TableWindow.GetColumnWidth [data.window, cell.column]];
      };
    data.cellContent [data.window, cell, data.clientData, measureString];
    END;
    };

  ChangeSize: SimpleTextEdit.ChangeSizeProc = {
    <<[f: SimpleTextEdit.Field, oldHeight: INTEGER,
       newHeight: INTEGER, repaint: BOOLEAN]>>
    data: Data = SimpleTextEdit.GetClientData [f];

    windowBox: Window.Box ← Window.GetBox [data.window];
    boxToShift: Window.Box ← Window.nullBox;
    placeToShiftTo: Window.Place ← [0,0];
    row: CARDINAL = FindCellFromField [data, f].row;
    oldRowHeight: CARDINAL = TableWindow.GetRowHeight [data.window, row];
    newRowHeight: CARDINAL ← 0;
    dif: INTEGER ← 0;

    IF newHeight < oldHeight AND repaint THEN {
      -- shrinking field.
      box: Window.Box ← SimpleTextEdit.GetBox [f];
      box.place.y ← box.place.y + newHeight;
      box.dims.h ← oldHeight - newHeight;
      Display.White [data.window, box]};

    newRowHeight ← RecalculateRowHeight [data, row];
    dif ← INTEGER[newRowHeight] - INTEGER[oldRowHeight];

    IF dif = 0 THEN RETURN;

    TableWindow.SetRowHeight [data.window, row, newRowHeight];

    ShiftFields [data, row, dif];

    IF repaint THEN {
      boxToShift ← SimpleTextEdit.GetBox [f];
      boxToShift.place ← [0, boxToShift.place.y + oldRowHeight];
      boxToShift.dims ← [windowBox.dims.w - boxToShift.place.x, windowBox.dims.h - boxToShift.place.y];

      placeToShiftTo ← [boxToShift.place.x, boxToShift.place.y + dif];

      Display.Shift [data.window, boxToShift, placeToShiftTo];

      IF dif > 0 THEN {
        boxToInvalidate: Window.Box ← boxToShift;
        boxToInvalidate.dims.h ← dif;
        Window.InvalidateBox [data.window, boxToInvalidate]};
      Window.Validate [data.window];
      };
    };

  Create: PUBLIC PROCEDURE [
    window: Window.Handle,
    columns: CARDINAL,
    rows: CARDINAL,
    cellContent: CellContentProc,
    columnWidths: TableWindow.FixedOrVarying ← fixed,
    initialColumnWidth: CARDINAL ← 100,
    rulingLinesThickness: [0..10] ← 2,
    offset: Window.Place ← [0,0],
    options: Options ← defaultOptions,
    canCellChange: CanCellChangeProc ← NIL,
    cellNotify: CellNotifyProc ← NIL,
    minDimsChangeProc: MinDimsChangeProc ← NIL,
    minColumnWidth: CARDINAL ← 0,
    maxColumnWidth: CARDINAL ← CARDINAL.LAST,
    clientData: LONG POINTER ← NIL]
    RETURNS [menuItems: MenuData.ArrayHandle ← NIL] = {

    data: Data ← sysZ.NEW [DataObject ← [
      window: window,
      cellContent: cellContent,
      canCellChange: canCellChange,
      options: options,
      cellNotify: cellNotify,
      minDimsChangeProc: minDimsChangeProc,
      clientData: clientData,
      steContext: SimpleTextEdit.CreateFieldContext [sysZ, window, ChangeSize] ]];
```

```
    Context.Create [context, data, DestroyContext, window];

    TableWindow.Create [window: window, columns: columns, rows: rows,
      cellDisplayProc: CellDisplay,
      notifyProc: Notify,
      rowHeights: varying,
      initialRowHeights: defaultRowHeight,
      obtainRowHeightProc: ObtainRowHeight,
      columnWidths: columnWidths,
      initialColumnWidths: initialColumnWidths,
      rulingLinesThickness: rulingLinesThickness,
      offset: offset,
      options: [userCanAdjustRows: FALSE, userCanAdjustColumns: options.userCanAdjustColumns],
      minColumnWidth: minColumnWidth,
      maxColumnWidth: maxColumnWidth ];

    };

  CreateFieldInfo: PROCEDURE [data: Data, cell: Cell]
    RETURNS [FieldInfoHandle] = {
    field: SimpleTextEdit.Field ← NIL;

    createField: PROCEDURE [r: XString.Reader] = {
      << A little kludge: we allow all rows to get as small as one text line tall by creating the field with proper dims.h. If we create
      the field with a taller dims.h, SimpleTextEdit won't ever allow the field to get smaller than that initial height.>>
      field ← SimpleTextEdit.CreateField [
        clientData: data,
        context: data.steContext,
        dims: [box.dims.w, SimpleTextDisplay.systemFontHeight],
        initString: r];
      };

    box: Window.Box = TableWindow.CellBox [data.window, cell];
    data.cellContent [data.window, cell, data.clientData, createField];
    IF field = NIL THEN createField[NIL];
    SimpleTextEdit.SetPlace [f: field, place: box.place];
    SimpleTextEdit.RepaintField [field];
    RETURN [sysZ.NEW [FieldInfo ← [cell: cell, next: NIL, field: field]] ];
    };

  Destroy: PUBLIC PROCEDURE [window: Window.Handle] = {
    TableWindow.Destroy [window];
    Context.Destroy [context, window];
    };

  DestroyContext: PROCEDURE [data: Data, window: Window.Handle] = {
    FreeFields [data];
    SimpleTextEdit.DestroyFieldContext [data.steContext];
    sysZ.FREE [@data];
    };

  EnumerateCells: PUBLIC PROCEDURE [window: Window.Handle, callBack: EnumProc,
    byRowOrColumn: RowOrColumn ← row] = {
    data: Data = GetContext [window];
    rb: XString.ReaderBody ← XString.nullReaderBody;
    cell: Cell ← TableWindow.nullCell;
    rows, columns: CARDINAL;
    [rows, columns] ← TableWindow.NumberofRowsAndColumns [window];
    FOR i: CARDINAL IN [0..IF byRowOrColumn = row THEN rows ELSE columns) DO
      FOR j: CARDINAL IN [0..IF byRowOrColumn = row THEN columns ELSE rows) DO
        cell ← IF byRowOrColumn = row THEN [i,j] ELSE [j,i];
        rb ← GetCellInternal [data, cell];
        IF callBack [window, cell, @rb] THEN RETURN;
        ENDLOOP;
      ENDLOOP;
    };

  EnumerateChangedCells: PUBLIC PROCEDURE [window: Window.Handle, callBack: EnumProc] = {
    data: Data = GetContext [window];
    FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
      rb: XString.ReaderBody ← SimpleTextEdit.GetValue [f.field];
      IF f.changed AND callBack [window, f.cell, @rb] THEN EXIT;
      ENDLOOP;
    };

  FindCellFromField: PROCEDURE [data: Data, field: SimpleTextEdit.Field]
    RETURNS [cell: Cell ← TableWindow.nullCell] = {
    FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
      IF f.field = field THEN RETURN [f.cell];
      ENDLOOP;
    };

  FindField: PROCEDURE [data: Data, cell: Cell]
    RETURNS [FieldInfoHandle] = {
    FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
      IF f.cell = cell THEN RETURN [f];
      ENDLOOP;
    -- If we get here, it wasn't in the list.
    RETURN [ NIL ];
    };

  FindOrCreateField: PROCEDURE [data: Data, cell: Cell]
    RETURNS [FieldInfoHandle] = {
    FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
      IF f.cell = cell THEN RETURN [f];
      ENDLOOP;
    -- If we get here, it wasn't in the list.
```

```
    RETURN [ AddFieldToList [data, cell] ];
    };

  FreeFields: PROCEDURE [data: Data] = {
    f: FieldInfoHandle;
    UNTIL data.fields = NIL DO
      f ← data.fields;
      data.fields ← f.next;
      SimpleTextEdit.DestroyField [f.field];
      sysZ.FREE [@f];
      ENDLOOP;
    };

  GetCellInternal: PROCEDURE [data: Data, cell: Cell] RETURNS [rb: XString.ReaderBody ← XString.nullReaderBody] = {
    -- does NOT copy bytes.
    fi: FieldInfoHandle ← NIL;
    IF (fi ← FindField [data, cell]) # NIL THEN
      rb ← SimpleTextEdit.GetValue [fi.field]<<
    ELSE IF data.cellContent = NIL
      THEN rb ← XString.nullReaderBody
      ELSE rb ← data.cellContent [data.window, cell, data.clientData]>>;
    };

  GetContext: PROCEDURE [body: Window.Handle] RETURNS [data: Data] = {
    data ← Context.Find[context, body];
    IF data = NIL THEN ERROR <<Error [notATableWindow]>>;
    };

  GetSelection: PUBLIC PROCEDURE [window: Window.Handle]
    RETURNS [upperLeft, lowerRight: Cell ← TableWindow.nullCell, selectionObject: SelectionObject ← nil] = {
    << This very crude first version of this procedure only handles the case where the selection is some text in a field. >>
    data: Data = GetContext [window];
    field: SimpleTextEdit.Field ← SpecialSimpleText.GetSelectedField [data.steContext];
    IF field # NIL THEN {
      upperLeft ← lowerRight ← FindCellFromField [data, field];
      selectionObject ← cellContent};
    };

  HandleNextKey: PROCEDURE [window: Window.Handle, data: Data] = {
    fieldInfo: FieldInfoHandle ← NIL;
    rows, columns: CARDINAL;
    IF data.inputFocus = TableWindow.nullCell THEN RETURN;
    [rows, columns] ← TableWindow.NumberofRowsAndColumns [window];

    IF data.inputFocus = [rows-1, columns-1] THEN {
      -- We're in the last cell
      IF ~data.options.nextKeyAtEndAddsARowOrColumn THEN RETURN;
      SELECT data.options.nextKeyDirection FROM
        row => {AppendRows [window, 1]; rows ← rows + 1};
        column => {AppendColumns [window, 1]; columns ← columns + 1};
        ENDCASE;
      };

    SELECT data.options.nextKeyDirection FROM
      row => data.inputFocus ←
        IF data.inputFocus.column+1 = columns THEN
          [data.inputFocus.row+1, 0]
        ELSE
          [data.inputFocus.row, data.inputFocus.column+1];
      column => data.inputFocus ←
        IF data.inputFocus.row+1 = rows THEN
          [data.inputFocus.column+1, 0]
        ELSE
          [data.inputFocus.column, data.inputFocus.row+1];
      ENDCASE;

    -- Now set the selection and input focus into the new cell
    fieldInfo ← FindOrCreateField [data, data.inputFocus];
    SimpleTextEdit.SetSelection [fieldInfo.field];
    SimpleTextEdit.SetInputFocus [fieldInfo.field];
    };

  HasAnyBeenChanged: PUBLIC PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN] = {
    data: Data = GetContext [window];
    RETURN [data.anyChanged];
    };

  HasBeenChanged: PUBLIC PROCEDURE [window: Window.Handle, cell: Cell]
    RETURNS [yes: BOOLEAN] = {
    data: Data = GetContext [window];
    f: FieldInfoHandle ← FindField [data, cell];
    RETURN [IF f = NIL THEN FALSE ELSE f.changed];
    };

  InitAtoms: PROCEDURE = {
    nextDown ← Atom.MakeAtom["NextDown"L];
    };

  IsIt: PUBLIC PROCEDURE [window: Window.Handle] RETURNS [yes: BOOLEAN] =
    {RETURN [ Context.Find[context, window] # NIL ]};

  LinesInString: PROCEDURE [r: XString.Reader, width: CARDINAL]
    RETURNS [lines: CARDINAL ← 0] = {
    rest: XString.ReaderBody ← r↑;
    result: SimpleTextDisplay.Result ← stop;
    UNTIL result = normal DO
      [, result, rest] ← SimpleTextDisplay.MeasureString [
```

```
        string: @rest, lineWidth: width, wordBreak: TRUE];
      lines ← lines + 1;
      ENDLOOP;
    };

  LookAtCell: PUBLIC PROCEDURE [window: Window.Handle, cell: Cell,
    callBack: EnumProc] = {
    data: Data = GetContext [window];
    rb: XString.ReaderBody ← GetCellInternal [data, cell];
    [] ← callBack [window, cell, @rb];
    };

  Notify: TableWindow.NotifyProc = {
    <<[window: Window.Handle, cell: Cell, results: TIP.Results]>>
    data: Data = GetContext [window];
    fieldInfo: FieldInfoHandle ← NIL;
    changed: BOOLEAN ← FALSE;
    tookFocus: BOOLEAN ← FALSE;

    CallSTE: PROCEDURE = {
      IF cell = TableWindow.nullCell AND data.inputFocus = TableWindow.nullCell THEN RETURN;
      fieldInfo ← FindOrCreateField [data,
        IF cell = TableWindow.nullCell THEN data.inputFocus ELSE cell];
      [tookFocus, changed] ← SimpleTextEdit.TIPResults [
        fieldInfo.field, results];
      IF tookFocus THEN data.inputFocus ← cell;
      fieldInfo.changed ← fieldInfo.changed OR changed;
      data.anyChanged ← data.anyChanged OR changed;
      };

    IF data.cellNotify # NIL AND data.cellNotify [window, IF cell = TableWindow.nullCell THEN data.inputFocus ELSE cell, results,
    data.clientData] THEN RETURN;

    FOR input: TIP.Results ← results, input.next UNTIL input = NIL DO
      WITH z: input SELECT FROM
        atom => {
          <<IF cell = TableWindow.nullCell THEN {
             This implies a non-mouse event, e.g. Delete or Next, so we probably want to check to see what the current selection is. If
             it's some text in a cell, then just pass it on (except for Next key), otherwise do something special.
             For now, we just call SimpleTextEdit always.>>
          IF z.a = nextDown THEN {HandleNextKey [window, data];EXIT};
          CallSTE [];
          EXIT;
          };
        string => {
          CallSTE [];
          EXIT;
          };
        ENDCASE;
      ENDLOOP;
    };

  ObtainRowHeight: TableWindow.ObtainRowHeightProc = {
    <<[window: Window.Handle,
    row: CARDINAL] RETURNS [height: CARDINAL]>>
    data: Data = GetContext [window];
    RETURN [RecalculateRowHeight [data, row]];
    };

  RecalculateRowHeight: PROCEDURE [data: Data, row: CARDINAL]
    RETURNS [height: CARDINAL ← 0] = {
    FOR c: CARDINAL IN [0..TableWindow.NumberofRowsAndColumns[data.window].columns) DO
      height ← MAX [height, CellHeight [data, [row, c]]];
      ENDLOOP;
    };

  RefigureAnyChanged: PROCEDURE [data: Data] = {
    FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
      IF f.changed THEN {data.anyChanged ← TRUE; RETURN};
      ENDLOOP;
    -- If we get here, there are no changed cells.
    data.anyChanged ← FALSE;
    };

  ResetChanged: PUBLIC PROCEDURE [window: Window.Handle, cell: Cell] = {
    -- Destroys the field, frees the FieldInfo, fixes up the list.
    -- If copyStrings was FALSE, destroys TableWindowXString's copy.
    data: Data = GetContext [window];
    previous: FieldInfoHandle ← NIL;
    FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
      IF f.cell = cell THEN {
        fi: FieldInfoHandle ← f;
        SimpleTextEdit.DestroyField [f.field];
        IF previous = NIL
          THEN data.fields ← f.next
          ELSE previous.next ← f.next;
        sysZ.FREE [@fi];
        EXIT};
      previous ← f;
      ENDLOOP;
    RefigureAnyChanged [data];
    };

  ResetAllChanged: PUBLIC PROCEDURE [window: Window.Handle] = {
    data: Data = GetContext [window];
    FreeFields [data];
    data.anyChanged ← FALSE;
```

```
  };

ShiftFields: PROCEDURE [data: Data, row: CARDINAL, dif: INTEGER] = {
  -- Does a SimpleTextEdit.SetPlace for all fields that are
  --  BELOW "row".
  newPlace: Window.Place ← [0,0];
  FOR f: FieldInfoHandle ← data.fields, f.next UNTIL f = NIL DO
    IF f.cell.row > row THEN {
      newPlace ← SimpleTextEdit.GetBox [f.field].place;
      newPlace.y ← newPlace.y + dif;
      SimpleTextEdit.SetPlace [f.field, newPlace]};
    ENDLOOP;
  };

-- Main line code

InitAtoms[];

END.
```

```
-- File: TelegraphLookupImpl.mesa - last edit:
-- Ando:IWA:Fuji Xerox   22-Aug-87 20:04:18
-- Mader.ES             19-Apr-85 11:34:44
-- Nagata.pa             1-Aug-85 11:11:48

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

-- Copyright (C) 1987 by Fuji Xerox Co., Ltd, Tokyo, Japan. All rights reserved.

DIRECTORY
  Catalog USING [beforeLogonSession],
  Dictionary USING [AqHomophone, Homophone, Homophones],
  Environment USING [wordsPerPage],
  Heap USING [Create],
  Lookup USING [
    BackspaceProc, DestroyProc, Handle, HomophonesProc, LookupProc, Procedures,
    ProceduresObject, ResetProc],
  NSFile USING [Handle],
  NSSegment USING [Map],
  Space USING [Unmap],
  TelegraphLookup,
  XChar USING [not, null],
  XCharSet0 USING [Make],
  XString USING [AppendChar, AppendReader, Character, ClearWriter, Context, Dereference, FreeWriterBytes, Lop, NewWriterBody,
nullReaderBody, nullWriterBody, ReaderBody, ReaderFromWriter, ReverseLop, WriterBody, WriterInfo];

TelegraphLookupImpl: PROGRAM
  IMPORTS Catalog, Heap, NSSegment, Space, XCharSet0, XString
  EXPORTS TelegraphLookup
  SHARES XString = BEGIN

  --TYPEs:

  TelegraphCode: TYPE = CARDINAL [0..9999];

  CodeArray: TYPE = ARRAY [0..maxHom) OF XString.Character;

  TelegraphCodeTable: TYPE = LONG POINTER TO ARRAY TelegraphCode OF CodeArray;

  TLH: TYPE = LONG POINTER TO TelegraphLookupObject;

  TelegraphLookupObject: TYPE = RECORD [
    procs: Lookup.Procedures ← @telegraphLookupProcs,
    buffer: XString.WriterBody ← XString.nullWriterBody,
    aqHomophoneArr: HomophoneObject ← ALL[[frequency: 0, partOfSpeech: 0, chars: XString.nullReaderBody, dictId: 0]],
    homophones: ARRAY [0..maxHom) OF Dictionary.Homophone ← ALL[NIL],
    chars: ARRAY [0..maxHom) OF XString.WriterBody ← ALL[XString.nullWriterBody],
    table: TelegraphCodeTable ← NIL];

  HomophoneObject: TYPE = ARRAY [0..maxHom) OF Dictionary.AqHomophone;

  telegraphLookupProcs: Lookup.ProceduresObject ← [
    backspace: BackspaceProc,
    destroy: DestroyProc,
    homophones: HomophonesProc,
    lookup: LookupProc,
    reset: ResetProc];

  --VARIABLEs:

  digit0: XString.Character = XCharSet0.Make[digit0];
  digit9: XString.Character = XCharSet0.Make[digit9];

  zone: UNCOUNTED ZONE = Heap.Create [initial: 1];

  maxHom: CARDINAL = 2;
  -- maxHom: CARDINAL = 3;
  -- Chinese uses only 2 telegraph code schemes
  -- 13577: Telegraph code table size

  --PUBLIC PROCEDUREs:

  Create: PUBLIC PROCEDURE [table: NSFile.Handle]
    RETURNS [handle: Lookup.Handle] =
  BEGIN
  tlh: TLH ← zone.NEW [TelegraphLookupObject];
  count: CARDINAL = ((TelegraphCode.LAST+1)*CodeArray.SIZE+(Environment.wordsPerPage-1))/Environment.wordsPerPage;

    tlh.table ← NSSegment.Map [
      origin: [file: table, base: 0, count: count],
      swapUnits: [uniform[1]],
      session: Catalog.beforeLogonSession].pointer;

    tlh.buffer ← XString.NewWriterBody [maxLength: 4, z: zone];
    FOR i:CARDINAL IN [0..maxHom) DO
      tlh.chars[i] ← XString.NewWriterBody [maxLength: 4, z: zone];
      tlh.homophones[i] ← @tlh.aqHomophoneArr[i];
      ENDLOOP;

    RETURN [LOOPHOLE [tlh]];
  END;  -- of Create


  --PRIVATE PROCEDUREs:

  BackspaceProc: Lookup.BackspaceProc =
  BEGIN
```

```
  tlh: TLH = TLHFromHandle [handle];
  endContext: XString.Context ← XString.WriterInfo [@tlh.buffer].endContext;

    FOR i: CARDINAL IN [0..count) DO
      [] ← XString.ReverseLop [XString.ReaderFromWriter[@tlh.buffer], @endContext];
    ENDLOOP;
    tlh.buffer.endContext ← endContext;

  END;  -- of BackspaceProc.

  DestroyProc: Lookup.DestroyProc =
  BEGIN
  tlh: TLH ← TLHFromHandle [handle];

    [] ← Space.Unmap [tlh.table];
    <<NSFile.Close [tlh.file];>>
    XString.FreeWriterBytes [@tlh.buffer];
    FOR i: CARDINAL IN [0..maxHom) DO
      XString.FreeWriterBytes [@tlh.chars[i]];
      ENDLOOP;
    zone.FREE [@tlh];
  END;

  HomophonesProc: Lookup.HomophonesProc =
  BEGIN
  tlh: TLH ← TLHFromHandle [handle];
  buffer: XString.ReaderBody ← XString.Dereference [XString.ReaderFromWriter [@tlh.buffer]];
  code, homCt: CARDINAL ← 0;
  char: XString.Character ← XChar.null;

    FOR c: CARDINAL IN [1..4] DO
      char ← XString.Lop [@buffer];
      IF ~char IN [digit0..digit9] THEN RETURN [NIL];
      code ← code * 10  + char - digit0;
    ENDLOOP;

    IF (char ← tlh.table[code][0]) = XChar.not THEN RETURN [NIL];
    FOR homCt IN [0..maxHom) DO
      IF (char ← tlh.table[code][homCt]) = XChar.not THEN EXIT;
      XString.ClearWriter [@tlh.chars[homCt]];
      XString.AppendChar [c: char, to: @tlh.chars[homCt]];
      XString.AppendReader [to: @tlh.chars[homCt], from: @buffer];
      tlh.aqHomophoneArr[homCt].chars ← XString.Dereference [XString.ReaderFromWriter [@tlh.chars[homCt]]];
      tlh.homophones[homCt] ← @tlh.aqHomophoneArr[homCt];
      REPEAT
        FINISHED => homCt ← homCt+1;
      ENDLOOP;
    RETURN [DESCRIPTOR [@tlh.homophones, homCt]];
  END;

  LookupProc: Lookup.LookupProc =
  BEGIN
  tlh: TLH = TLHFromHandle [handle];

    XString.ClearWriter [@tlh.buffer]; -- each call gives whole lookup string
    XString.AppendReader [@tlh.buffer, input];
  END;

  ResetProc: Lookup.ResetProc =
  BEGIN
  tlh: TLH = TLHFromHandle [handle];

    XString.ClearWriter [@tlh.buffer];
    FOR i: CARDINAL IN [0..maxHom) DO
      XString.ClearWriter [@tlh.chars[i]];
      ENDLOOP;
  END;  -- of ResetProc

  TLHFromHandle: PROCEDURE [handle: Lookup.Handle] RETURNS [tlh: TLH] =
  BEGIN
    IF handle^ = @telegraphLookupProcs THEN RETURN [LOOPHOLE [handle]];
    ERROR;
  END;  -- of TLHFromHandle.

  END.  -- of TelegraphLookupImpl


LOG (date - name - changes)
19-Apr-85 - Mader - Created.
22-Aug-87 - Ando - 13577: Telegraph code table size
```

```
-- File: FormWindowLayoutTool.doc - last edit:
-- Gobbel.pa           29-Apr-85 15:27:06
-- Diamond.PA          20-Aug-84 19:19:47
-- Breisacher.ES       19-Apr-84 14:52:04
-- Caro.pa              1-Apr-85 14:07:15 (but no fooling here!)
```


[4.0g Version: Please see Restrictions List]

The FormWindowLayoutTool allows a BWS programmer to graphically layout a FormWindow or PropertySheet. The tool automatically generates much of the Mesa source needed to produce a FormWindow or PropertySheet. These sources can then be compiled and executed and the resulting FormWindow will look like the one laid out earlier.

Description of the FormWindowLayoutTool:

After loading the tool in the Basic Workstation, a "FormWindow Layout Tool" menu item will be added to the Attention window menu. Bugging this brings up the layout tool, which has the following items:

ItemType: {Choice, Decimal, Integer, Boolean, Text, Command, Tagonly, Window}
  This enumerated item is used the select the type of form item that you want to put on your form.

Item Name:
  In the "Item Name:" field is the name for the next item that you want to put in your form.

Root:
  This is the root name of the programs, files, and tool or property sheet that will be generated.

Layout window:
  This is where the form will be "drawn".

The following menu items are in the header of the tool's window:

Close
  Puts the tool away. All data is lost.

DoIt
  This will cause the layout tool to generate Mesa sources. See description of options below.

Clear
  This will cause the layout window to be cleared.

Save
  This saves the items in the layout subwindow in a file. Load can then be used at a later time to bring the items back. The file name is the contents of the "root" item with "Form.by" appended. The file is a simple text file which can be copied back to the copilot volume if desired.

Load
  This will load a previously "SAVEd" form. The file read is the contents of the root item with "Form.by" automatically appended.

Options
  This will bring up an option sheet which will allow you to specify exactly which Mesa files to produce, etc. See description of options below.

Plagiarize
  This allows you to copy a FormWindow from a existing one. Just bug Plagiarize! and the cursor will change to a double circle. Move the cursor over the FormWindow you want to copy and hit Point. If you want to abort, hit the Adjust button.

Methods of Operation:

The layout tool has two basic modes: a layout mode and an editing mode. The two modes are easily distingushed. When the cursor is moved into the layout subwindow, if it remains an arrow you are in the editing mode. If the cursor changes to a picutre of the item described by the top fields then you are in the layout mode. The layout mode is used to add items to the subwindow. The edit mode is used to change the existing items in the layout subwindow.

The tool comes up in the editing mode. To get into the layout mode, put some text into the 'Item Name:' field. Now when the cursor is brought into the layout subwindow it changes to a brush in the form of the item selected. This item can then be moved around the subwindow to the desired position. Clicking the Point button will place the selected item in the subwindow. After the item is put in the window the cursor reverts back to an arrow and the mode goes back to the editing mode. The tool remains in the edit mode until any of the top fields which describe the items are changed. Changing

any of the top fields cause the tool to switch back to layout mode. If you are in the layout mode and you decide you do not want place the item in the layout subwindow, simply delete all the text in the 'Item Name' field. Whenever there is no text in this field the tool will remain in the editing mode.

When in the editing mode you can now select items to be manipulated by hitting Point over that item. Selected items are displayed by inverting the bits on the item. There can only be one item at a time which is selected. Once an item is selected, the following function can be done using the keyboard keys:

DELETE: will cause the current selection to be deleted.
UNDO: will bring back the last deleted item
MOVE: will allow you to move the selected item
STOP: will abort a move when in the middle of a move
PROPS: will bring up a property sheet on the selected item.

The properties of each item are the parameters needed for the call to FormWindow.MakexxxItem. Every item has such properties as Tag and Suffix. Each type of item has additional properties specific to that item type. There are field for each parameter for MakeXXXItem. By filling in these fields, little, if any, editing of the output code is necessary.

After working on the form, bug Options. The Options Property sheet allows you to specify exactly which files to produce and the style of those files. After the options have been set, you bug DoIt to generate the Mesa source. The files will have the name in the Root: field followed by .mesa.

Options and .mesa sources generated:

This tool has several uses. It can be used to generate code for "Tajo-like" tools which run in the Basic Workstation. It can be used to generate code for property sheets. It can be used to generate code for Star icon property sheets. Exactly what code gets generated is controlled by options which may be set by bugging the Options menu item. The desired options should be set before bugging DoIt.

The first option is "Type of Output". Choosing "Tool" will casue a "Tajo-like" tool to be generated. The file <root>Tool.mesa will be generated. When compiled and executed, this code will add a menu item to the attention window menu which, when selected, will bring up a StarWindowShell with the designed form as its body window. The FormWindowLayoutTool itself is an example of this type of tool.

Choosing "Type of Output" as "Property Sheet" will generate code for a property sheet. Exacty which type of property sheet generated will depend on the setting of the other options. Once the type of output is selected only the options pertaining to the specified type of output will appear.

If the type of output is a Tool the folowing options are dispalyed:

> Tool width and Tool height: these specify the display size of the tool.

> Scroll Bars: these specify whether the tool should have vertical and horizontal scroll bars.

If the type of output is Property Sheet the followinfg options will appear:

> Use XMessages: This indicates that the code generated should make full use of the XMessage interface and there should be NO strings in the code. If this option is off, no calls to XMessage will appear, there will be lots of local string constants.

> Type of property Sheet. Two styles of property sheets can be generated: If "NS File Atribute Backed" is selected this indicates that the property sheet is backed by NSFile attributes. This is true of so-called icon property sheets - those property sheets that appear when a Star icon is selected and PROPS is hit. When this option is on, each of the form items' property sheets will have an NSFileAttribute text item. The code generated will then have the appropriate calls to NSFile.GetAttributes, NSFile.ChangeAttributes, etc.
If Regular Property Sheet is selected a Tajo style property sheet is generated.

>> The next three boolean items control which files are produced. One, two or all three can be selected.

> rootPSheet.mesa -- This will cause a <root>PSheet.mesa file to be generated. This module will EXPORTs MakePropertySheet. Calling this procedure will cause the property sheet to be generated and displayed. This module contains the code that will produce the desired property sheet. This module will almost certainly require editing. The actual semantics of what happens when the user bugs "Help" or "Start" on a property sheet must obviously be supplied. The shells for procedures to be called when "Help", "Start, and "Reset" is bugged are provided. One only needs to fill in

the code to produce what is wanted. For "Done", "Apply", "Defaults", and "Cancel" routines are provided. For "Done" the tool writes a procedure to store the changed values in the property sheet.
> rootOps.mesa -- This will cause a <root>Ops.mesa DEFINITIONS file to be generated. This will contain TYPEs, the procedure MakePropertySheet, and message keys (if UseMessages is on) used in the tool. If this option is not used the type for the property sheet will be put in the <root>PSheet.mesa file. If messages are being used this file should be generated.

> rootMessagesImpl.mesa -- This will cause a <root>MessagesImpl.mesa file to be generated. This will contain all the strings for the property sheet. This module will EXPORT <root>Ops.GetMessageHandle. Again, if messages are beign used this file should be generated.

Note that all three of <root>PSheet.mesa, <root>Ops.mesa, <root>MessagesImpl.mesa are required to produce a "legal" Star property sheet.

> The next four options specify the size and location of the property sheets. If the size specified does not hold the all the items, the user will be able to scroll around to get to all the items.

> The final options are the possible header choices for property sheets. Each one of these selected will cause the corresponding item to appear at the top of the property sheet. For each selected a dummy procedure will be written which can be filled in by the user (with the more complete procedures noted above).

Getting the sources onto your Copilot volume:

[IGNORE THIS:
The tool runs in the Basic Workstation and the mesa files generated are placed in the system catalog on the Basic Workstation volume (usually the User volume). A hack, NSSnarf, makes it easy to retreive these files onto the Copilot volume. The FSWindowTool may be used also (but it is rather heavyweight for this simple retrieve). NSSnarf runs in Tajo/Copilot in the Exec. The default is to retrieve files from the system catalog of the User volume. ServicesStubsConfig.bcd MUST BE LOADED BEFORE NSSnarf!!! See the NSSnarf documentation for more information.]
Instead, get a filedrawer on your desktop (either using the filedrawer application or by booting Star) and store the .mesa file in that filedrawer, then retrieve it to your development environment Copilot/Tajo.

Current limitations, restrictions, known bugs:

> Some of the code that is generated WILL NOT COMPILE. This mainly is when required options are not specified. Filling in the property sheets for each item will eliminate these bugs. The code will usually produce several warnings about IMPORTed configurations which are not used. I have tried to make the tool smart enough to only put the files that are actually used in the IMPORT listed but a couple extras might appear. Just edit them out.

> With window items the larger than one line the line height for that item is not set correctly. In the LayoutProc it is necceary to change the height of the line with the window item to the height of the window. If this is not done the items will be placed on top of each other.

> Wrapped and vertically displayed choice items are "semi-supported" - the code generated will work, but the item will not be displayed in the tool's layout window the way it will actually look. Wrap indicators are denoted by a choice value string of "/w".

{4.0g additional restrictions}
Don't use the Plagarize command! It almost always crashes, especially if the target formwindow has a choice item.

I have put in minimal effort to fix the code generation. Mesa code generated will USUALLY need minor modification before it will compile correctly.

For best results, put the desktop in TWO LINE HEADER mode. Most of the command buttons overflow into the AUX menu with only one line of header.

Potential future features:

> Message window for error messages and help messages.
> User-specifiable menu items.
> Unique identifiers for each item.

Notes:
Use the property sheets to make the form as close as you can to the final code, ie put in the real

variable names you're going to use, rather than the defaults the tool provides.

It is not a bad idea to save the rootForm.by file generated by the Save command. This way, the next person to work on the program will be able to load this file and easily modify your tool's looks. Once you start doing massive edits to the generated source, the ability to easily do modifications become very difficult.

00

Converter, ConverterMsg

0.1 Overview

Converter is a conversion registration facility. A conversion is a program that reads an object that is in one format and produces an equivalent object of a different format. A conversion registered using the **Converter** interface can use all of the exterior trappings of the converter icon, including option sheets, folder and extended selection processing, and the converter History File. (See the *Viewpoint Reference Guide* for information about using the converter icon.) A conversion registers itself with the converter by providing entries for the property and option sheets, as well as the procedures which actually execute the conversion.

The **ConverterMsg** interface provides access to common user messages. These messages are used by some of the currently supported Viewpoint conversions, and are generic enough to be made made available for all conversions to use.

0.2 Interface Items

0.2.1 Converter

0.2.1.1 Procedures

```
Register: PROCEDURE [
  srcType: NSFile.Type,
  srcFormat,destFormat: XString.ReaderBody,
  convertProc: ConvertProc,
  sizeChange: CARDINAL ← 100,
  override: BOOLEAN ← TRUE]
RETURNS [
  old: ConvertProc,
  status: Status ← registered];
```

Allows clients to plug in conversions to the Converter. It must be called once for each (type, source format, destination format) triplet. The parameters are defined as follows:

srcType: The type of file in which the source format of the conversion can be found. A client can make many calls to Register, each specifying a different file type, if the source format can possibly be found in files of different types. For instance, ASCII documents can be found in files of type 0 or 2, so 2 separate calls to **Register** are used in the system's ASCII conversion, each one with a different **srcType** value.

srcFormat: This will appear in the ***Source Format*** line in the property and option sheets. This string is the name of the source format. The Converter makes a copy of this parameter, so the client may allocate it out of the local frame.

destFormat: This will appear in the ***Destination Format*** line in the property and option sheets. This string is the name of the destination format. The Converter makes a copy of this parameter, so the client may allocate it out of the local frame.

convertProc: This is the actual procedure which will execute the conversion. Clients are responsible for reading from the source file, and returning a handle to the resulting file. If the conversion has failed for any reason, **NSFile.nullHandle** must be returned, and the client should post an explanatory message.

sizeChange: This parameter is used for estimating the size of the object resulting from this conversion, relative to the size of the source object. This is used to determine whether or not there is enough disk space left to complete the conversion. It is a percentage. For example, if the resulting object will be about the same size as the original, 100 (the default) should be used. If the resulting object will be approximately half of the size, 50 should be used.

override: If a conversion already exists for the source-destination pair given (as determined by an **XString.Compare** of the strings), this parameter is consulted. If it is **TRUE,** the new registration will override the old one, and the **status** returned from **Register** will be ***overridden***. If it is **FALSE**, the registration will not take place, and the **status** returned will be ***alreadyThere***.

old: This allows the user to save the old conversion procedure. If no conversion was previously registered for the given source-destination pair, this parameter will equal the **convertProc** parameter passed to **Register**.

status: This returns the results of the registration. If no conversion existed for the indicated source-destination pair, and all of the parameters passed are valid (i.e. no empty strings) the **status** will be ***registered***. If a conversion previously existed, and the override parameter is **TRUE**, the registration will take place, and the status will be ***overridden***. If a conversion previously existed, and the override parameter is **FALSE**, the **status** will be ***alreadyThere***, and the registration will not take place. This return parameter can be used for the case where a client wishes to use a new procedure in place of an old one for a particular conversion, but wishes to use the old one either for use in setting up another conversion, or for replacing the new one in the future.

PostMessage: PROCEDURE [
msg: XString.ReaderBody, cvData: CvData];

Provides clients with a way to both post messages to the user in the attention window, and to put them in the Converter History File. The client is responsible for making sure all messages are fully multinational, if necessary. The **cvData** parameter must be the one that is passed to each ConvertProc. If **cvData** is **NIL**, **PostMessage** will do nothing.

GetZone: PROCEDURE RETURNS [UNCOUNTED ZONE] = INLINE{...};

For convenience, the Converter implementation provides a public zone for clients to allocate storage from for the duration of their conversions. Since this zone is shared by all conversion, plus the Converter, it is important that *all storage allocated from it is freed* once the conversion has completed.

GetPOption: PROCEDURE RETURNS [PaginateOption] = INLINE{...};

If a conversion is going to generate a ViewPoint document, the user executing that conversion is allowed to determine the pagination option affecting the resulting document. The user specifies which is preferred by including the following entry in the UserProfile:

[Conversion]
Document Pagination: *option*

where ***option*** is one of the following:

compress: This provides all of the outwardly visible signs of pagination (e.g. headings, footings, page numbers and other page format properties), and leaves the structure of the document in its optimized form. This makes any subsequent document accesses potentially faster, depending on the structure of the document. This pagination also takes the longest to execute.

simple: This provides all of the outwardly visible signs of pagination (e.g. headings, footings, page numbers and other page format properties), but could possibly leave the document fragmented, so that it may be larger than optimal, and scattered, so that different pieces of it cannot be accessed as quickly during. This pagination is faster than **compress**, but it could possibly lead to slower document accessing.

none: This provides no pagination at all, but leaves the document in its raw form. This is by far the fastest method, but doesn't show the output document in its completed form, and could make subsequent document accessing very slow.

When the conversion calls a Documents procedure which needs a pagination option, the expected parameter is usually of a different type, usually something internal to Documents. To use the Converter paginate option, the following method is

recommended:

```
(SELECT Converter.GetPOption[] FROM
    compress => corresponding value,
    simple => corresponding value,
    none => corresponding value,
    ENDCASE => ERROR)
```

0.2.1.2 Types

```
ConvertProc: TYPE = PROCEDURE [
  source: NSFile.Handle, cvData: CvData]
RETURNS [
  dest: NSFile.Handle← NSFile.nullHandle];
```

Conversion procedures are declared to be of this type. The client is responsible for posting any error messages, using **PostMessage** and the **cvData** parameter, and for returning either a valid **NSFile.Handle** to the resulting file, or **NSFile.nullHandle** if the conversion was not executed. If a **nullHandle** is returned, it is expected that the conversion will have posted a user message explaining why the conversion was not completed. The client is responsible for determining whether or not the contents of the source file are actually in the indicated source format, and must handle garbage gracefully, with appropriate error messages where needed.

```
Status: TYPE = {registered, overridden, alreadyThere};
```

```
CvData: TYPE [2];
```

```
PaginateOption: TYPE = {compress, simple, none};
```

See **GetPOption** above for a description of the use of this type.

0.2.2 ConverterMsg

0.2.2.1 Procedures

```
Get: PROCEDURE [key: XMessage.MsgKey]
  RETURNS [XString.ReaderBody] = INLINE{...};
```

This is the equivalent of a call to **XMessage.Get**, except with the private **ConverterMsg** message handle and with the message key indicated. The message key MUST be one of the public message keys provided by the **ConverterMsg** interface (see below), or results will be unpredictable.

0.2.2.1 Constants

```
kdataSkipped, kuserAbort, kdamagedDocument,
kfileInUse, knoDocuments, kcantOpen, koutOfSpace,
kunknownProblem, kincompatible, kokNoSpaceToPaginate,
kokUnknownPaginateProblem : XMessage.MsgKey;
```

Each of the constants below represent a different message. The text of the messages is as follows:

kdataSkipped
"Some part of the source object was skipped during conversion."

kuserAbort
"The conversion was aborted by the user."

kdamagedDocument
"The selected VP document is damaged and cannot be opened."

kfileInUse
"The source object is currently in use."

knoDocuments
"Please load VP Document Editor before attempting any document conversions."

kcantOpen
"The source object could not be opened."

koutOfSpace
"There is not enough disk space left to complete the conversion."

kunknownProblem
"The source object was not converted due to an unexpected error."

kincompatible
"The source object cannot be converted because it is an incompatible version."

kokNoSpaceToPaginate
"There is not enough disk space left to paginate the converted document."

kokUnknownPaginateProblem
"The converted document was not paginated due to an unexpected error."

0.3 Usage

Suppose a procedure had been written to implement ViewPoint document to Foobar document conversion. A possible implementation for plugging this into the Converter would be:

```
-------------TYPES--------------

MessageKey: TYPE = {documentVP, fooBar, noConverter};

-----------MESSAGES----------

h: XMessage.Handle ← NIL;

knoConverter: XMessage.Msgkey = MessageKey.noConverter.ORD;
kdocumentVP: XMessage.Msgkey = MessageKey.documentVP.ORD;
kfooBar: XMessage.Msgkey = MessageKey.fooBar.ORD;

InitMessages: PROCEDURE =
{
    msgArray: ARRAY MessageKey OF XMessage.MsgEntry ← [
      document8000: [
          msgkey: kdocument8000,
          msg: XString.FromSTRING["VP Document"L],
          type: userMsg,
          id: 0],
      fooBar: [
          msgkey: kfooBar,
          msg: XString.FromSTRING["Foobar Document"L],
          type: userMsg,
          id: 1],
      noConverter: [
          msgkey: knoConverter,
          msg: XString.FromSTRING["Please load the Conversion Common Software
      before attempting any conversions."L],
          type: userMsg,
          id:2]];
    messages: XMessage.Messages ← DESCRIPTOR [
      LOOPHOLE[msgArray, ARRAY[0..MessageKey.LAST.ORD] OF
      XMessage.MsgEntry] ];
    h ← XMessage.AllocateMessages [
      applicationName: "VP To Foobar Conversion"L,
      maxMessages:MessageKey.LAST.ORD + 1,
      clientData: NIL,
      proc: DeleteMessages ];
    XMessage.RegisterMessages[
      h: h,
      messages: messages,
      stringBodiesAreReal: FALSE];
}; -- InitMessages

DeleteMessages: PROCEDURE [clientData: XMessage.ClientData] = {};

ConvertStarToFoobar: Converter.ConvertProc =
```

```
{
    attr: ARRAY [0..1) OF NSFile.Attribute ← [[type[StarFileTypes.text]]];
    IF ~CheckVersion[source] THEN{
      OPEN Cm:ConverterMsg;
      Converter.PostMessage[Cm.Get[Cm.kincompatible], cvData];
      RETURN};
    dest ← NSFile.Create[NSFile.nullHandle, DESCRIPTOR[attrArray]];
    .
    -- do conversion here
    .
    .
}; -- ConvertStarToFoobar

RegisterNow: PROCEDURE =
{ -- converter is now loaded
    [ ] ← Converter.Register[
      StarFileTypes.document,
      XMessage.Get[h, kdocument8000],
      XMessage.Get[h, kfooBar],
      ConvertStarToFoobar];
}; -- RegisterNow

RetryRegistration : Event.AgentProcedure =
{
    RegisterNow[];
    remove ← TRUE;
}; -- RetryRegistration

CheckVersion: PROCEDURE [source: NSFile.Handle] RETURNS [BOOLEAN] =
{
    -- perform checking to make sure file is in correct Foobar format
}; -- CheckFormat

-- Mainline Code
InitMessages[];
IF Runtime.IsBound[LOOPHOLE[Converter.Register]] THEN RegisterNow[]
ELSE [] ← Event.AddDependency[
          RetryRegistration,
          NIL,
          Atom.MakeAtom["ConverterLoaded"L]];
}.
```

0.4 Conventions

0.4.1 User Messages

While a conversion is executing, the client is responsible for generating all information messages. Clients should use the ConverterMsg messages as much as possible, when they are appropriate.

0.4.2 Aborting Conversion

The Converter is designed to recognize the STOP key between objects being converted, thus allowing the conversion of many objects to be aborted before it is complete. Each client conversion, though, is responsible for recognizing the STOP key within the conversion procedure. It is recommended that this be done more often than just once or twice, but not as often as every character. This is left primarily up to the discretion of the implementor. To provide a consistent user interface, it is recommended that as long as the integrity of the result can be guaranteed, partial results should be provided after the STOP key is recognized, and the message from the **ConverterMsg** interface should be posted.

0.4.3 ViewPoint Document Pagination

Whenever a conversion creates a ViewPoint document, the client procedure should be sure to use the **GetPOption** procedure for the appropriate parameter to the Documents world, as shown above. This is the expected application of the Conversion entry in the User Profile.

0.4.4 Event Notification

When a conversion routine is loaded, it must register itself with the Converter using the **Register** procedure. The client should be careful about checking for the fact that this procedure is bound, that the Conversion Common Software has been loaded. If **Register** is not bound, the client should use the method shown in the **Usage** section above to add a dependency on the "ConverterLoaded" atom. Adding this dependency guarantees that the registration will wait until the Conversion Common Software is started before executing. Also, the client should be sure and post an appropriate message to the user, as shown above.

0.5 Index of Interface Items

OfflineDiagKernel.doc: This files contains the documentation for
OfflineDiagKernelDove and OfflineDiagKernelDLion.
Last updated on 14-Jan-86 8:30:41 by KL

Documentation base:

OfflineDiagInterface.mesa
OfflineDiagInterfaceExtra.mesa
OfflineDiagnosticControlModule.mesa
OFLFloppyExecImpl.mesa
OfflineDiagnosticVersionImpl.mesa
OfflineDiagTTYDove.mesa
OfflineDiagTTYImplDove.mesa
CMDiagMsgKeysDove.mesa
CMDiagMsgKeysImplDove.mesa

1.0 Introduction

OfflineDiagKernel (OfflineDiagKernelDove or OfflineDiagKernelDLion) is a Utility Pilot client. It is a binary configuration file that is bound with UtilityPilotKernel, either DLion or Dove basic heads and all the components required to build a bootable system. Offline diagnostics can then run as clients of OfflineDiagKernel by simply binding with OfflineDiagKernelDove or OfflineDiagKernelDLion.

The kernel package provides standardized and, hopefully, user-friendly interfaces between the users of the various offline diagnostic packages and the diagnostic clients: it provides facilities to the diagnostic programs to interact with the users of the various diagnostic packages via menus, option tables and/or interactive prompts; correspondingly, users control the running of the various diagnostic programs by interacting with the kernel module by selecting options from an option table or menu items from a menu or by simply responding to interactive prompts. Various flexible data presentation facilities are available to the programmers to present both static and/or dynamic test data to the users. In this manner, an uniform and consistent user interface is provided to both the users of the diagnostic packages and the diagnostic programmers.

Functions such as error checking, management of common memmory spaces, menu generation, screen management, transfer between the kernel and client worlds and system interfacing are automatically performed by OfflineDiagKernel. For details, please see OfflineDiagInterface, OfflineDiagInterfaceExtra, OfflineDiagTTYDove and their implementaion modules.

OfflineDiagKernel supports single client boot files as well as multiple client boot files; in concert with OFLFloppyExecDove, clients spanning multiple floppy disks are supported. These various configurations can be created by simply binding the various clients in the same config file with the kernel modules (OfflineDiagKernelDove.bcd or OfflineDiagKernelDLion.bcd).

1.1 Definitions

To minimize confusion, a few definitions are in order.

a) OfflineDiagKernel or kernel is the generic term used for OfflineDiagKernelDLion and OfflineDiagKernelDove. The two are identical except OfflineDiagKernelDLion binds with BasicHeadsDLion and OfflineDiagKernelDove binds with BasicHeadsDove.

b) A user is defined as someone who runs a diagnostic ssytem. It can be a normal user of a workstation, a system administrator, a techical support personnel, a manufacturing personnel or a programmer. Whenever a user logs on, a security level is assigned which determines what the user can do.

c) A diagnostic client, or simply a client, refers to a program/programmer combination which wishes to use the facilities provided by OfflineDiagKernel to implement an application.

d) Menu, to the user, is a list of items from which the user can make

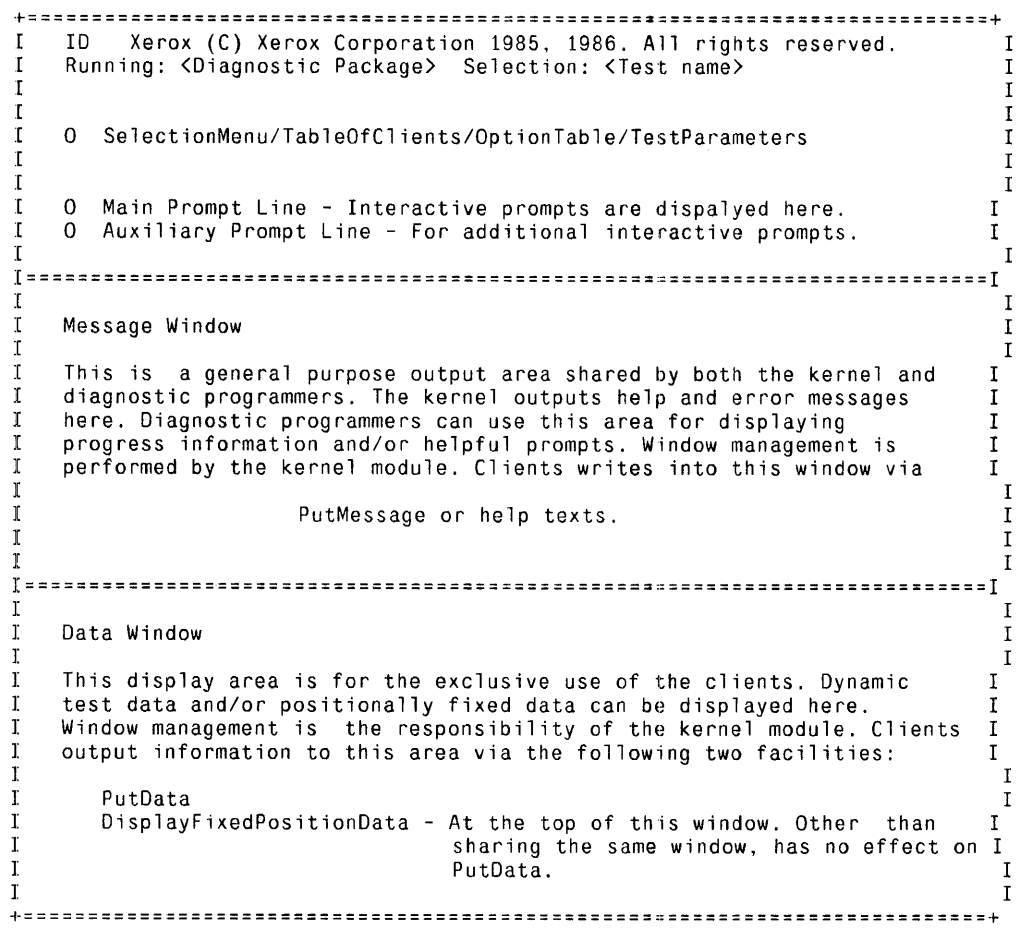
selections simply by entering the numbers associated with the selections. This end-user menu is different for different classes of users in terms of contents and ordering of menu items.

To the diagnostic client, a menu is a collection of logically related test items from which OfflineDiagKernel can build an end-user menu based on the end-user security level. Normally, an end-user menu is a subset of the diagnostic client menu.

Internally, in reference to the kernel module, a menu is a node in a general tree. These menu nodes are built from a data base of test items a client has deposited in a heap. Each menu item can be an actual test or a pointer to another menu. The depth (???) of the tree is only limited by the available real memmory.

2.0 Dove Diagnostic Screens

OfflineDiagKernel uses the keyboards and bitmapped displays supported by both DLion and Dove machines as the respective input and output media. Generally, the bitmapped screen is divided into 3 orthorgonal windows as follows:



Window creation and management are automatic functions performed by the kernel module. The message and data windows are not displayed until they are needed. If the Message Window is needed and the Data Window is not used, the Message Window takes up the entire area below the prompts. When the need arises to create the Data Window, the Message Window will be automatically trimmed. When both windows are created, the Data Window is twice the size of the Message Window.

Both the Data and Message windows are created whenever the Data Window is created.

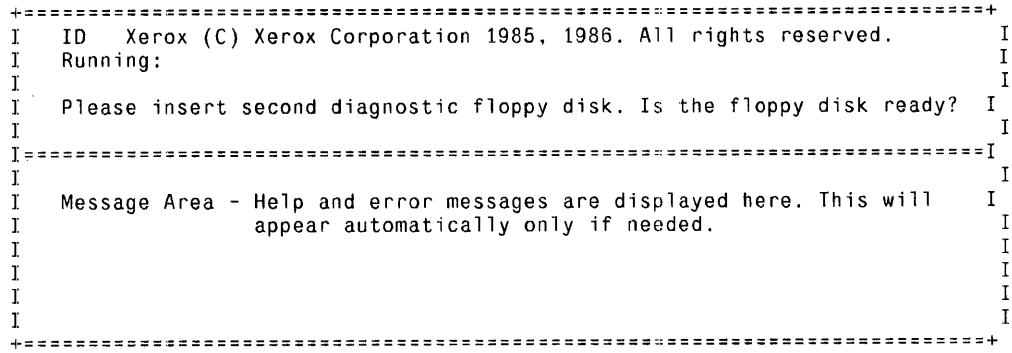
The sizes of the windows differ from screen to screen. They use the space

left over by the top window.

2.1 Floppy Executive Screen

This screen is displayed only if the Floppy Executive is bound in the boot file. This is the case when client pacakges are stored on one or more floppy disks; the Floppy Executive then loads the clients from these floppy disks.

2.1.2 Screen Lay-Out



When the second floppy disk is

2.2.2 Valid inputs

"Please insert second diagnostic floppy disk. Is the floppy disk ready?" prompts the user to insert the second floppy diskette containing the diagnsotic programs.

"Y" loads the program files on the second floppy diskette. While loading, the following progress message is displayed:

"Loading diagnostics from the second floppy disk..."

"N" redisplays the prompt.

Once the programs are loaded, the kernel will check the version numbers of the clients. If the version numbers of the kernel and the clients do not match, the following message is displayed:

"Version mismatch!!! First floppy version: <> Second floppy version: <>".

Along with the above message, the user is promnpted with:

"Should I ignore version mismatch and continue? [Y/N]".

"Y" exits and presents the Login Screen to the user.
"N" reprompts the user to load the second diagnostic diskette.

2.3 Login Screen

This is the first screen presented to the user after a successful boot. The purpose of this screen is to allow users to log in at the different security levels.

Diagnostic users are divided into 5 classes as follows:

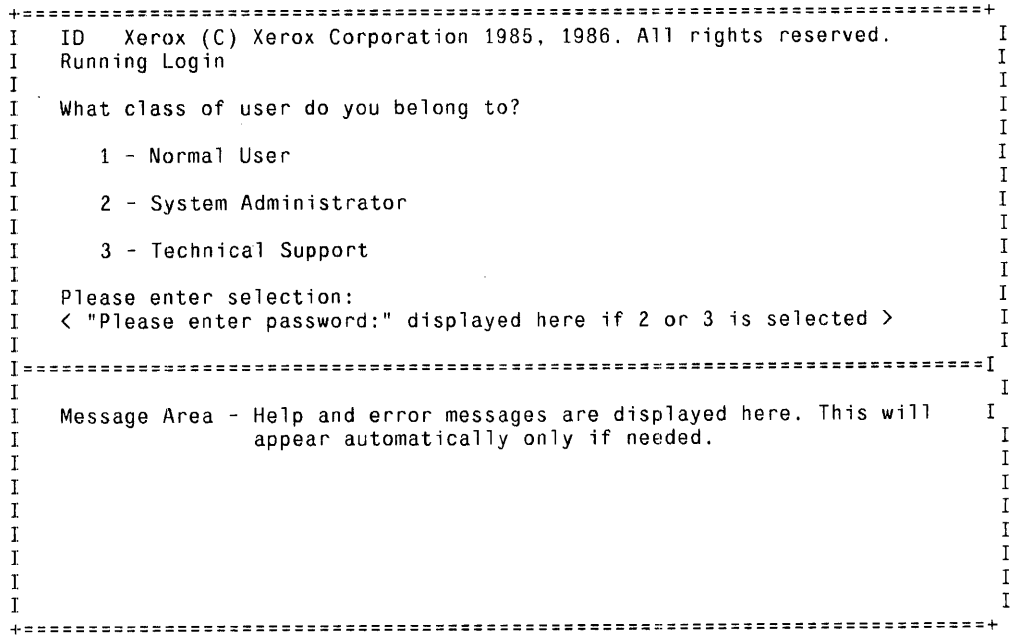
Normal Users - No login needed. Selecting 1 will advance to next menu.
System Administrators - "rgmsn"
Field Service - "rexifsn"
Manufacturing - "fle"
Programmers/Engineers - "do8"

The tests each class of users can run may be different depending on the riskiness of the tests. Generally, the test runnable by the Normal Users are

harmless, so no login is required. All the other classes of users must enter a valid password. The passwords entered determine the contents of the menus that will be generated and presented to the users for selection. The specification of who can run each test is done by the diagnostic programmer. Also, the presentation order of the menu items may be different for different classes of users.

The passwords are meant to make the user aware of the increased in riskiness of damaging the system when other than Normal Users is used; they are not meant to deny access to facilities.

2.3.1 Screen Lay-Out



2.3.2 Valid inputs

The valid inputs for this screen are "1", "2", "3", "?" and STOP.

Entering a question mark displays this screen's help text on the screen. This screen's help text is as follows:

Log In Help
Normal users can only run harmless tests that give qualitative indications. No login is needed
Other users can run tests that may damage the system. They must login with a valid password.

"1" identifies the user as a normal user. All menu generated will only contain items selectable by this class of users. This selection exits the Login Screen and presents the Subsystem Selection Screen to the user.

"2" identifies the user as a System Administrator. The user will be prompted to enter a password. If the password entered is incorrect, the user must reselect a menu item and start over.

"3" identifies the user as a Technical Representative, from Manufacturing or a programmer. The user will be prompted to enter a password. The password entered will identify the user's organization. Again, if the password entered is incorrect, the user must reselect a menu item and start over.

STOP aborts the current action or redisplays the Login Menu.

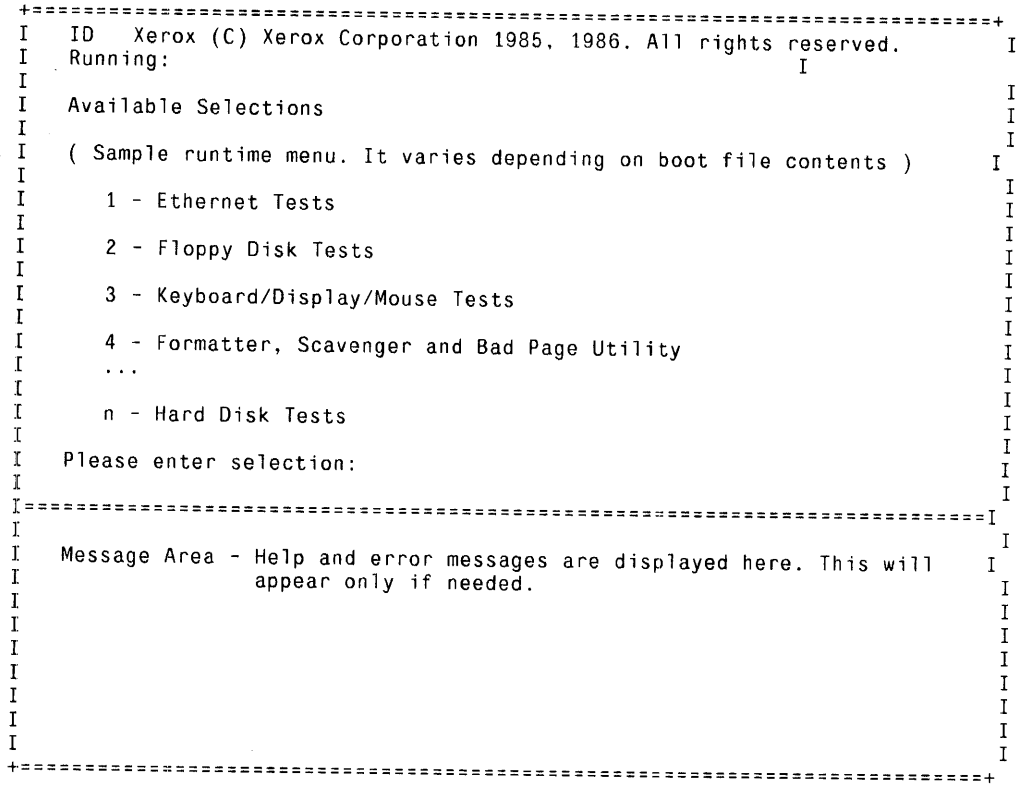
After entering the correct password, the Subsystem Selection Screen is presented to the user if the number of diagnsotic clients is greater than one.

2.4 Subsystem Selection Screen

This screen is displayed after a user has successfully logged in. This screen presents to the user the diagnostic packages bound into this boot file. If the boot file contains only on diagnostics package, this screen is bypassed automatically; and the diagnostic screen containing the top-level menu for the subsystem under test is presented to the user for selection.

If there are more than one diagnostic packages bound into the boot file, the Subsystem Selection Menu containing all the available packages is displayed for selection.

2.4.1 Screen Lay-Out



2.4.2 Valid inputs

The valid inputs for this screen are "1" to "n", "?" and STOP.

"1" to "n" select the associated subsystem for testing. Once a vaild subsystem has been selected, the top-level menu for the subsystem under test is presented to the user for selection.

STOP exits this menu and returns to the Login Menu.

"?" displays some useful information concerning features to the user:

Useful information
The blinking cursor points to the applicable prompt. Do as prompted
SPACE and CR are input terminators. BACKSPACE erases the last input character
A question mark alone (?) gives help to the entire menu
<n> followed by ? gives help to the item identified by n
UNDO inverts the screen
STOP aborts the current test, if allowed; or exits the current menu

3.0 Data presentation facilities

Because diagnostics need to give user information of various forms, OfflineDiagKernel provides many simple facilities to the programmers to present test information and data to the users. These include positionally

fixed test paramters and data, menus and option tables, dynamic helps messages and data, and interactive prompting. Additionally, the kernel automatically displays relevant information such as subsystem under test and test selected, test results and a simple message indicating what the last selection, if any, was whenever a menu is entered.

3.1 Positionally fixed test information

Clients of OfflineDiagKernel can present data to the user in all three regions of the display. At the top window, positionally fixed info can be displayed in place of the menu, the latter of which will be automatically redisplayed after the test has completed. All the data fields can be updated dynamically. This is meant to display test set up information. The display has the following format:

```
<Display Title - optional>

    Item1  Item2  Item3 ...  (Items per row is limited by screen width)
    ...
    Itemn  Itemm  Itemz ...  (Number of rows is up to programmer)
```

Each display item can have:

a) Name position b) Name c) String value d) Numeric value and position

The facility to use is PutTestParameters:

```
PutTestParameters: PROCEDURE [
   parameters: LONG POINTER TO FixedPositionDisplayRecord ← NIL,
   upDateOnly:  BOOLEAN ← TRUE];  -- TRUE => Print values only.
```

When this is used, the remainder of the screen retains all its properties. The only restriction is that option tables can not be used, since it occupies the same physical space. If it is desireable to use option tables and positionally fixed displays, then the option table must be displayed at the top and the positionally fixed data displayed at the top of the Data Window via the facility DisplayFixedPositionData, which will be discribed later.

3.2 Option tables

Clients of OfflineDiagKernel can present option tables, one at a time, in place of the menu for client selection. An option table takes the following form:

```
<Option Table Title - optional>

    Option1   Option2   Option3  .... (Screen width is limiting factor)
    ....

    OptionN  OptionM   OptionZ  .... (Number of rows is up to programmer)
```

The facility to use is GetAnOption:

```
GetAnOption: PROCEDURE [optionTable: LONG POINTER TO OptionsRecord ← NIL,
             defaultOption: CARDINAL ← 0, -- When CR = the only input
             optionPrompt: LONG STRING ← NIL,  -- Prompt for input.
             optionHelp: LONG POINTER TO HelpText ← NIL,
             justDisplayTable: BOOLEAN ← FALSE]
             RETURNS [selectedOption: CARDINAL];
```

The option table, once passed to the kernel after initial use, remains valid until another option table is passed to the kernel via GetAnOption. This means that "optionTable" should be set to NIL after the first use of GetAnOption if the client intends to loop on the contends of the option table by using GetAnOption.

The option table is just a client-created menu. It behaves just like the kernel-generated menu.

3.3 Help texts

Help texts can be inserted into menus, every menu item, option tables, every item in an option table and for any instance of interaction with the user. They are all displayed in the Message Window. The global help text for a menu or option table is displayed when a lone question mark (?) is entered; the individual explanations are displayed when the selection number followed by a question mark are entered; interactive help texts are displayed by entering a lone question mark.

When users inputs an illegal entry, helpful prompts will be displayed to the user to assist the user to carry on.

3.4 Dynamic messages

The Message Window is a shared area which can be used to give progress and/or helpful information to the user any time. Any data can be output to it. The kernel also use this area for outputting error messages or helpful information to the users.

The facility to use is PutMessage:

```
PutMessage: PROCEDURE [message: LONG STRING ← NIL,
            beep: BOOLEAN ← FALSE,
            -- startWithNewLine TRUE invalidates spaceBeforePrinting
            startWithNewLine: BOOLEAN ← TRUE,  -- CRLF
            -- One can insert numOfBlankLine blank lines.
            numOfBlankLines: CARDINAL ← 0,
            blankSpaces: CARDINAL ← 1,
            --clearMessageAreaFirst TRUE invalidates startWithNewLine
            -- and spaceBeforePrinting
            clearMessageAreaFirst: BOOLEAN ← FALSE];
```

3.5 Positionally fixed test data

Clients of OfflineDiagKernel can present positionally fixed test data to the user at the top of the Data Window. The data fields can be updated dyamically. The display has the following format:

```
<Display Title - optional>

    Item1   Item2   Item3 ...   (Items per row is limited by screen width)
    ...

    Itemn   Itemm   Itemz ...   (Number of rows is up to programmer)
```

Each data item can have the following fields:

a) Name position b) Name c) String value d) Numeric value and position

The facility to use is DisplayFixedPositionData:

```
DisplayFixedPositionData: PROCEDURE
   [displayData: LONG POINTER TO FixedPositionDisplayRecord ← NIL,
    clearDataArea: BOOLEAN ← FALSE, -- TRUE clears before printing.
    upDateOnly:  BOOLEAN ← TRUE];  -- TRUE => Print values only.
```

While this is being displayed, the rest of the data window retains its properties, and PutData retains its full usage with one exception: PutData can not access the area occupied by DisplayFixedPositionData.

3.6 Random test data

Any data can be displayed in the Data Window by PutData, including a fixed heading. If a heading is displayed, it is displayed at the top of the Data Window or below the positionally fixed data if the latter exits. Automatical clearing of the window area, a kernel function, does not affect both the heading and positionally fixed data.

The Data Window is the private area of the client, who is guarrantteed of its exclusive usage.

The facility to use is PutData:

```
PutData: PROCEDURE [data: LONG STRING ← NIL, -- String to be printed.
                    -- numberAfterData = LAST[LONG CARDINAL] => no number to
                    -- be printed after the data STRING. Any other value will
                    -- be printed after data.
                    numberAfterData: LONG CARDINAL ← LAST[LONG CARDINAL],
                    dataAreaHeading: LONG STRING ← NIL, -- This is not cleared
                    -- clearHeadingAndData TRUE clears the entire Data Area
                    -- including dataAreaHeading. dataAreaHeading must be NIL.
                    clearHeadingAndData: BOOLEAN ← FALSE,
                    -- clearDataAreaOnly does not clear the heading. Only the
                    -- data below the heading is cleared.
                    clearDataAreaOnly: BOOLEAN ← FALSE,

                    -- Control variables for formatting dispaly data.
                    startWithNewLine: BOOLEAN ← FALSE, -- CRLF
                    numOfBlankLines: CARDINAL ← 0, --  Blank lines inserted
                    -- spaceBeforePrinting has effect only if startWithNewLine
                    -- and clearDataAreaOnly are both FALSE.
                    blankSpaces: CARDINAL ← 0,  -- Spaces before printing
                    -- xPosition = 0 => Let Control module and blankSpaces
                    -- specify the position on the line to print data.
                    -- Non-Zero xPosition overrides automatic positioning.
                    -- The Control Module will start printing at xPosition
                    -- after calculating the current line.
                    xPosition: CARDINAL ← 0, -- Position to start printing
                    -- Pause at the bottom of the Data Area so that the user
                    -- can review the data before they are cleared.
                    pauseAtBottomOfDataArea: BOOLEAN ← FALSE];
```

3.7 Information displayed automatically by the kernel modules

Information such as menus, subsystem under test and current test being run are automatically generated and displayed at the top of the screen. Test results - passed (P), failed (F), ambiguous (?) and none (blank) - are displayed, if applicable, in front of each menu item whenever a menu is printed or a test has returned. When at least one test has been run from a menu, a message indicating the last menu selection is displayed when the menu is entered.

4.0 Interactive facilities

Various facilities are provided by the kernel package to interact with the user. These allow the user to control the running of a diagnostic program and allows the diagnostic program to get runtime input from the users.

All interactions take place on the Main Prompt Line or the Auxiliary Prompt Line. The acive prompt is the one with the blinking cursor. All prompt line management is performed by the kernel.

4.1
```
GetYesNo PROCEDURE [prompt: LONG STRING ← NIL,
                    help: LONG POINTER TO HelpText ← NIL,
                    defaultSpecified: BOOLEAN ← FALSE,
                    default: BOOLEAN ← FALSE]
          RETURNS [YesReturnsTrue: BOOLEAN];
```

This outputs "Please enter Y(es) or N(o):" or the customized prompt supplied by the programmer on the Auxiliary Prompt Line. A "Y" or "y" returns TRUE; an "N" or "n" returns FALSE; and CR returns the default value. Any other input will cause "Please enter Y(es) or N(o)" to be issued by the kernel. The procedure will not return until a valid input is received, or the STOP key is entered to abort the operation in progress.

4.2
```
GetANumber: PROCEDURE [
             prompt: LONG STRING ← NIL,  -- Personalized prompt.
             help: LONG POINTER TO HelpText ← NIL,  -- Explanation
             lowLimit: LONG CARDINAL ← 1,
             upperLimit: LONG CARDINAL ← LAST [LONG CARDINAL],
             numberIsHexadecimal: BOOLEAN ← FALSE,
             numberIsLong: BOOLEAN ← FALSE,
             defaultNumber: LONG CARDINAL ← LAST[LONG CARDINAL]]
        RETURNS [longNumber: LONG CARDINAL,  -- 0 if number is not long
```

```
        -- number is 0 if numberIsLong is TRUE.
        number: CARDINAL,
        -- Directional default is FOREWARD.
        foreward: BOOLEAN ← TRUE, -- FALSE => reverse
        numberInStringFormat: LONG STRING ]; -- In specified base
```

The default prompts are "Please enter a hexadecimal number" or "Please enter a decimal number", depending on the BOOLEAN numberIsHexadecimal. It is printed on the Auxiliary Prompt Line.

This procedure returns a number as specified by the programmer. The procedure returns only when the user enters a vaild number as specified in the input parameters.

This procedure is also very useful for creating small option tables that fits on a line. This can be done by creating a STRING enumerating the options. Bounds can be set appropriately to ensure that the option returned is allowed. For example, the following string... let's say, optionString...

```
  "Please select bad page disposition[ 1-Ignore  2-Mark Bad  3-Repair ]"
```

can be passed as the "prompt" argument in GetANumber as follows:

```
   userSelectedOption ← GetANumber[prompt: optionString,
                                   upperLimit: 3,
                                   defaultNumber: 3]
```

The program can take the appropriate action depending on userSelectedOption.

4.3 GetAnOption:

```
PROCEDURE [
        optionTable: LONG POINTER TO OptionsRecord ← NIL,
        defaultOption: CARDINAL ← 0, -- When CR = the only input
        optionPrompt: LONG STRING ← NIL,  -- Prompt for input.
        optionHelp: LONG POINTER TO HelpText ← NIL,
        justDisplayTable: BOOLEAN ← FALSE]
    RETURNS [selectedOption: CARDINAL];
```

The optionPrompt is printed on the Main Prompt Line. This procedure returns only if the option selection is a vaild option in the optionTable; ie. it scans the optionTable for the existence of the option in the table before returning. The message "No such option" is printed on the message area if the user input is not in the table.

4.4 GetAString:

```
PROCEDURE [prompt: LONG STRING ← NIL, -- Personalized prompt
           defaultString: LONG STRING ← NIL,
           help: LONG POINTER TO HelpText ← NIL, -- Help
           echoWithStar: BOOLEAN ← FALSE]
    RETURNS [LONG STRING];
```

GetAString is for getting a string input, such as a host name, from the user. This procedure should not be used as a means to get a command from the user. All commands should be hidden behind menu/option selections. The idea is:

> What you see is what you get... And if you don't see it, an erroneous input or entering a question mark should give you what you to go on.

4.5 HitAnyKeyToContinue:

```
PROCEDURE [
        prompt: LONG STRING ← NIL,
        beep: BOOLEAN ← TRUE];
```

HitAnyKeyToContinue issues "Enter any key to continue:" or prompt: on the Auxiliary Prompt Line. It temporarily pauses the test until the user enters a key. If the key is the STOP key, the test is terminated; all other keys cause the test to continue at the point of pause.

5.0 Transfer of control

An offline diagnostic package is a boot file made from a bcd created by binding the client modules with OfflinfDiagKernel<Dove or DLion>.bcd. Once booted, the user must go through System Configuration Verification, Login and Subsystem Selection screens before getting to the client menus. Once

there, operation is straight forward. User selects a menu item for execution. If the selection is a genuine test, it is executed; if the selection selects another menu, a new menu is generated based on the user security level and displayed on the screen for user selection.

Whenever a menu selection is a test, control is passed to the client program until, normally, the test runs to a point and RETURNs with a test result. In addition to this CALL/RETURN transfer mechanism, the kernel provides several additional transfer facilities. These include the STOP key, a signal and two procedures.

5.1 The signal AbortCurrentTest

The basic non-CALL/RETURN transfer mechanism between the kernel and client packages is implemented by a PUBLIC signal:

```
AbortCurrentTest: SIGNAL
```

This signal can be intercepted by the client, and therefore, can be used as an internal global control transfer mechanism within the client's modules. If the client should REJECT this signal, then control is transfer back to the kernel, which interprets it as a genuine request for aborting the test. It then proceeds to clean up its internal states, clean up the screen as needed and redisplay the menu if the client has used the menu area for displaying option tables and/or positionally fixed test parameters. If the signal causes a complete exit from the top level menu of the client package, the user is given the following prompt:

"Test data will be deleted upon exit. Is this OK ?"

If the reponse is "Y" or "y" all the client heaps are delete, the client package is exited and control is given to the Subsystem Selection Menu or the Login Menu depending on whether there are more than one client packages present or not.

5.2 STOP key

Whenever the user is being prompted for a response, the STOP key can always be entered. The consequence of hitting the Abort/Stop Key, as far as the kernel is concerned, is one of the following:

1) If the user is running a test, the test is aborted. The menu is redisplayed if the aborted test hogged the entire screen.
2) If the client is in the process of selecting a menu item, then the parent menu will be re-entered after exiting the current menu. In this environment, it is used as a means to backtrack up the menu tree.

The internal mechanism used is the signal AbortCurrentTest. Therefore, if the STOP key is depressed while still in the client context, the client can intercept this signal and perform whatever local operation it desires.

5.3 LookForAbort

LookForAbort is meant to be used in a client loop which permits the random entry of the STOP key. Whenever the loop detects the depression of the STOP key, the kernel raises AbortCurrentTest, which the client can again intercept and/or reject. A note of caution: This does not always work if something in the loop disables the STOP key or hogs all the CPU cycles.

5.4 HitAnyKeyToContinue: PROCEDURE [prompt: LONG STRING ← NIL, beep: BOOLEAN ← TRUE];

This permits a client to pause its tests. The user can then choose to abort by entering the STOP key or continue by entering any other key. In the case of the STOP key, the signal AbortCurrentTest is again raised.

6.0 References

For those who need further details, please refer to the following program files:

Dove Offline Diagnostics - User Interfaces (Out of date. Original doc)
OfflineDiagInterface.mesa - Up to the minute details. Best reference.
OfflineDiagInterfaceExtra.mesa - For adding clients and misc stuffs
OfflineDiagnosticControlModule.mesa - Control Module (UtilityPilotClient)
OFLFloppyExecImpl.mesa - Floppy Disk Executive
OfflineDiagnosticVersionImpl.mesa - Contains the client version definition
OfflineDiagTTYDove.mesa - Bitmap screen interface
OfflineDiagTTYImplDove.mesa
CMDiagMsgKeysDove.mesa - Message files used by the kernel modules
CMDiagMsgKeysImplDove.mesa

LOG
Created on 19-Jun-85 by KL
8-Jan-86: Added all the updates for 12.2 (1.2)

1. The Courier Compiler

Overview of Operations

The Mesa Courier compiler translates Courier source files into a set of Mesa[2] source files that implement the protocol defined by the Courier source file. Courier is a remote procedure call language defined in Courier, The Remote Procedure Call Protocol[1]. To make the translation from Courier to Mesa more simple, the Mesa Courier compiler uses slightly different grammar than specified in [1] (see section 1.5).

This document describes how to operate the compiler, how to use the Mesa source files that result, and describes the differences between the language the compiler understands and the language specified in [1].

In the descriptions that follow, these terms are significant:

Client: refers to the active system element in the communication model presented in section 1.2 of [1].

Server: refers to the passive system element in the communication model presented in section 1.2 of [1].

A note on terminology: The term compiler refers to the Courier compiler. Any other compiler mentioned will be fully qualified, i.e., the Mesa compiler.

1.1 Files You Need to Begin

To use the Courier compiler, you need the Courier compiler and your Courier source program. The Courier compiler is called CCompiler.bcd. You will need the Mesa programming tools [3] (compiler, binder, etc) to create the server and client programs.

Note: To understand the description of the files the Courier compiler generates a few terms need to be defined. Figure 1.1 illustrates the flow of information in a remote program from the application program to the server program; each block is described below.

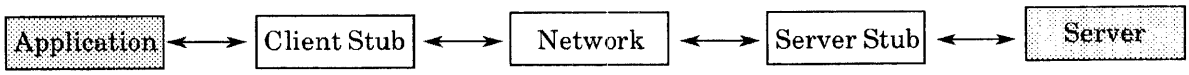


Figure 1.1. Information flow

The application program and the server program (shaded grey) are programs written by the person using the Courier compiler. The client stub and the server stub are programs written by the Courier compiler.

1.2 Using the Compiler (Setting it up)

The Courier compiler runs in the Executive and is invoked when you type the command:

>CCompiler sourcefile1 sourcefile2 ... sourcefilen

in the executive window.

If you supply a source file with no extension, then the extension .courier is assumed by the compiler. Any error messages or diagnostics are written to the log file CCompiler.log. Given the source file Foo.courier, the Executive command to compile this Courier source program is:

>CCompiler Foo.courier

The Courier compiler generates the Mesa source files Foo.mesa, FooCourier.mesa, FooDescription.mesa, FooClientImpl.mesa, FooServerImpl.mesa and Foo.cm. The files are described below.

Foo.mesa Foo.mesa is the public Mesa definitions module that the compiler creates. It contains: 1) the complete set of Mesa data structure defined in the Courier source; and 2) the Mesa procedure, error, and constant definitions. This module is exported by the client stub and imported by the server stub.

FooCourier.mesa FooCourier.mesa is the private Mesa definitions module that the compiler creates. It contains constants, types, and procedures needed by the client and server stub.

Note: This definitions module should only be used by the client and server stubs. Application and server programs should not reference any of the types or procedures defined in this module.

FooDescription.mesa FooDescription.mesa is the Mesa program module that contains the Courier description routines [3] needed to serialize and deserialize Mesa data structure to the wire. This module is included in both the server and client configurations.

FooClientImpl.mesa FooClientImpl.mesa implements the client stub that in turn makes remote procedure calls to the remote program. This module exports the Foo.mesa interface. This module is included in the client configuration.

FooServerImpl.mesa FooServerImpl.mesa implements the server stub that accepts remote procedure calls from the client stub. This module imports the Foo.mesa interface. This module is included in the server configuration.

Foo.cm Foo.cm is a command file to rebuild the source files generated by the Courier compiler.

1.3 Building Remote Programs

After you compile the Courier source file, use the Mesa compiler to compile the resulting output files in the following order: Foo.mesa FooCourier.mesa, FooDescription.mesa, FooClientImpl.mesa, and FooServerImpl.mesa. If there are errors after you do the Mesa compilation, then change the Courier source file to fix the problems in the Mesa output files. Repeat this process until the Courier compiler output files successfully compile. This process will be apparent after you read the next sections.

When the five generated modules compile, you can write your application and server program. The application program should import the Foo.bcd definition module. The server implementation should export the Foo.bcd definition module.

For the application program to make remote procedure calls to the server, it must be bound with the FooDescription.bcd and FooClientImpl.bcd files using the Mesa binder. For a service to export a remote program; that is, accept remote procedure calls, the service must be bound to FooDescription.bcd and FooServerImpl.bcd using the Mesa binder. An example of the configuration files needed to build a client and a server is given in section 1.6.

1.4 Using the Remote Program

The user's programs call into the generated client and server stub to make and accept remote procedure calls. The next sections explain how the application programs make remote procedure calls, and how server programs accept them.

1.4.1 Remote Binding (Application)

To make a remote procedure call, an application program must bind to a remote program. You do this by calling the Foo.RemoteBind procedure in the Foo.bcd definition module:

```
Foo.RemoteBind: PROCEDURE[
  host: System.NetworkAddress, zone: UNCOUNTED ZONE ← NIL]
  RETURNS[bH: Foo.BindHandle];
```

This procedure returns a **Foo.BindHandle** that is used in calls to the **Foo** interface. The **host** is the network address of the remote program. If the **zone** is unspecified, then **Heap.systemZone** is used. Anytime a procedure returns a pointer (or a descriptor) to a data structure, the data structure is allocated from this zone. It is the application program's responsibility to free this data when it is finished with the data. Freeing returned data structures is explained in detail in section 1.4.3.

When an application program is finished with the remote program, it then unbinds itself from that program by calling:

```
Foo.RemoteUnbind: PROCEDURE[
  bH: Foo.BindHandle] RETURNS [nil: Foo.BindHandle];
```

This procedure frees up any resources used to make the remote procedure calls. It always returns a **Foo.BindHandle** whose value is **NIL**.

1.4.2 Remote Binding (Server)

The server program semantics of **Foo.RemoteBind** and **Foo.RemoteUnbind** procedures are different from the application program's. When the server program calls **Foo.RemoteBind**, the server stub exports the remote program. The **host** argument is ignored by the server stub. The server stub uses the **zone** argument to create the data structures that are passed to the server program. Data structures allocated by the server program are freed by the server stub (explained in section 1.4.3).

When the server stub receives a remote procedure call, it calls the server programs procedure that corresponds to the current remote procedure call. The **Foo.BindHandle** passed to the server program is always be **NIL**.

To take a remote program off the network; that is, to unexport the remote program, the server must call **Foo.RemoteUnbind**. After a remote program is unexported, no further call is made into the server program until the server calls **Foo.RemoteBind**. Any remote procedure calls in progress are unaffected.

1.4.3 Freeing Allocated Data Structures

The implementors of the Foo interface must sometime allocate data from a heap. To free this data, the Foo interface has data freeing procedures. Suppose a Courier source program Foo.courier contained the following declaration:

```
NameList: TYPE = ARRAY 25 OF STRING;
  GetNameList: PROCEDURE RETURNS [nameList: NameList] = 1;
```

The corresponding public definitions file Foo.mesa would have the following:

```
Foo.NameList: TYPE = ARRAY [0..25) OF LONG STRING;
Foo.GetNameList: PROC RETURNS [nameList: LONG POINTER To Foo.NameList];
```

If an application program calls **Foo.GetNameList**, then it must free the data structure returned by the client stub. To free this data, the application program can call the procedure defined in the public definitions module:

```
Foo.FreeGetNameListResult: PROCEDURE[
  bH: Foo.BindHandle, nameList: LONG POINTER TO Foo.NameList];
```

On the server side, the server program may return a result that has data allocated from a heap. The server stub calls **Foo.FreeGetNameListResult** to free the data structures that the server program allocated.

1.4.4 Using Bulk Data Transfer

Two predefined types, **SOURCE** and **SINK**, have been added to the Courier language to support bulk data transfer.Suppose a courier program Foo.courier contained the following procedure:

```
Receive: PROCEDURE[name: STRING, data: SINK] = 2;
```

The resulting Mesa declaration is:

```
Foo.Receive: PROCEDURE[name: LONG STRING, data: Stream.Handle] ;
```

The application program passes a stream to the client stub for the bulk data transfer. With a **SINK** type, the stream passed to the client stub will receive data. If a **SOURCE** type is used for bulk data, the stream passed to the client stub will send data. The client stub will check the streams options to determine if the end of data is marked by a signal or returnedstatus. The client stub will not delete the stream at any time.

On the server side, the server program is passed a stream. For a **SINK** type, the server program sends data to the stream. After the data is transferred, the server program must NOT delete the stream. For a **SOURCE** type, the server program reads data from the stream. The stream can either signal a Stream.EndOfStream or return end of stream status, as determined by the stream's options.

1.4.5 Using Remote Errors

The Courier language allows for the specification of remote errors. With the model the Courier compiler uses for remote procedure calls, only the server program may raise a remote error, and the application program must be willing to catch them.

Courier error declarations are translated to Mesa error declarations by the Courier compiler. For a server program to raise a remote error it need only raise the translated Mesa error in the public definitions file. This Mesa error is converted into a Courier error by the server stub. The client stub on the other end will then take the Courier error and convert it into the orginal Mesa error for the application program.

An example of using remote errors is in section 1.6.

1.5 Courier Grammar Changes

The main change in the Courier grammar is the restrictive use of constructed data types. Constructed data types are data types that contain other data types. The restriction placed by the Courier compiler is that constructed data types must be composed of pre-defined types and referenced types. Constructed types may not include other constructed types.

In [1] the following is a legal statement:

```
PrintAttribute: TYPE = SEQUENCE 10 OF RECORD[  -- illegal
   name: STRING,
   date: [year, month, day: CARDINAL],
   length: INTEGER];
```

To get a valid declaration with the same semantics using the Courier compiler, this type must be stated as:

```
PrintAttribute: TYPE = SEQUENCE 10 OF PrintRecord;  -- legal
PrintRecord: TYPE = RECORD[
   name: STRING,
   date: DateRecord,
   length: INTEGER];
DateRecord: TYPE = RECORD[year, month, day: CARDINAL];
```

Though the second form of the example is more cumbersome for the programmer, the translation to Mesa is easier for the Courier compiler. With the first example, the error message would be (in CCompiler.log):

```
Mesa 12.2 Courier Compiler of 6-Mar-86 18:03:16
Foo.
 PrintAttribute: TYPE = SEQUENCE 10 OF RECORD[
 . . . . . ^ Syntax Error [130]

No recovery found.
```

```
lines: 13, errors: 1, time: 7
```

The grammar is different from the grammar in [1] in the following ways:

Courier Grammar specified in [1]

```
Type            ::= PredefinedType | ConstructedType
                    | ReferencedType
ConstructedType ::= ARRAY NumericValue OF Type |
                    SEQUENCE MaximumNumber OF Type
Candidate       ::= DesignatorList => Type
Field           ::= NameList : Type
```

The Courier compiler's grammar:

```
Type            ::= SimpleType | ConstructedType
SimpleType      ::= PredefinedType | ReferencedType
ConstructedType ::= ARRAY NumericValue OF SimpleType
                    SEQUENCE MaximumNumber OF SimpleType
Candidate       ::= DesignatorList => SimpleType
Field           ::= NameList : SimpleType
```

1.5.1 Bulk Data Transfer

Two new predefined types have been added to the Courier language to support the Bulk Data Transfer protocol of Courier. The new types, SOURCE and SINK are used as procedure argument types for the bulk data source and sink abstractions.

The new grammar for predefined types is:

```
PredefinedType ::= BOOLEAN | CARDINAL | LONG CARDINAL | INTEGER |
                   LONG INTEGER | STRING | UNSPECIFIED | SOURCE |
                   SINK
```

1.6 An Example of a Courier Program

This is a factorial server program. The Courier source files is as follows:

Courier Program

```
-- Factorial.courier
Factorial: PROGRAM 999 VERSION 1 =
    BEGIN
    Compute: PROCEDURE[num: INTEGER] RETURNS [result: LONG CARDINAL]
        REPORTS [Error] = 1;
    ErrorCode: TYPE = {
        numberTooLarge(0), negativeNotAllowed(1)};
    Error: ERROR [code: ErrorCode] = 1;
    END.
```

The application program uses the resulting client stub as follows:

Application Program

```
-- FactorialToolImpl.mesa
DoIt: FormSW.ProcType = {
  addr: System.NetworkAddress;
  answer: LONG CARDINAL;
  bH: Factorial.BindHandle ← NIL;
  BEGIN
  addr ← AddressTranslation.StringToNetworkAddress[data.host!
    AddressTranslation.Error => {
      AddressTranslation.PrintError[errorRecord, Msg];
      GOTO nope}].addr;
  bH ← Factorial.RemoteBind[addr];
  answer ← Factorial.Compute[bH, data.number!
    Factorial.Error => {
      SELECT code FROM
        numberTooLarge => Msg["Number Too Large.\n"L];
        negativeNotAllowed => Msg[
          "Negative numbers not allowed.\n"L];
      ENDCASE;
      GOTO nope}];
  Format.Number[Write, data.number, []];
  Write[" factorial is "L];
  Format.LongNumber[Write, answer, []];
  Write["\n"L];
  EXITS nope => NULL;
  END;
  IF bH # NIL THEN bH ← Factorial.RemoteUnbind[bH];
  };
```

The server program is as follows:

Server Program

```
-- FactorialImpl.mesa
DIRECTORY Exec, Factorial, System;
FactorialImpl: PROGRAM
  IMPORTS Exec, Factorial
  EXPORTS Factorial = PUBLIC {
  OPEN Factorial;
  Error: ERROR [bH: BindHandle, code: ErrorCode] = CODE;
  bH: BindHandle ← NIL;
  Compute: PROCEDURE[
    bH: BindHandle, num: INTEGER]
    RETURNS [result: LONG CARDINAL] = {
    IF num < 0 THEN Error[bH, negativeNotAllowed];
    IF num > 100 THEN Error[bH, numberTooLarge];
    result ← 1;
    FOR i: INTEGER IN [1..num] DO
      result ← result * i;
      ENDLOOP;
    };
  Main: Exec.ExecProc = {};
  Unload: Exec.ExecProc = {
    h.RemoveCommand["FactorialServer.~"L];
    bH ← Factorial.RemoteUnbind[bH];
```

```
    };
  Init: PROCEDURE = {
    bH ← Factorial.RemoteBind[System.nullNetworkAddress];
    Exec.AddCommand[
      name: "FactorialServer.~"L, proc: Main, unload: Unload];
    };
  }.
```

The application and server configurations look as follows:

Server Configuration

```
-- FactorialServer.config
FactorialServer: CONFIG
  IMPORTS Exec, Courier, Heap CONTROL FactorialImpl = {
  FactorialImpl;
  FactorialServerImpl;
  FactorialDescription;
  }.
```

Application Configuration

```
-- FactorialClient.config
FactorialClient: CONFIG
  IMPORTS
    Courier, Heap, AddressTranslation,
    Format, FormSW, Heap, Put, String, Tool
  CONTROL FactorialToolImpl = {
  FactorialToolImpl;
  FactorialClientImpl;
  FactorialDescription;
  }.
```

1.7 Restrictions

The following restrictions apply to various Courier statements that successfully compile and represent either runtime or Mesa compile time errors.

1.7.1 Type and Constant Restrictions

Variant Records

Tag types of variant records must start at 0 and have no holes, that is, each value must be exactly one greater that the previous value. The following declarations will not work with the current implementation of Courier in Pilot:

```
TagType: TYPE = {red(2), white(4), blue(6)};  -- illegal
Flag: TYPE = CHOICE OF {
  red => INTEGER,
  white => CARDINAL,
  blue => STRING};
```

The correct declaration of TagType is:

```
TagType: TYPE = {red(0), white(1), blue(2)};  -- legal
```

Procedure Types Procedure types are not currently supported. The declaration:

```
ProcType: TYPE = PROCEDURE[a, b: INTEGER];  -- illegal
DoIt: ProcType = 1;
```

must be declared as:

```
DoIt: PROCEDURE[a, b: INTEGER];  -- legal
```

Sequence constants Sequence constants are not allowed. The following is an illegal declaration:

```
SeqConst: SEQUENCE 3 OF INTEGER = [1, 2, 3];  -- illegal
```

Array constants Array constants are allowed, however the Courier compiler will not type check the constant. The following declaration will compile but be incorrect:

```
ArrayConst: ARRAY 10 OF INTEGER = [1,2];  -- illegal
```

1.7.2 Importing Courier Programs

Currently, importing **ARRAY**, **RECORD**, **CHOICE**, and **LONG STRING** types from another Courier program will result in incorrect code. For more information, see section 1.7.3

1.7.3 Future Enhancements

Importing Courier Types For the Courier compiler to correctly deal with imported Courier types, it must generate a symbol table data file. The symbol table of the imported Courier source must be present when compiling a program that imports a Courier program.

Importing Mesa Types Importing existing Mesa types from existing Mesa definitions modules is a difficult problem; here are two ways to solve it.

The first way, though easier to implement, is more difficult for the Courier programmer. The programmer supplies the Courier compiler with a parameter (text) file, that includes the imported Mesa types used and their respective Courier description routines.

The second way is easier on the Courier programmer, but much more difficult for the Courier compiler. The Courier compiler opens the Mesa definitions module, reads the symbol table, and generates the correct Courier description routines. This solution is similar to Bruce Nelson's Diplomat[4] program for the Envoy[6] remote procedure call protocol.

Protocol Encapsulation Some protocol designers like to use encapsulated protocols (Filing). Below is an example:

If an attribute's **type** value is **checksum** (0) , then the **value** field of the attribute is interpreted as a **Checksum** (**CARDINAL**). Likewise, if the **type** value is **createdby** (1), then the **value** field is interpreted as a **CreateBy** types.

```
Attribute: RECORD[
   type: AttributeType, value: SEQUENCE OF UNSPECIFIED];
-- interpreted attribute definitions
checksum: AttributeType = 0;
Checksum: TYPE = CARDINAL;
createdby: AttributeType = 1;
CreateBy: TYPE = User;
```

This type of protocol encapsulation is not defined by the current Courier language. To implement this, a change to the Courier language is required.

1.8 References

[1] Courier: The Remote Procedure Call Protocol. Xerox System Integration Standard. Stamford, Connecticut; 1981 December; XSIS 038112

[2] Mesa Language Manual, Version 11.0. Xerox Office Systems Division. May 1984.

[3] Mesa User's Guide, Version 10.0. Xerox Office Systems Division. February 1983.

[4] Nelson, Bruce. Diplomat: Attache To Envoy. Xerox Office Products Division. Palo Alto, California; August 1980.

[5] Pilot Programmer's Manual, Version 11.0. Xerox Office Systems Division. May 1984.

[6] White, Jim. Envoy Functional Specification, Version 4.0. March 1980.

Listing of hacks on [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>
Documentation from [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>
Listing created 16-Mar-89 17:48:38

	Hack Name	Last Writer	Create Date
1.	Activity.bcd	Stephen B. Tom:OSBU North:Xerox	13-Aug-85 11:38:24
2.	BKG.bcd	Stephen B. Tom:OSBU North:Xerox	13-Dec-85 14:41:34
3.	BlackJack.bcd	Franklin Lee Yien:OSBU North:Xerox	6-Aug-86 17:37:22
4.	BrushDMT.bcd	Stephen B. Tom:OSBU North:Xerox	13-Aug 85 10:44:11
5.	BWSActivity.bcd	W. Dale Knutsen:OSBU North:Xerox	23-May-86 13:26:56
6.	BWSAddressTranslation.bcd	Bryan Yamamoto:OSBU North:Xerox	2-Apr-85 17:56:51
7.	BWSAddressTranslationImpl.bcd	Bryan Yamamoto:OSBU North:Xerox	30-Jul-85 9:44:43
8.	BWSBrushDisplay.bcd	Perry A. Caro:OSBU North:Xerox	23-Oct-85 14:31:53
9.	BWSCompiler.bcd	Lori S. Nagata:OSBU North:Xerox	3-Jul-86 17:26:32
10.	BWSFgrep.bcd	William H. McCoy:OSBU North:Xerox	18-Aug-85 15:28:47
11.	BWSHeapCheckTool.bcd	Lee F. Breisacher:OSBU South:Xerox	20-Aug-85 11:33:59
12.	BWSIconEditor.bcd	Gerard S. Zonus:OSBU South:Xerox	6-Aug-85 15:32:58
13.	BWSLibrarianTool.bcd	Brian T. Lewis:OSBU North:Xerox	14-Feb-86 13:51:11
14.	BWSMahJong.bcd	Bruce S. Lee:OSBU South:Xerox	24-Jul-85 14:27:04
15.	BWSPowerMouse.bcd	Jeremy M. Goodell:OSBU South:Xerox	9-Jul-85 11:33:35
16.	BWSSol.bcd	Gerard S. Zonus:OSBU South:Xerox	25-Jul-86 16:34:41
17.	BWSStatusIcon.bcd	Stephen W. Bartlett:OSBU South:Xerox	11-Jan-85 14:26:30
18.	BWSTDFParser.bcd	Richard Balcon:SBD-E:RX	13-Jan-86 6:19:16
19.	BWSTest.bcd	Sybil A. Johnson:OSBU South:Xerox	29-May-86 18:33:27
20.	CartoonTool.bcd	Bryan Yamamoto:OSBU North:Xerox	20-Jan-86 18:01:02
21.	CheckCursor.bcd	Bruce K. Whittaker:OSBU North:Xerox	18-Jul-85 13:56:06
22.	CHIcon.bcd	Nanette E. Harter:OSBU North:Xerox	14-Mar 86 11:19:55
23.	CreateCanvas.bcd	Donald W. Gillies:OSBU North:Xerox	18-Jul-86 20:18:53
24.	CreateChar.bcd	Deborah J. Lewis:OSBU South:Xerox	9-Oct-86 19:33:03
25.	Cruncher.bcd	Perry A. Caro:OSBU North:Xerox	20 Nov-85 13:51:57
26.	CuFilmTool.bcd	Deborah J. Lewis:OSBU South:Xerox	26-Jul-85 12:13:18
27.	DefaultIconProps.bcd	James G. Sandman:OSBU North:Xerox	12-Aug-85 13:43:00
28.	DefaultIconPSheet.bcd	J. Paul Holbrook:OSBU South:Xerox	18-Dec-84 15:45:24
29.	DesktopIconPictureImpl.bcd	Gerard S. Zonus:OSBU South:Xerox	16-Jan-85 15:27:07
30.	DocumentStatistics.bcd	Bruce S. Lee:OSBU South:Xerox	11-Dec-85 13:08:08
31.	DoneLoading.bcd	J. Paul Holbrook:OSBU South:Xerox	25-Jan-85 13:47:57
32.	Drinks.bcd	Mark K. Hahn:OSBU North:Xerox	1-May-85 13:26:22
33.	FFF.bcd	Lee F. Breisacher:OSBU South:Xerox	6-Jan-86 13:29:02
34.	FileProps.bcd	Brian T. Lewis:OSBU North:Xerox	26-Sep-85 13:39:05
35.	FileTimeout.bcd	Russell Sonnenschein:OSBU South:Xerox	6-Sep-85 15:52:10
36.	FilmConverter.bcd	Deborah J. Lewis:OSBU South:Xerox	26-Jul-85 15:06:09
37.	FlushOut.bcd	J. Paul Holbrook:OSBU South:Xerox	7-Mar-85 13:30:26
38.	FlySwatter.bcd	Mark K. Hahn:OSBU North:Xerox	2-May-85 12:19:38
39.	FontedTextEditor2.bcd	Alfredo E. Camacho:OSBU South:Xerox	5-Jun-85 11:06:32
40.	FormWindowLayoutTool.bcd	Randy Gobbel:OSBU North:Xerox	30-Apr-85 16:54:30
41.	IconFileImpl.bcd	Gerard S. Zonus:OSBU South:Xerox	16-Jan-85 15:25:12
42.	IPressStream.bcd	Bruce K. Whittaker:OSBU North:Xerox	7-Sep-85 18:30:39
43.	KeepWindows.bcd	Harold J. Shinsato:OSBU North:Xerox	16-May-85 12:17:26
44.	Life.bcd	Gerard S. Zonus:OSBU South:Xerox	26-Feb-85 17:15:37
45.	MahJong.bcd	Bruce S. Lee:OSBU South:Xerox	1-Apr-85 17:07:03
46.	MakeIPFileType.bcd	Frank H. Bowers:OSBU North:Xerox	25-Feb-86 17:20:31
47.	MakeRESFileType.bcd	Darrel E. Strom:OSBU South:Xerox	20-Jan-86 10:55:09
48.	Mazewar.bcd	Bryan Yamamoto:OSBU North:Xerox	2-Apr-85 19:58:59
49.	MJ.bcd	Stephen B. Tom:OSBU North:Xerox	4-Mar-85 19:27:07
50.	OpenAsFolder.bcd	James G. Sandman:OSBU North:Xerox	13-Nov-85 14:56:41
51.	PacMan.bcd	Stephen B. Tom:OSBU North:Xerox	28-Aug-85 15:21:25
52.	PasswordTool.bcd	Robert Sperry:WBST128:Xerox	10-Mar-86 7:43:53
53.	PermanentZONE.bcd	Steven W. Epp:OSBU North:Xerox	5-Nov-85 10:40:48
54.	PermanentZONEImpl.bcd	Steven W. Epp:OSBU North:Xerox	1-May-86 14:16:51
55.	PupFetchTool.bcd	James G. Sandman:OSBU North:Xerox	12-Aug-85 16:41:10
56.	RegisteredPSheet.bcd	Jeffrey A. Johnson:OSBU North:Xerox	25-Jul-86 14:22:19
57.	RegisteredPSheetImpl.bcd	Jeffrey A. Johnson:OSBU North:Xerox	29-Jul-86 9:39:56
58.	Ripples.bcd	Stephen B. Tom:OSBU North:Xerox	25-Feb 85 20:43:45
59.	RootPicture.bcd	Stephen B. Tom:OSBU North:Xerox	14-Mar 86 13:44:32
60.	RootPuzzle.bcd	Stephen B. Tom:OSBU North:Xerox	14 Mar 86 14:17:03
61.	ScreenCamera.bcd	Darrel E. Strom:OSBU South:Xerox	25-Sep-85 14:52:44
62.	ScrollBar.bcd	Bruce K. Whittaker:OSBU North:Xerox	22 Sep-85 14:17:52
63.	ShowChars.bcd	Eric R. Mader:OSBU South:Xerox	7-Aug 85 9:28:33
64.	ShowFile.bcd	Franz-Josef Denig:Visitors PA:Xerox	Visitors 6 Jun-85 10:44:53
65.	ShowVM.bcd	Brian T. Lewis:OSBU North:Xerox	6-Mar 85 17:32:42
66.	SortMenu.bcd	Lee F. Breisacher:OSBU South:Xerox	2 Aug 85 16:08:16
67.	SortMenuConfig.bcd	Lee F. Breisacher:OSBU South:Xerox	12 Aug 85 9:06:34
68.	SpecialDocumentEditor.bcd	Richard Balcon:SBD-E:RX	14-Jan 86 6:51:38
69.	StarT.bcd	Sybil A. Johnson:OSBU South:Xerox	27 Jan 85 13:12:55
70.	SwitchFolderType.bcd	John A. Davidson:Roch0846:Xerox	26-Feb-88 9:25:21
71.	TableWindow.bcd	Lee F. Breisacher:OSBU South:Xerox	21 Nov 84 15:07:01
72.	TableWindows.bcd	Lee F. Breisacher:OSBU South:Xerox	18-Dec-84 10:08:13
73.	TestBWSAT.bcd	Bryan Yamamoto:OSBU North:Xerox	30-Jul-85 9:59:59
74.	TipTest.bcd	Stephen B. Tom:OSBU North:Xerox	13-Aug-85 16:52:06
75.	VMStats.bcd	James G. Sandman:OSBU North:Xerox	3-Mar-86 9 41:52
76.	XDEKeyboard.bcd	Janie D. Phillips:OSBU South:Xerox	10-Sep-85 12:11:21
77.	XStringPrinter.bcd	Lee F. Breisacher:OSBU South:Xerox	5-Oct 84 16.00:41
78.	XStringTableWindow.bcd	Lee F. Breisacher:OSBU South:Xerox	21 Nov 84 15:07:50

1. Activity.bcd Stephen B. Tom:OSBU North:Xerox 13-Aug 85 11:38:24
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>Activity.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>Activity.doc of 13-Aug-85 11:41:00
Description:
Document name: Activity.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA 13-Aug-85 11:40:58
Owner: Tom:PA

Activity registers the menu command, "Toggle Activity" in the Attention window and as the name suggests, acts like a toggle switch. Invoking the menu command will bring up the window, and invoking it again will close.

What is Required: Activity.bcd

Activity is an outgrowth of CpuMonitor and StorageFaultMonitor. Using a very small window, it displays two bar graphs that indicate machine activity. One graph shows the percentage of cpu utilization. The other graph shows a count of either disk IO operations or page faults. The Activity window itself will be unobtrusively positioned at the lower left corner of the screen.

The graphs are updated once per second and give the instantaneous activity (for that second) in black, and an average activity over the past 10 seconds in gray. The cpu utilization percentage and IO activity for the second are a...

---

2. BKG.bcd Stephen B. Tom:OSBU North:Xerox 13-Dec-85 14:41:34
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BKG.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BKG.doc of 13-Dec-85 15:31:01
Description:
Document name: BKG.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA 13-Dec-85 15:31:01
Owner: Tom:PA

BKG is a two-player backgammon game played over the network through two workstations (any Dandelion or Daybreak combination). Deserved credit goes to Charles Haynes who originally wrote the Tajo prototype. To start the program, both players should run BKG.bcd either from the ViewPoint Loader or Command Central.

What is Required: BKG.bcd

Note: If the "O" boot switch is used the following data file is needed:
Data file: [Alt:OSBU North:Xerox]BWSHacks/4.0/Data/Backgammon.TIPC

BKG registers the menu command, "Backgammon" in the Attention window. By evoking the menu item, a backgammon board will appear on the screen. After both players have a board up, the match can commence.

There is a form subwindow on the top of the board which resembles the following:

Color: [BLACK|WHITE]
Start...

---

3. BlackJack.bcd Franklin Lee Yien:OSBU North:Xerox 6-Aug-86 17:37:22
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BlackJack.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BlackJack.doc of 7-May-85 16:20:01
Description:
-- BlackJack.doc
-- Frank Yien 7-May-85 16:20:00

BlackJack pits you against the dealer in this popular casino game. When you select "Deal," the dealer will deal two cards to you and two cards to himself. One of the dealer's cards will be face up. The object of the game is to get closer to a point total of 21 than the dealer without going over 21. Cards 2 thru 9 are worth their face value; 10's and all picture cards are worth 10 points; aces are worth 1 or 11 points, depending on which would be the most advantageous for the person holding that card. The dealer plays by a pre-determined set of rules (discussed later).

The parameters that you can control are "Number of Decks," "Initial Money," and your amount bet. When "New Table" is selected (or when the tool is initially run), the dealer will use at the number of decks selected, shuffle the cards, and give you the initial money for you to bet with. You place a bet by using...

---

4. BrushDMT.bcd Stephen B. Tom:OSBU North:Xerox 13-Aug-85 10:44:11
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BrushDMT.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BrushDMT.doc of 25-Feb-85 19:13:13
Description:
Document name: BrushDMT.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA 25-Feb-85 19:11:39
Owner: Tom:PA

BrushDMT registers the menu command, "BrushDMT" in the Attention window. By toggling the menu item, a tiny window located in the upper righthand side of the screen will be activated/deactivated. Chording over this window will invoke a menu listing all "*.brush" files in the system catalog. Selecting a particular choice will display the corresponding brush. Invoking the item "Default" will display the built-in brush. Hitting the STOP key will deactivate any bouncing brush.

What is Required:
a) BrushDMT.bcd
b) One or more files in the system catalog named "*.brush". Note any file that worked with the Tajo 11.0 version of BrushDMT will work here. A deposit of many brushes can be found on [Goofy:OSBU North:Xerox]Hacks/Data/Brushes/.

Instructions:
To copy brush file over to BWS from CoPilot ANY of the methods below are acceptable.
a) Command Cent...

---

5. BWSActivity.bcd W. Dale Knutsen:OSBU North:Xerox 23-May-86 13:26:56
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSActivity.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSActivity.doc of 6-Jan-87 15:26:04
  Description:
    BWSActivity.doc 6-Jan-87 15:23:48 by DKnutsen

BWSActivity displays bar graphs that monitor the machine's activity. The tool displays any of the following four performance variables: cpu usage, disk IO, page faults, and VM backing file creations. Using a small window, the tool displays up to four bar graphs in horizontal histogram fashion. The graphs are updated once per second and give the activity for that second in black, and an average activity over the past ten seconds in grey. BWSActivity is activated at Logon and will deactivate at Logoff. The window defaults to appear in the lower right hand corner of the screen.

Each bar graph has the following format:

  <variable name> <instantaneous value> [BAR GRAPH] <limit value>

The variable name is the name of the variable being monitored (i.e. Cpu, DiskIO, Faults, VMFiles). The instantaneous value is the value of the variable for the most rec...

---

6. BWSAddressTranslation.bcd Bryan Yamamoto:OSBU North:Xerox 2-Apr-85 17:56:51
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSAddressTranslation.bcd
  Documentation: none

---

7. BWSAddressTranslationImpl.bcd Bryan Yamamoto:OSBU North:Xerox 30-Jul-85 9:44:43
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSAddressTranslationImpl.bcd
  Documentation: none

---

8. BWSBrushDisplay.bcd Perry A. Caro:OSBU North:Xerox 23-Oct-85 14:31:53
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSBrushDisplay.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSBrushDisplay.doc of 23-Oct-85 14:42:31
  Description:
    -- BWSBrushDisplay.doc
-- Caro 23-Oct-85 14:34:14

BWSBrushDisplay.bcd is an interim tool for converting XDE Doodle brushes to VP Freehand Drawing (RES) format. BWSBrushDisplay simply displays the brush in a window so that you can use the VP Freehand Drawing command "Copy Screen" to transfer the bits to the RES canvas.

Using this tool and VP Freehand Drawing, you can now insert XDE Doodle brushes into documents!

Get

[Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools
    BWSBrushDisplay.bcd

on you VP desktop.

Running BWSBrushDisplay:

1. Either drop BWSBrushDisplay on the Loader, or select it in the Loader and bug Run.

Fetching Brushes:

Get as many XDE Doodle brushes on your VP desktop as you like. Using a filedrawer is the handiest way.

Brushes can be found on [Goofy:OSBU North:Xerox]<Hacks>Data>Brushes

Converting Brushes:

Once BWSBrushDisplay is running, do the follow...

---

9. BWSCompiler.bcd Lori S. Nagata:OSBU North:Xerox 3-Jul 86 17:26:32
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSCompiler.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSCompiler doc of 4-Sep-86 16:30:56
  Description:
    -- File: BWSCompiler.doc
-- last edit by Nagata 4-Sep-86 16:03:48

The BWSCompiler is the VP version of the Mesa compiler. The compiler will compile both simple and Viewpoint documents. Procedure and variable names can contain non-ASCII characters. The 4.0 version is compatible with VP 1.0 and the 4.2 version is compatible with VP 1.1.

How to run the compiler --
  - Use the loader or CommandCentral to run BWSCompiler.bcd.
  - A compiler icon will appear in the Basic Icons folder.
  - Copy the icon and put it on your desktop.
  - If compiling VP documents, then the VP Document Editor needs to be running
  - Select the source document and drop it on the compiler icon.
  - A .bcd icon will be produced if the compilation is successful which can be dropped on the loader. If there are any errors, an error

log will be produced.
- To change the switches, select the compiler icon and hit <PROPS>.

---

10. BWSFgrep.bcd William H. McCoy:OSBU North:Xerox 18-Aug-85 15:28:47
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSFgrep.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSFgrep.doc of 18-Aug-85 16:43:59
Description:
--BWSFgrep.Doc
--WHMcCoy, 18-Aug-85
-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.
Fgrep is run from the System Attention Menu in the Basic Workstation. It creates a Tajo-style tool window which allows the user to specify the file name of a SimpleText file, a number of strings, and a file from which strings are to be read. The input file will be searched for instances of the specified strings; for each match found, the tool will output the string and the byte position in the file of the first character of the match. The speed of the search is fast, and independent of the number of strings searched for.

RESTRICTIONS: BWS Fgrep can search only simple text files which exist in the System catalog. Strings containing spaces must be quoted. If a pattern file is named, it must also be simple text and in the System catalog, and the patterns are considered to be delimited by carriage returns. Fgrep maps characters in the file and the string into a subset of CHARs: up...

---

11. BWSHeapCheckTool.bcd Lee F. Breisacher:OSBU South:Xerox 20-Aug-85 11:33:59
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSHeapCheckTool.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSHeapCheckTool.doc of 20-Aug-85 13:51:12
Description:
-- File: BWSHeapCheckTool.doc - last edit:
-- Breisacher.ES 20-Aug-85 13:51:12

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

BWSHeapCheckTool is just like HeapCheckTool, but runs in BWS. This makes it extremely quick and easy to find heap leaks. Please read HeapCheckTool.doc first! (It's on the Mesa hacks directory.)

To use

To use BWSHeapCheckTool, boot BWSPerfDLion.boot with the '6 switch (it won't work with the BWSDLion.boot!) and run BWSHeapCheckTool.bcd in BWS. Select "HeapCheckTool" in the attention window menu. The commands are just like the ones in HeapCheckTool.

New Feature

When you turn on "UseSavedState", the "Pages" and "Nodes" columns display with a third number - the difference between the saved value and the current value. For example:

Heap#	Owner		HC Handle	Pages	Used	Nodes
20	ContextImpl	G: 433430B	3202654B	8	18/63/45	2/7/5

45=63-18 and 5=7-2, so you can...

---

12. BWSIconEditor.bcd Gerard S. Zonus:OSBU South:Xerox 6-Aug-85 15:32:58
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSIconEditor.bcd
Documentation: none

---

13. BWSLibrarianTool.bcd Brian T. Lewis:OSBU North:Xerox 14-Feb-86 13:51:11
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSLibrarianTool.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSLibrarianTool.doc of 14-Feb-86 13:52:42
Description:
BWSLibrarianTool.doc - last edit by Brian Lewis on 14-Feb-86 13:52:40

BWSLibrarianTool.bcd is a BWS version of the XDE LibrarianTool released as part of Mesa 12.0. With few excpetions, it operates as described in the "Librarian Tool" chapter released with the Mesa 12.0 documentation. The differences are the following:

- o There are no executive commands; all interaction is through a BWS StarWindowShell.
- o Only one libject "name" can be specified. This name may contain wildcards, however.
- o There is no batch mode for libject creation.
- o The "log subwindow" is not file backed: lines that scroll off the top are lost.
- o If no domain or organization is specified for the Librarian database's name, defaults are taken from a [Librarian] section in the UserProfile:

```
[Librarian]
Domain: OSBU North
Organization: Xerox
```

---

14. BWSMahJong.bcd Bruce S. Lee:OSBU South:Xerox 24-Jul-85 14:27:04
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSMahJong.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSMahJong.doc of 27-Nov-85 13:59:48
  Description:
    --BWSMahJong.doc

This tool adds a command to the system menu called "Mah Jong Tool". When this command is invoked, the mah jong tool window opens.

OBJECT:

Arrange your tiles such that they conform to the following pattern:

four sets and a pair

where a "set" may be 1) a run such as 234 in the same suit, 2) three of a kind or 3) four of a kind

STRUCTURE OF THE DECK:
1) four copies of each tile
2) three suits numbered from 1 to 9
3) one "suit" with no ordering...only three-of-a-kind and four-of-a-kind sets may be produced from tiles in this class. This suit contains four winds (north, south, east, west) plus three chinese characters: hong-jong (looks like a sword), bat-ban (a white square) and fat-choy (looks like a birds nest).

RULES OF PLAY:
This version of MJ has most all of the usual rules encoded. All you have to do is respond to the various prompts listed below.

MESSAGES AND THEIR MEANING:

[That card doesn't go with the discard. Please try again.]
        The tile you just pointed...

---

15. BWSPowerMouse.bcd Jeremy M. Goodell:OSBU South:Xerox 9-Jul-85 11:33:35
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSPowerMouse.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSPowerMouse.doc of 9-Jul-85 12:00:48
  Description:
    -- File: BWSPowerMouse.doc - last edit:
-- Goodell.ES 9-Jul-85 12:00:29

        PowerMouse is a tool which modifies the behavior of the mouse icon, or cursor, with respect to the motion of the mouse itself. There are two parameters which describe this modification: 'Threshold' and 'Amplification'. Threshold is a speed (in pixels per screen refresh), which is the minimum speed of mouse movement for which cursor movement will be modified. Amplification is a percentage, describing the maximum amount of movement modification. Amplification = 100 is normal mouse action. Amplification = 200 is a maximum of doubling cursor response.

        The tool communicates to the user through a form window. Amplification and Threshold are readonly number items, which may be modified by using the associated "+" and "-" command items.

        The tool can be run as a .autorun or from the Command Central run line or through the loader. Running the tool adds a dependency to the logon event. When the tool i...

---

16. BWSSol.bcd Gerard S. Zonus:OSBU South:Xerox 25-Jul-86 16:34:41
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSSol.bcd
  Documentation: none

---

17. BWSStatusIcon.bcd Stephen W. Bartlett:OSBU South:Xerox 11-Jan-85 14:26:30
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSStatusIcon.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSStatusIcon.doc of 10-Sep-84 15:09:08
  Description:
    -- File: BWSStatusIcon.doc - last edit:
-- Bartlett.ES 10-Sep-84 15:09:08

BWSStatusIcon Documentation
Stephen W. Bartlett

A Status Icon displays how much disk space is left on a particular logical volume (much like the corner of the CoPilot Herald window). Through the use of the icon's property sheet, it is possible to change the volume, the information to display (amount used or free), the type of display (percentage or amount, digital or analog), and the update interval in seconds. Any number of these icons may be on the desktop at once; this provides for the display of information for different volumes at the same time.

GETTING STARTED

Copy the prototype Status Icon to the desktop and run BWSStatusIcon.bcd. The prototype icon displays the amount of space free on the system volume (the one that contains the boot file). Its update period is 15 seconds. The property sheet is fairly self-explanatory.

RESTRICTIONS

The analog display is currently not implemented; sel...

---

18. BWSTDFParser.bcd Richard Balcon:SBD-E:RX 13-Jan-86 6:19:16
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSTDFParser.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSTDFParser.doc of 27-Jan-86 6:46:36
  Description:
    -- File: BWSTDFParser.doc - last edit:

-- Balcon.rx 27-Jan-86 14:46:36

BWSTDFParser is a program to parse TDF (Terminal Descriptor Files) files and their Remote Batch Service equivalents, RBDF (Remote Batch Descriptor Files) files. It will report all errors within the file and try to recover from any found in order to continue parsing.

On finding an error it will report what token it expected, its position and the last token read at the point the error occurred.

Running the program will register four commands "Parse TDF", "Parse TDF (verbose)", "Parse RBDF" and "Parse RBDF (verbose)" in the attention menu.

TDF -
"Parse TDF" - parses the TDF and reports the errors.
"Parse TDF (verbose)" - parses the TDF and reports both the errors and structure of the file.

RBDF -
"Parse RBDF" - parses the RBDF and reports the errors.
"Parse RBDF (verbose)" - parses the RBDF and reports both the errors and struc...

---

19. BWSTest.bcd Sybil A. Johnson:OSBU South:Xerox 29-May-86 18:33:27
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>BWSTest.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BWSTest.doc of 29-May-86 18:58:05
Description:
-- BWSTest.doc last revised:
-- SAJohnson.es 29-May-86 18:58:04

FOR Users
=========

Bringing up the Tool

To use the BWSTestTool, load BWSTest.bcd. Load the .bcd's of any modules that contain tests you wish to run. Copy the sample icon from the Prototypes directory to your desktop. Opening the icon will cause the BWS Test Tool window to appear on the screen.

Listing Registered Tests

All of the currently registered tests can be listed by selecting the command item "List". The results will appear in the tool's message window.

Changing Testing Options

Several testing options are available: call debugger, message level and auto run. If Call Debugger is on, any time an unexpected error occurs the test goes to the debugger. (This is a proceedable SIGNAL.) Message level indicates the level of informational messages to be put in the message window. If auto run is on the test will be run as soon as it is ...

---

20. CartoonTool.bcd Bryan Yamamoto:OSBU North:Xerox 20-Jan-86 18:01:02
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>CartoonTool.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>CartoonTool.doc of 20-Jan-86 18:17:07
Description:
-- CartoonTool.doc
-- BGY 9-Jan-86 12:56:59

CartoonTool is a program that allows you to see the Cartoon-of-the-day stored on a cartoon server. To use this tool, you need not know the location of a cartoon server.

After loading the CartoonTool, select the CartoonTool item from the Attention window. This will create a window. Hit the Fetch button on the window to get the cartoon.

If there is a CartoonServer within five hops of your net, the CartoonTool will display the cartoon of the day. If you are more than five hops from a CartoonServer, or the CartoonServer you reach takes a long time to transfer the cartoon, you might consider installing a local cartoon server. See [Goofy: OSBU North:Xerox]<Hacks>12.0>Doc>CartoonServer.doc for more information.

---

21. CheckCursor.bcd Bruce K. Whittaker:OSBU North:Xerox 18-Jul-85 13:56:06
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>CheckCursor.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>CheckCursor.doc of 18-Jul-85 14:28:50
Description:
-- File: CheckCursor.doc - Copyright (c) Xerox Corporation 1985. Last edit:
-- Whittaker.PA 18-Jul-85 14:28:50

CheckCursor is a handy way to see what the different system provided cursors look like. CheckCursor.bcd registers CheckCursor in the Attention window. Selecting this command brings up a form window containing a New Cursor command and a Current Cursor field. Bugging New Cursor displays both the bitmap of the cursor, and its corresponding Cursor.Type name. Continued bugging will cycle around the entire Cursor.Type enumeration.

---

22. CHIcon.bcd Nanette E. Harter:OSBU North:Xerox 14-Mar-86 11:19:55
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>CHIcon.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>CHIcon.doc of 21-Feb-86 11:24:06
  Description:
    -- File: CHIcon.doc - Last edit:
-- Harter.PA 21-Feb-86 11:24:06

This is a prototype the Group and User Icons in the Viewpoint environment. The Icons provide access to Clearinghouse information.

The first application, Make Clearinghouse Icon, allows the user to create icons on the desktop associated with a clearinghouse entry through a menu item in the attention window menu. The other two applications, User Icon and Group Icon, provide access to the clearinghouse information associated with the clearinghouse entry represented by the icon.

Make Clearinghouse Icon

The Make Clearinghouse Icon tool creates user and group icons and places them on the desktop. The icons have a direct correspondence to an entry in the clearinghouse.

A new menu item, Make Clearinghouse Icon, is entered in the attention window menu when the application is run. Selecting the Make Clearinghouse Icon entry will bring up a window...

---

23. CreateCanvas.bcd Donald W. Gillies:OSBU North:Xerox 18-Jul-86 20:18:53
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>CreateCanvas.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>CreateCanvas.doc of 15-Jan-86 14:54:05
  Description:
    -- File: CreateCanvas.doc - last edit:
-- Strom.ES 15-Jan-86 14:54:05

CreateCanvas is a Viewpoint hack that reads the current selection and creates a new canvas (RES file) from it.

To use this hack, select an icon containing the bitmap you want to make into a canvas (RES file), and then select "Create Canvas" in the system menu. You should see the hour glass cursor, and a bit later a new canvas will pop up on your desktop.

CreateCanvas knows how to read the following bitmap formats:
  AIS files (file type = 8192)
  brush files (file type = 8888)
  CU files (file type = 110000)
  RES files (file type = 4428) (yes RES to RES - there was a bug in VP 1.0 RES files, and this will rebuild them correctly)

If anyone has other bitmap formats they would like to see included in this hack, drop me a line.

Enjoy,

Darrel

---

24. CreateChar.bcd Deborah J. Lewis:OSBU South:Xerox 9-Oct-86 19:33:03
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>CreateChar.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>CreateChar.doc of 15-Oct-86 17:32:52
  Description:
    -- File: CreateChar.doc - last edit:
-- Lewis.ES 15-Oct-86 17:32:52

CreateChar allows the creation of arbitrary XChar.Character's. It is useful for creating wierd characters during multilingual development.

IMPORTANT CAUTION: Since characters can be created which are not defined in the Xerox Character Code Standard (XCCS) or which are not intended to be stored in Viewpoint objects (e.g., runtime character set 376B), users must exercise discretion in how they utilize wierd characters.

In keeping with the tradition of the XCCS, CreateChar speaks only octal. All values are specified in octal without a trailing B, e.g. "171" means 171B.

Bugging the "Create Char" command in the Attention window menu brings up a tool window. The Append! comand appends one or more characters to the (unnamed) string item in the tool window. The string can be edited as usual and can be copied into other windows through th...

---

25. Cruncher.bcd Perry A. Caro:OSBU North:Xerox 20-Nov-85 13:51:57
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>Cruncher.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>Cruncher.doc of 20-Nov-85 13:59:59
  Description:
    -- Cruncher.doc - last edited by:
-- Caro: 20-Nov-85 13:42:19

Cruncher is conversion software that plugs into the ViewPoint Converter icon. It takes any NSFile and runs it through the Woods Crunch

Kernel (an enhanced Run/Huffman encoder/compactor by Don Woods). Attributes and multiple segments are preserved by first serializing (via NSFile.Serialize) the file, and then crunching the serialized file. The encoding reduces that total size of the file without loosing ANY information. Cruncher also provides the inverse operation (uncrunch) to restore the original data.

FEATURES

* Useful for archived files: mail, documentation, back ups, etc.

* Archiving/Transporting desktops

* There are three target options: Crunched, Crunched In Background, Crunched XDE ... and their corresponding UnCrunched targets.

* Very efficient. 50% compression is common for VP Documents.

* Compatible with the XDE "Crunch", as an option

F...

---

26. CuFilmTool.bcd Deborah J. Lewis:OSBU South:Xerox 26-Jul-85 12:13:18
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>CuFilmTool.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>CuFilmTool.doc of 1-Aug-85 16:50:11
Description:
-- CuFilmTool.doc

-- Last edited by: On:
-- Lewis.ES 1-Aug-85 16:49:59

OVERVIEW

CuFilmTool allows CU files to be displayed and manipulated on a BWS desktop. Its primary use is to view and scale CU pictures produced by the ScreenCamera tool before moving them to an OS5.0 desktop, where they can be merged into Star documents in an image frame. When CuFilmTool is running, the icon for CU files is represented as a roll of film and the OPEN and PROPS keys are supported.

OPERATION

Retrieve CuFilmTool.bcd from the hacks directory and copy it to the Loader icon on the BWS desktop. CU files will then respond to OPEN and PROPS.

OPENing a CU film icon creates a read-only window which displays the bitmap stored in the CU file. The CU file is always displayed at a resolution of one sample bit per screen pixel and is not affected by the scale property of the file (see below).

Selecting a CU film icon...

---

27. DefaultIconProps.bcd James G. Sandman:OSBU North:Xerox 12-Aug-85 13:43:00
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>DefaultIconProps.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>DefaultIconProps.doc of 12-Aug-85 13:45:53
Description:
-- File: DefaultIconProps.doc - last edit:

DefaultIconProps registers a system menu command that will invoke the default icon property sheet for the selected file. This allows developers to change the names of any files. One use for this tool is to rename old running application folders so new ones may be retrieved.

---

28. DefaultIconPSheet.bcd J. Paul Holbrook:OSBU South:Xerox 18-Dec-84 15:45:24
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>DefaultIconPSheet.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>DefaultIconPSheet.doc of 19-Dec-84 8:24:55
Description:
-- File: DefaultIconPSheet.doc - last edit:
-- Holbrook.ES 19-Dec-84 8:24:55
-- Breisacher.ES 20-Apr-84 15:17:15

DefaultIconPSheet will bring up a property sheet for icons that have no application loaded for them yet. The property sheet displays the name of the file, the NSFile file type, the create date of the file, its size in pages, and the remote pathname of the file.

The name and file type items can be changed.

The FileType item makes it very easy to change the name and/or filetype of an icon. For example, say you need an icon of filetype 100011 for the GoDemo. BEFORE loading the GoDemo, simply copy some other icon that has no application loaded, then select it, hit props, change the file type and bug Done. Then load the GoDemo.

The remote pathname information is the string stored on the StarAttributeTypes.remoteName extended attribute. If the file was copied or moved from a remote server by FileContainerSource, this item will show the fully qualified pathname...

---

29. DesktopIconPictureImpl.bcd Gerard S. Zonus:OSBU South:Xerox 16-Jan-85 15:27:07
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>DesktopIconPictureImpl.bcd

Documentation: none

---

30. DocumentStatistics.bcd Bruce S. Lee:OSBU South:Xerox 11-Dec-85 13:08:08
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>DocumentStatistics.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>DocumentStatistics.doc of 11-Dec-85 13:14:56
Description:
-- DocumentStatistics.doc - last edit:
-- BLee.ES 11-Dec-85 12:55:03

Places a menu command "Document Statistics" into the system mouse menu. To use, simply select one or more closed icons and invoke "Document Statistics". The tool will enumerate them and collect statistics. If you have an open document, simply place the selection into the open document and invoke "Document Statistics".

Statistics will be posted in the attention window. Future UI options might include output to a text item in a form window or output to a log file.

Sample Output:

Documents examined: 1
Characters: 324; Table Entries: 44; Graphics Objects: 5
Details:
Points: 2; Straight lines: 6; Triangles: 20; Rectangles: 12; Curve lines: 90; Ellipses: 1; Pie slices: 5; Text frames: 4; Graphic frames: 2; Cusp buttons: 1; Clusters: 2; Bar charts: 5; Line charts: 3; Pie charts: 1; Captions: 10; Bitmap frames: 2; Square inches in bitmap frames:...

---

31. DoneLoading.bcd J. Paul Holbrook:OSBU South:Xerox 25-Jan-85 13:47:57
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>DoneLoading.bcd
Documentation: none

---

32. Drinks.bcd Mark K. Hahn:OSBU North:Xerox 1-May-85 13:26:22
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>Drinks.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>Drinks.doc of 7-May-85 16:34:26
Description:
-- Drinks.doc
-- hahn.pa 7-May-85 16:11:13

Drinks is a simple database for storing various bar drinks using NSFiles.
The drinks database is primarily a programming exercise for students taking the ViewPoint Unit.

Drinks runs from the desktop with an icon called "Drinks Tool". The tool creates storage files when the first drink is added to the database. The data files reside in a catalog named catalogName (type 110111) in files of type 110011. directions for using the database follow.

1. To add drinks to the database you must enter a "drink name"; specify the drinks contents via booleans; and enter directions for mixing the drink in the directions field. The drink is then added by selecting the menu command "Make Drink". If the drink name does not already exist a new drink is added to a popup menu called "drinks on file". If the drink exists the contents and directions fields are rewritten.

2. There are 2 methods of retrieving a drink:
1. You can retrieve a drink by selecting th...

---

33. FFF.bcd Lee F. Breisacher:OSBU South:Xerox 6-Jan-86 13:29:02
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FFF.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FFF.doc of 3-Jan-86 12:01:36
Description:
-- File: FFF.doc - last edit:
-- Breisacher.es 3-Jan-86 12:01:36

FFF stands for Folder/FileDrawer Filter. It allows you to open folders and file drawers with a name filter. For example, you could type "*.symbols" into FFF and open [Stanford:OSBU South:Xerox]<ABasicWS>4.0>Private> and you'd get only the .symbols files rather than all the files in that subdirectory. Real handy for large directories. Sortof gives you some FileTool-like functionality.

Anyway, here's how to use it. After running FFF.bcd, there'll be a "Folder/FileDrawer Filter" item in the attention window menu. Selecting this item will bring up a little option sheet with a text item and boolean item. You can enter any file name with (or without) asterisk wild cards. If you want only the highest version of the files, turn on the HighestVersion boolean. After filling in the name, select the folder or file drawer you want to open, then bug App...

---

34. FileProps.bcd Brian T. Lewis:OSBU North:Xerox 26 Sep 85 13:39:05
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FileProps.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FileProps.doc of 17-Sep-85 12:09:54
Description:
-- FileProps.doc Documentation for FileProps
-- Wayne K. Yamamoto 17-Sep-85 12:09:47

I . Abstract on Application

FileProps is a tool that runs in the BWS world that allows the user to examine and change various attributes of a file. Specifically, it allows the user to modify a file's Access List, Default Access List, name, and type. It allows the user to examine "low-level details (file length, create date, where it is backed up, etc.)" about the file. It is similar in functionality to the XDE tool "FSWindowTool."

II. Where to Get It

FileProps is located on
[Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FileProps.bcd.

This document is located on
[Alt:OSBU North:Xerox]<BWSHacks>4.0>Docs>FileProps.doc.

III. USAGE

FileProps runs out of the Attention Window. By selecting "File Properties" from the Attention Menu, the file properties of the current selection -- which may be EITHE...

---

35. FileTimeout.bcd Russell Sonnenschein:OSBU South:Xerox 6-Sep-85 15:52:10
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FileTimeout.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FileTimeout.doc of 9-Sep-85 14:56:35
Description:
-- File: FileTimeout.Doc - last edit:
-- Sonnenschein.ES 6-Sep-85 18:22:01

FileTimeout resets the NSFile default timeout through a user profile entry. With FileTimeout the user can set the default timeout from 1 to 60 seconds.

To use FileTimeout you must add the following to your User Profile:

[FileTimeout]
Seconds: # -- some number in the range (0..60]

The Default Timeout specified by the user remains in effect between Logon and Logoff. After Logoff, the default timeout changes back to NSFile's default (60 seconds).

The current NSFile implemention uses 60 seconds as its default timeout. Almost all BWS applications use this default. For many/most applications 60 seconds can be too long, expecially if an application only deals with local files.

Using a shorter default time can give the user better response time when an application is having problems accessing a file. Also, it may be interesting for a deve...

---

36. FilmConverter.bcd Deborah J. Lewis:OSBU South:Xerox 26-Jul-85 15:06:09
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FilmConverter.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FilmConverter.doc of 1-Aug-85 16:44:27
Description:
-- FilmConverter.doc

-- Last edited by: On:
-- Lewis.ES 1-Aug-85 16:44:18

OVERVIEW

FilmConverter runs on a BWS desktop and allows picture files created by ScreenCamera to be converted to different film formats. Currently, only CU -> AIS conversion is supported.

OPERATION

Retrieve FilmConverter.bcd from the hacks directory and copy it to the Loader icon on the BWS desktop. A command is registered in the Attention window menu which activates the FilmConverter option sheet.

When "FILM CONVERTER" is selected in the Attention window menu, an option sheet for the converter tool is opened. The option sheet contains two sections, one describing the source file that is to be converted and the other describing the destination file which is to be created.

The following parameters appear in the Source file section:

o Location - choice item - the file may be obtained from either the desktop or the Sy...

---

37. FlushOut.bcd J. Paul Holbrook:OSBU South:Xerox 7-Mar-85 13:30:26
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FlushOut.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FlushOut.doc of 7-Mar-85 13:58:26
Description:

```
    -- File: FlushOut.doc - last edit:
-- Holbrook.ES          7-Mar-85 13:58:26
```

FlushOut puts the command "Broom Memory" in the global aux menu. Invoking this command will free up as many real pages as possible by flushing everything out.

FlushOut may be useful for performance tests to determine working sets. (Broom memory, run the test, break to CoPilot and use MemoryMap to check on the pages that have been swapped in.)

Note that flushout isn't perfect; in order to redisplay the screen after invoking the command, some pages will be swapped for the window package will be swapped back in. A certain amount of Pilot remains in as well. However, doing a flushout should reduce the number of pages in real memory down to about 200.

---

38. FlySwatter.bcd Mark K. Hahn:OSBU North:Xerox 2-May-85 12:19:38
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FlySwatter.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FlySwatter.doc of 7-May-85 16:06:56
  Description:
    -- FlySwatter.doc
-- hahn.pa 9-Apr-85 23:04:11

FlySwatter is a game played in the BWS.

The game runs from the attention menu and has no icon. The user plays the game by selecting startgame from the menu. Playing the game consists of clicking point over fly-bitmaps, that appear in random locations on the window, before they disappear. If the user is fast enough and hits (Points over) a fly before it disappears a point is added to his score. The game is over after 50 flys have appeared on the display or when the user selects endgame.

Features:
The user may adjust the length of time a fly remains on the screen via a popup menu. Three levels are provided Beginner, intermediate or expert (.75 sec. - .4 sec.). A high pitched tone will sound when a fly is struck and a low tone when missed.

The user may also increase (decrease) the difficulty of the game by changing the size of the StarWindowShell. The random fly's location is dependent on the size of the Window and will dynamically ad...

---

39. FontedTextEditor2.bcd Alfredo E. Camacho:OSBU South:Xerox 5-Jun-85 11:06:32
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FontedTextEditor2.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FontedTextEditor2.doc of 27-Jun-85 14:56:24
  Description:
    -- File: fontedtexteditor2.doc - last edit:
-- Camacho.ES 27-Jun-85 14:56:24

WARNING! This hack replaces the implementation for file type = 2 (text file type)

Run line for FontedTextEditor2 is:

AttrISConverterConfig StarKeyboards FontMgrConfig FontedTextFieldConfig FontedTextEditor2.

The FontedTextEditor2 hack works very similar to the SimpleEditor except that it contains functionality to deal with attributed text. A fonted text editor window will allow the user to change fonts by using the softkeys and font key on the keyboard. A property sheet also allows the user to change attributes of the text inside the window.

FontedTextEditor2 overrides the implementation of file type = 2 or simple text files. Once FontedTextEditor2 is loaded, the Open key calls upon FontedTextEditor2's implementation instead of the BWS's implementation for simple text files.

In addition to the editing capability of FontedText...

---

40. FormWindowLayoutTool.bcd Randy Gobbel:OSBU North:Xerox 30-Apr-85 16:54:30
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>FormWindowLayoutTool.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>FormWindowLayoutTool.doc of 29-Apr-85 15:27:06
  Description:
    -- File: FormWindowLayoutTool.doc - last edit:
-- Gobbel.pa 29-Apr-85 15:27:06
-- Diamond.PA 20-Aug-84 19:19:47
-- Breisacher.ES 19-Apr-84 14:52:04
  Caro.pa 1-Apr-85 14:07:15 (but no fooling here!)

[4.0g Version: Please see Restrictions List]

The FormWindowLayoutTool allows a BWS programmer to graphically layout a FormWindow or PropertySheet. The tool automatically generates much of the Mesa source needed to produce a FormWindow or PropertySheet. These sources can then be compiled and executed and the resulting FormWindow will look like the one laid out earlier.

Description of the FormWindowLayoutTool:

After loading the tool in the Basic Workstation, a "FormWindow Layout Tool" menu item will be added to the Attention window menu. Bugging this brings up the layout tool, which has the following items:

ItemType: {Choice, Decimal, Integer, Boolean, Text, Command, Tagonly, Window}
  ...

---

41. IconFileImpl.bcd Gerard S. Zonus:OSBU South:Xerox 16-Jan-85 15:25:12
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>IconFileImpl.bcd
  Documentation: none

---

42. IPressStream.bcd Bruce K. Whittaker:OSBU North:Xerox 7-Sep-85 18:30:39
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>IPressStream.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>IPressStream.doc of 24-Sep-85 18:33:11
  Description:
    -- File: IPressStream.doc - Copyright (c) Xerox Corporation 1985. Last edit:
-- Whittaker.PA 24-Sep-85 18:33:11

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

  IPressStream is a BWS translation of the Tajo PressStream interface. It is based upon Interpress, hence the name.

  See documentation for PressStream for details.

---

43. KeepWindows.bcd Harold J. Shinsato:OSBU North:Xerox 16-May-85 12:17:26
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>KeepWindows.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>KeepWindows.doc of 30-Apr-85 21:53:17
  Description:
    -- File: KeepWindows.doc
-- Last Edit: Shinsato 30-Apr-85 21:53:12
-- Owner: Shinsato:OSBU North:Xerox

-- Copyright (C) Xerox Corporation 1985. All rights reserved.

INTRODUCTION

KeepWindows is a simple hack which saves the place and size of all Star window shells associated with the icons on the desktop. The information is saved during logoff, and is restored during logon.

HOW TO USE KEEPWINDOWS

In order to run KeepWindows, all you need to do is to retrieve KeepWindows.bcd or KeepWindows.autorun from the [Alt:OSBU North]<BWSHacks>4.0>Tools> directory, and drop the file on the Loader Icon of your desktop. The autorun version will be started automatically when the Basic Workstation is booted.

IMPLEMENTATION DETAILS

Before every logoff, KeepWindows stores the size and place of each icon's StarWindowShell in a file, "UserDesktop". This file is placed in the user's Directory Icon. During logon, the file is retrieved from the Directory Icon, and the windows are restored. Since I...

---

44. Life.bcd Gerard S. Zonus:OSBU South:Xerox 26-Feb-85 17:15:37
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>Life.bcd
  Documentation: none

---

45. MahJong.bcd Bruce S. Lee:OSBU South:Xerox 1-Apr-85 17:07:03
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>MahJong.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>MahJong.doc of 1-Apr-85 16:29:48
  Description:
    --MahJong.doc

This tool adds a command to the system menu called "Mah Jong Tool". When this command is invoked, the mah jong tool window opens.

OBJECT:

Arrange your tiles such that they conform to the following pattern:

four sets and a pair

where a "set" may be 1) a run such as 234 in the same suit, 2) three of a kind or 3) four of a kind

STRUCTURE OF THE DECK:
1) four copies of each tile
2) three suits numbered from 1 to 9
3) one "suit" with no ordering...only three-of-a-kind and four-of-a-kind sets may be produced from tiles in this class. This suit contains four winds (north, south, east, west) plus three chinese characters: hong-jong (looks like a sword), bat-ban (a white square) and fat-choy (looks like a birds nest).

RULES OF PLAY:
This version of MJ has most all of the usual rules encoded. All you have to do is respond to the various prompts listed below.

MESSAGES AND THEIR MEANING:

[That card doesn't go with the discard. Please try again.]
  The tile you just pointed at...

---

46. MakeIPFileType.bcd Frank H. Bowers:OSBU North:Xerox 25-Feb-86 17:20:31
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>MakeIPFileType.bcd

Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>MakeIPFileType.doc of 26-Feb-86 12:39:58
Description:
-- File: MakeIPFileType.doc - last edit:
-- Bowers.PA 26-Feb-86 12:39:58

MakeIPFileType is a Viewpoint hack that looks at the current selection and, if it is a file in IP format, changes the file type to that of a Viewpoint printer format file (IP file type = 4361).

To use this hack, select a foreign interpress master icon (interpress master generated in XDE described as "unknown object in Viewpoint) and then invoke "Change to IP File Type" in the system menu. The file type will be changed and the icon should then be repainted as an interpress master icon (IP file).

Warning, changing file types can be hazardous to your health.

Code Stolen from Darrel Strom's MakeRESFileType.

---

47. MakeRESFileType.bcd Darrel E. Strom:OSBU South:Xerox 20-Jan-86 10:55:09
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>MakeRESFileType.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>MakeRESFileType.doc of 15-Jan-86 17:15:42
Description:
-- File: MakeRESFileType.doc - last edit:
-- Strom.ES 15-Jan-86 17:15:42

MakeRESFileType is a Viewpoint hack that looks at the current selection and, if it is a file in RES format, changes the file type to that of a Canvas (RES file type = 4428).

To use this hack, select an icon containing a bitmap in RES format (but with the wrong file type) and then select "Change to RES File Type" in the system menu. The file type will be changed and the icon should then be repainted as a canvas icon (RES file).

Warning, changing file types can be hazardous to your health.

Darrel

---

48. Mazewar.bcd Bryan Yamamoto:OSBU North:Xerox 2-Apr-85 19:58:59
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>Mazewar.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>Mazewar.doc of 2-Oct-85 14:51:23
Description:
-- MazeWar.doc
-- Yamamoto 3-Nov-84 15:59:24

How To Start

After running MazeWar.bcd in the Basic workstation, a file ("MazeWar") will appear in the System Folder. Copy MazeWar to your desktop and open it. To play a game, type is a name into the "Your Name:" field and hit "Start Game". You will then either join another existing game or if there is no one else playing, you will start a game. The person to first start a game like this is known as the duke.

If more than one games is going on and you know the duke of a particular game, and if you know the duke's network address, type that address into the "Duke Name:" field before starting the game. The Duke Name: field uses AddressTranslation to parse its input. (see below for joining games on other networks)

To quit, hit the Close button on the Star window shell.

Game Description

There are three subwindows in the MazeWar game, one ...

---

49. MJ.bcd Stephen B. Tom:OSBU North:Xerox 4-Mar-85 19:27:07
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>MJ.bcd
Documentation: none

---

50. OpenAsFolder.bcd James G. Sandman:OSBU North:Xerox 13-Nov-85 14:56:41
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>OpenAsFolder.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>OpenAsFolder.doc of 30-Jul-84 9:22:05
Description:
-- File: OpenAsFolder.doc - last edit:
-- Breisacher.ES 30-Jul-84 9:22:05

OpenAsFolder will open any file (icon) that has the isDirectory attribute. Simply select the icon, then select the "OpenAsFolder" menu item in the Attention window. The children of the file will be displayed just like a folder and they may be opened, deleted, moved, copied, etc. just like a folder.

---

51. PacMan.bcd Stephen B. Tom:OSBU North:Xerox 28-Aug-85 15:21:25
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>PacMan.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>PacMan.doc of 28-Aug-85 15:21:21
  Description:
    Document name: PacMan.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA 28-Aug-85 15:19:51
Owner: Tom:PA

PacMan registers the menu command, "PacMan" in the Attention window. Invoking the menu command will bring up the window. Hit [Start] to begin the game. Moving the mouse cursor in and out of the window will pause the current session to give the player a break. Demo mode is provided.

To control directions of the PacMan use the following keys:

8010 Dandelion hardware:
  HELP => up
  UNDO => right
  MARGINS => down
  NEXT => left

6085 Daybreak hardware:
  KeypadEight => up
  KeypadSix => right
  KeypadTwo => down
  KeypadFour => left

What is Required: PacMan.bcd

Note: If the "O" boot switch is used the following data file is needed:
Data file: [Alt:OSBU North:Xerox]BWSHacks/4.0/Data/PacMan.TIPC

---

52. PasswordTool.bcd Robert Sperry:WBST128:Xerox 10-Mar-86 7:43:53
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>PasswordTool.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>PasswordTool.doc of 10-Mar-86 7:29:49
  Description:
    -- File: PasswordTool.doc By R. Sperry 5-Mar-86 10:39:54

Previously, the only way in which a VP user could change their password was by opening a remote executive connection to a Clearinghouse. This method of changing a password is cumbersome, and results in a considerable security exposure. This tool solves both problems. To use it:

1) Copy the file PasswordTool.autorun to the loader icon; then end your current session. When you logon the item "Change Password" will be registered in the attention menu (i.e. the popup menu which appears when the cursor is moved to the top right of the screen and a mouse button is depressed.

2) Select Change Password from the attention menu. This will cause a form to be displayed on the screen.

3) Select the password type from {simple, strong, both}. If you are uncertain about which one to choose leave it at the default value of {both}.

4) Fillin the "New Password" field with the new password value. This value is displayed in clear text so that you...

---

53. PermanentZONE.bcd Steven W. Epp:OSBU North:Xerox 5-Nov-85 10:40:48
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>PermanentZONE.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>PermanentZONE.doc of 7-Nov-85 15:35:43
  Description:
    PermanentZONE.doc 31-Oct-85 17:12:53 by Epp

PermanentZONE is a storage allocation package for use with permanent or long-lived data. PermanentZONE is quite similar to Heap, but is very fast and has less storage overhead. PermanentZONE depends only on Pilot, and thus is usable in BWS, XDE, and a server environment.

The client uses PermanentZONE to create a Mesa UNCOUNTED ZONE, from which storage is then allocated. Storage allocated from such ZONEs remains allocated until the entire ZONE is destroyed. The client may do FREEs, but FREE actually does nothing.

Typical uses of PermanentZONE:

- For allocation of permanent or very long lived storage
- For allocation of storage where performance is much more important than short-term freeing of storage.
- For allocation of storage where the client's garbage collection strategy is to free storage en masse when some object is destroyed.

For detailed programming information, see the comments in Permanent...

---

54. PermanentZONEImpl.bcd Steven W. Epp:OSBU North:Xerox 1-May-86 14:16:51
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>PermanentZONEImpl.bcd
  Documentation: none

---

55. PupFetchTool.bcd James G. Sandman:OSBU North:Xerox 12-Aug-85 16:41:10
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>PupFetchTool.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>PupFetchTool.doc of 19-May-86 10:08:11
  Description:
    -- File: PupFetchTool.doc - last edit:
-- Sandman.pa   21-Mar-85 10:21:31
-- JThomson:El Segundo:Xerox 19-May-86 10:06:11

PupFetchTool.doc is an tool that allows file access from the BWS to Pup based file servers, typically IFSs and STPServers.

The tool registers a command with the system menu.

It has a formwindow with the following items:

  Host: Directory:
  Source:
  Dest name: Dest type:
  User name: Password:

It has three main commands:
  List, Fetch and Close Connection.

It's operation should be obvious. Files are fetched to the system folder.

This is a quick and dirty hack so there are lots of rough edges.

The file types listed by the Mesa 12.0 FSWindowTool are as follows:

1(Directory, Folder)
4101(Binary)
4226(BravoText)
4352(Desktop)
4(Empty,MailNote)
4098(FileDrawer)
4355(Inbasket)
4361(Interpress)
4360(Outbasket)
4290(Print Service Fonts)
4365(Record File)
3(Serialized)
4353(Star ...

---

56. RegisteredPSheet.bcd Jeffrey A. Johnson:OSBU North:Xerox 25-Jul-86 14:22:19
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>RegisteredPSheet.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>RegisteredPSheet.doc of 17-Jul-86 15:25:07
  Description:
    --RegisteredPSheet.doc

-- Last edited by: JeffJohnson   17-Jul-86 11:17:29

This doc file is just a placeholder. The REAL documentation for RegisteredPSheet is a ViewPoint document filed on [Gangplank:OSBU North:Xerox]PICS Fancy Programming Doc/RegisteredPSheet Interface. Its format is similar to that of chapters in the ViewPoint Programmers' Manual.

RegisteredPSheet is an interface that simplifies property sheet creation and makes it possible for more than one client to display the same property sheet. It was developed by Bruce Whittaker and me to support the Spinnaker Styles facility. Now that Spinnaker is gone I decided to release it as a hack in case anyone else can use it.

Property sheet implementations register themselves with the RegisteredPSheet facility. Clients that wish to display property sheets do so by identifying the sheet(s) to be displayed and passing the initial property values via a LI...

---

57. RegisteredPSheetImpl.bcd Jeffrey A. Johnson:OSBU North:Xerox 29-Jul-86 9:39:56
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>RegisteredPSheetImpl.bcd
  Documentation: none

---

58. Ripples.bcd   Stephen B. Tom:OSBU North:Xerox   25-Feb-85 20:43:45
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>Ripples.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>Ripples.doc of 25-Feb-85 20:03:04
  Description:
    Document name: Ripples.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA   25-Feb-85 20:03:03
Owner: Tom:PA

Ripples registers the menu command, "Ripples" in the Attention window. It is activated by invoking the menu item and deactivated by depressing the STOP key.

What is Required: Ripples.bcd

---

59. RootPicture.bcd Stephen B. Tom:OSBU North:Xerox 14-Mar-86 13:44:32
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>RootPicture.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>RootPicture.doc of 14-Jan-86 17:07:10
  Description:
    Document name: RootPicture.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA 14-Jan-86 17:07:10
Owner: Tom:PA

RootPicture registers the menu command, "RootPicture" in the Attention window. By toggling the menu item, a tiny window located in the upper righthand side of the screen will be activated/deactivated. Chording over this window will invoke a menu listing all "*.press" files in the system catalog. Selecting a particular choice will display its corresponding rootpicture. Invoking the item "Desktop" will display the customary desktop gray.

What is Required:
  a) RootPicture.bcd
  b) One or more files in the system catalog named "*.press". Note any file that worked with the Tajo 11.0 version of RootPicture will work here. A cache of many rootpictures can be found on [Goofy:OSBU North:Xerox]Hacks/Data/RootPictures/.

Instructions:
To copy press file over to BWS...

---

60. RootPuzzle.bcd Stephen B. Tom:OSBU North:Xerox 14-Mar-86 14:17:03
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>RootPuzzle.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>RootPuzzle.doc of 16-Jan-86 15:18:53
  Description:
    USER DOCUMENTATION FOR THE ROOTPUZZLE HACK

The RootPuzzle hack for BWS Viewpoint is a variation of Stephen Tom's RootPicture hack. The user can select a full-screen root picture from a menu of press-format files on the user volume. The selected picture can be left intact or can be scrambled into 16 rectangular pieces, 15 of which are displayed. The mouse cursor and point button can be used to unscramble the picture by moving selected pieces, one at a time, into a single empty space.

Loading and running RootPuzzle.bcd adds a "RootPuzzle" command to the Attention window menu. Invoking this command displays a very small "RootPuzzle" window in the upper right corner of the screen. The mouse point button is used to display and select from a pop-up menu of root pictures previously loaded into the BWS user volume. To display the usual gray background instead of a root picture, select "Desktop" from this menu. Any press-format root picture (such as Viking.press) can be used, although some m...

---

61. ScreenCamera.bcd Darrel E. Strom:OSBU South:Xerox 25-Sep-85 14:52:44
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>ScreenCamera.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>ScreenCamera.doc of 6-Aug-85 13:50:27
  Description:
    -- ScreenCamera.doc

-- Last edited by: On:
-- Lewis.ES 6-Aug-85 13:45:03

OVERVIEW

ScreenCamera is a tool for taking pictures of a Dandelion display, with the picture developed in one of several possible film formats. Currently, AIS, CU, and RES formats are supported. One application of ScreenCamera is to take screen pictures of a BWS desktop for insertion into Star/Viewpoint documents. In particular, this is an easy way to create a diagram of a property sheet for an FS/ESCN.

PLEASE NOTE: The official format for Viewpoint is RES. Both the bitmap editor and document bitmap frames will support RES as of the Beta-test release of the software. Early internal releases of OUT software used AIS as an interim format. However, RES will supplant AIS by the time the Viewpoint software is released for Beta test at customer sites.

AIS pictures may be used with a bitmap frame in a Viewpoint (Star 4.0) document....

---

62. ScrollBar.bcd Bruce K. Whittaker:OSBU North:Xerox 22-Sep-85 14:17:52
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>ScrollBar.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>ScrollBar.doc of 25-Sep-85 18:24:01
  Description:
    -- File: ScrollBar.doc - Copyright (c) Xerox Corporation 1985. Last edit:
-- Whittaker.PA 25-Sep-85 18:24:01

ScrollBar

0.0.1 Overview

The ScrollBar interface provides a mechanism for scrolling individual body windows of a StarWindowShell.
***** WARNING *****
1) Clients of this interface should not use StarWindowShell scrolling. The interaction between StarWindowShell scrolling and Scrollbar scrolling is undefined.
2) Clients of Scrollbar should make and get Contexts only on the the window explicitly handed to MakeScrollBar and returned by MakeScrollWindow. In particular, dependencies on StarWindowShell.GetBody will FAIL.

0.0.2 Interface Items

```
  MakeScrollBar: PROCEDURE [
    scrollee: Window.Handle,
    contentHeightProc: ContentHeightProc,
    stepSize: INTEGER ← 0] RETURNS [scrollbar: Window.Handle];

  ContentHeightProc: TYPE = PROC [window: Window.Handle] RETURNS [INTEGER];

        MakeScrollBar...
```

```
63. ShowChars.bcd       Eric R. Mader:OSBU South:Xerox      7-Aug-85  9:28:33
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>ShowChars.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>ShowChars.doc of 7-Aug-85 10:01:35
  Description:
    -- File: ShowChars.doc - last edit:
-- Mader.ES             7-Aug-85 10:01:35

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.
```

ShowChars is a hack which shows you information about characters. It registers two commands in the attention window menu, "Show Characters" and "Show Character Codes".

"Show Characters" converts the current selection to a string and dispalys the string in the attention window. This is usefull when you're looking at text displayed in some font other than the system font and you want to see what the characters look like in the system font. This is especially usefull if the characters are displayed as black boxes in the other font.

"Show Character Codes" converts the current selection to a string and dispalys the character codes for the characters in the string in the attention window. The output looks like this:

```
  [0B, 101B], [0B, 142B], [0B, 143B]
```

```
64. ShowFile.bcd        Franz-Josef Denig:Visitors PA:Xerox Visitors 6-Jun-85 10:44:53
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>ShowFile.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>ShowFile.doc of 5-Jun-85 15:53:39
  Description:
    ShowFile.doc       5-Jun-85 15:53:38 by Denig

Copyright (C) 1985 by Xerox Corporation. All rights reserved.
```

ShowFile is a tool which runs in the Basic Work Station. Like the FSWindowTool's file attributes window, ShowFile shows properties of files which are either on your local disc or on a File Service.

ShowFile registers the menu command "Show File Prop's" in the Attention window. By selecting a file (icon, container window item, etc.) and toggling the menu item, a window on the screen will appear with the properties of the selected file. You can also select children of an opened folder/directory.

In this version of the tool you can't change any of the properties.

```
65. ShowVM.bcd          Brian T. Lewis:OSBU North:Xerox     6-Mar-85 17:32:42
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>ShowVM.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>ShowVM.doc of 2-Oct-85 14:55:05
  Description:
    Document name: ShowVM.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/ShowVM.bcd
Last edited by Lewis on  7-Mar-85 10:14:37

-- Copyright (C) Xerox Corporation 1984, 1985. All rights reserved.
```

ShowVM creates a small window that displays virtual memory usage as a bitmap: mapped VM pages are represented by black pixels while unmapped pages are left "blank". (Actually, two vertical pixels represent each page for readability.) The window is refreshed every five seconds to give a rough indication of how VM is being used at that time. This display can be used to track the VM requirements for a collection of BWS applications. It can also be used to discover VM storage leaks.

ShowVM registers a menu command, "Show Virtual Memory", in the Attention window. Invoking this command will bring up the VM display window. The window also displays the total number of mapped pages, the percentage of VM that is mapped, and the size of the largest mapped interval.

```
66. SortMenu.bcd        Lee F. Breisacher:OSBU South:Xerox  2-Aug-85 16:08:16
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>SortMenu.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>SortMenu.doc of 12-Aug-85  9:45:50
  Description:
    -- File: SortMenu.doc - last edit;
-- Breisacher.ES       12-Aug-85  9:45:50

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.
```

SortMenuConfig.bcd EXPORTS SortMenu which will sort and keep a MenuData.MenuHandle sorted. Here is SortMenu.mesa:

```
SortMenu: DEFINITIONS = BEGIN

  Sort: PROCEDURE [menu: MenuData.MenuHandle];
  -- Whenever an item is added or swapped in 'menu',
```

```
   --  the menu is sorted.

   Unsort: PROCEDURE [menu: MenuData.MenuHandle];
   -- Sets the menu back to unsorted.

   END.
```

SortMenuConfig also calls SortMenu.Sort for the attention window menu, thus running SortMenuConfig will sort your attention window menu.

---

67. SortMenuConfig.bcd Lee F. Breisacher:OSBU South:Xerox 12-Aug-85 9:06:34
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>SortMenuConfig.bcd
  Documentation: none

---

68. SpecialDocumentEditor.bcd Richard Balcon:SBD-E:RX 14-Jan-86 6:51:38
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>SpecialDocumentEditor.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>SpecialDocumentEditor.doc of 27-Jan-86 6:51:21
  Description:

```
    -- File: SpecialDocumentEditor.Doc - last edit:
-- Balcon.rx          27-Jan-86 14:51:21

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
```

This program implements a simple text file editor for viewpoint. Its intended use is for creating and revising national data files e.g. Terminal Descriptor files.

It is based on the simple text editor already available in viewpoint but includes additional functionality in order to help multi-nationalisers, i.e. people creating multinational versions of the data files.

Additional features are:

a) Position - a function which allows the user to select a number relating to a position within a document and then set the current selection to that position (as implemented in XDE).

b) Find and FindNext - a function which allows the user to select a string and then search for it within the file. If found the string will be highlighted.
"Find" will find the first position of the string within the file. "FindNext" uses the last posi...

---

69. StarT.bcd Sybil A. Johnson:OSBU South:Xerox 27-Jan-85 13:12:55
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>StarT.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>StarT.doc of 31-Jan-85 15:58:05
  Description:

```
    --StarT.doc
```

This is a changed version of the prototype StarT test tool that allows for communication between BWSTestTool and StarT.

DOCUMENTATION ON HOW TO USE THE OLD STAR TEST TOOL

This message basically has two parts: the first part describes the fastest way to get used to using the tool without having to remember what all the options mean, the second part describes the tool in more detail and explains how you can tailor the tool to your needs. You don't need to read the second part to use the tool.

Also, please make sure you read the NOTES AND CAUTIONS section at the end of this message.

Part 1. HOW TO USE (FOR STARTERS):

GETTING STARTED

If this is the first time using the tool stuff the following line in the run line of Command Central without the quotes: "TestTool.TIP/-e StarT". Make sure that client switches in Command Central are not set to "O", otherwise you will return to the debugger with the "Bogus Tip Table"...

---

70. SwitchFolderType.bcd John A. Davidson:Roch0846:Xerox 26-Feb-88 9:25:21
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>SwitchFolderType.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>SwitchFolderType.doc of 26-Feb-88 10:12:07
  Description:

```
    --  SwitchFoldertype.Doc
-- Author: JDavidson:ROCH0846:XEROX
-- Date: 26-Feb-88
```

Description:

This program will take the selected Icon if its a Folder it changes the Icon's file type to that of a Mail Folder. If the selected icon is a Mail Folder then the icon's file type is changed to that of a Folder.

Why would you want to do this? Well, the Cruncher Tool works best on mail folders and this is the fastest way of going from Folder to Mail Folder to Crunched or in the opposite direction. Couldn't I do that with the File Props tool? Yes, but this Tool checks the file type if it isn't a mail folder or folder. It doesn't do anything to the icon. (Note: Since the Icon is open it does change a the last modified date stamp). This version was tested under VP 1.1.2 and is available as both a BCD and an Application folder.

---

71. TableWindow.bcd Lee F. Breisacher:OSBU South:Xerox 21-Nov-84 15:07:01
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>TableWindow.bcd
  Documentation: none

---

72. TableWindows.bcd Lee F. Breisacher:OSBU South:Xerox 18-Dec-84 10:08:13
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>TableWindows.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>TableWindows.doc of 18-Dec-84 15:17:05
  Description:
    -- File: TableWindows.doc - last edit:
-- Breisacher.ES 18-Dec-84 15:17:05

This doc describes two interfaces that support the creation and manipulation of a simple table in a window.

THESE INTERFACES ARE ONLY PARTIALLY IMPLEMENTED. There IS enough working to make them useful. A complete list of what's implemented and what isn't is given below.

Clients are encouraged to use the interfaces and are especially encouraged to comment on them. PLEASE let me know if you use these interfaces.

Here is a brief overview of each interface. There is a little more detail in the mesa file for each interface, including a list of potential features that could be added.

> XStringTableWindow

XStringTableWindow supports a table of "cells" each of which is backed by an XString. The storage for the strings is provided by the client. When the user edits a string, the string is copied and the edited copy is maintained by XStringTableWindow. The client can then call XStringTableWindow later to obtai...

---

73. TestBWSAT.bcd Bryan Yamamoto:OSBU North:Xerox 30-Jul-85 9:59:59
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>TestBWSAT.bcd
  Documentation: none

---

74. TipTest.bcd Stephen B. Tom:OSBU North:Xerox 13-Aug-85 16:52:06
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>TipTest.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>TipTest.doc of 13-Aug-85 16:32:30
  Description:
    Document name: TIPTest.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.0/Tools/
Last Revised by: Tom:PA 13-Aug-85 11:40:58
Owner: Tom:PA

TIP Test provides a way of testing TIP tables before you commit them to a system that you're going to try to run on. It lets you parse tables to check for syntax errors.

What is Required: TipTest.bcd

The tool consists of the usual message subwindow and form subwindow.

The commands available are:

Create - Calls TIP.CreateTable with the filename in the Name: field of the subwindow. If the parse is successful, the new TIP.Table is associated with the bottom subwindow of the tool. At this point, only real-estate events are sent to the window. This command calls the Destroy command before attempting any of this.

Destroy - removes the TIP.Table, if any, from the bottom window and releases the storage associated with it.

---

75. VMStats.bcd James G. Sandman:OSBU North:Xerox 3-Mar-86 9:41:52
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>VMStats.bcd
  Documentation: none

---

76. XDEKeyboard.bcd Janie D. Phillips:OSBU South:Xerox 10-Sep-85 12:11:21
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>XDEKeyboard.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>XDEKeyboard.doc of 10-Sep-85 11:07:27
  Description:
    -- File: XDEKeyboard.Doc - last edit:
-- JPhillips.ES 10-Sep-85 11:07:27

For DLion: retrieve XDEKeyboardPicture.bits to your data volume.
For Daybreak: retrieve DbkXDEKeyboardPicture.bits renaming it to XDEKeyboardPicture.bits.

Retrieve XDEKeyboard.bcd and drop it on your Loader Icon.

XDEKeyboard registers an XDE keyboard with the System keyboards. The main advantage to having an XDE keyboard is the presence of the arrows used in programming and program documentation.

To invoke the XDE keyboard hold down the keyboard key and press the soft key labeled XDE in the SoftKeys window. (It may be necessary to press the More key several times before the XDE keyboard choice is presented.)

To install the XDE keyboard as your default keyboard add the following to your User Profile:

```
[System]
DefaultKeyboard: XDE
```

---

77. XStringPrinter.bcd Lee F. Breisacher:OSBU South:Xerox 5-Oct-84 16:00:41
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>XStringPrinter.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>XStringPrinter.doc of 5-Oct-84 16:01:10
  Description:
    -- File: XStringPrinter.doc - last edit:
-- Sandman.pa 5-Oct-84 16:01:10

XStringPrinter.doc

XStringPrinter.bcd is a hack which registers a Printer with the debugger. Printers are client supplied routines that supplement CoPilots own display routines. It displays XString.ReaderBodys, XString.WriterBodys and Atom.ATOMs in human-readable form.

ReaderBodys get displayed as:

```
ReaderBody[context: <context>
<Bytes address>↑[<offset>..<limit>)"<characters with \bbb for wierd chars>"
```

WriterBodys get displayed as:

```
WriterBody[context: <context>
endContext: <endContext>
maxLimit: <maxLimit>, zone: <zone>
<Bytes address>↑[<offset>..<limit>)"<characters with \bbb for wierd chars>"
```

ATOMs get displayed as:

```
ATOM[<numeric value>] "<Textual name>"
```

Examples:

```
> r↑
ReaderBody[context: [suffixSize: 1, homogeneous: FALSE, prefix: 0]
3253366B↑[0..12)"<8>:<9>:<10>"

> w↑
WriterBody[context: [suffixSize: 1, homogeneous: FALSE, prefix: 0]
endContext: [suffixSize: 1, homogeneous: FALSE, prefix...
```

---

78. XStringTableWindow.bcd Lee F. Breisacher:OSBU South:Xerox 21-Nov-84 15:07:50
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.0>Tools>XStringTableWindow.bcd
  Documentation: none

Listing of hacks on [Alt:OSBU North:Xerox]<BWSHacks>4.1>Tools>
Documentation from [Alt:OSBU North:Xerox]<<BWSHacks>4.1>Doc>>
Listing created 16-Mar-89 17:42:20

	Hack Name	Last Writer	Create Date
1.	BackTalk.bcd	Harold J. Shinsato:OSBU North:Xerox	22-Apr-86 18:08:36
2.	DocumentStatistics.bcd	Bruce S. Lee:OSBU South:Xerox	14-Mar-86 12:28:18
3.	RunSomeAppls.bcd	Lee F. Breisacher:OSBU South:Xerox	7-Mar-86 13:24:45
4.	SortMenu.bcd	Harold J. Shinsato:OSBU North:Xerox	2-Aug-85 16:08:16
5.	SortMenuConfig.bcd	Harold J. Shinsato:OSBU North:Xerox	16-Apr-86 23:49:31

---

1. BackTalk.bcd Harold J. Shinsato:OSBU North:Xerox 22-Apr-86 18:08:36
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.1>Tools>BackTalk.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>BackTalk.doc of 22-Apr-86 18:55:02
Description:

```
-- File: BackTalk.doc
-- Last Revised by: Shinsato 22-Apr-86 17:55:00
-- Owner: Shinsato:OSBU North:Xerox

-- Copyright (C) Xerox Corporation 1986. All rights reserved.
```

OVERVIEW

BackTalk is a hack which allows Basic Workstation users to give a single message response to connection requests from the XDE hack Talk.bcd.

This hack allows BWS users to tell people who want to Talk.~ to them that they are unable to Talk.~ because they are in the BWS. Without this hack, users who request a connection will only get the uninformative response "GateStream ERROR: gapNotExported".

OPERATION

BackTalk listens for Talk requests from the XDE hack Talk.bcd, replies to the request, and then closes the connection.

BalkTalk will reply with the current user's name, for example:
"Harold J. Shinsato is in the Basic Workstation."
If noone is logged in, then BackTalk replies:
"This machine is in the Basic Workstation."

The user may specify a different reply with a UserProfile entry. In order ...

---

2. DocumentStatistics.bcd Bruce S. Lee:OSBU South:Xerox 14-Mar-86 12:28:18
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.1>Tools>DocumentStatistics.bcd
Documentation: none

---

3. RunSomeAppls.bcd Lee F. Breisacher:OSBU South:Xerox 7-Mar-86 13:24:45
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.1>Tools>RunSomeAppls.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>RunSomeAppls.doc of 7-Mar-86 14:27:42
Description:

```
-- File: RunSomeAppls.doc - last edit:
-- Breisacher.es 7-Mar-86 14:27:42

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
```

RunSomeAppls is sorta like the InitialCommand in an XDE User.cm. It is handy for developers that like to boot with the 'N switch (which causes nothing to automatically run at boot time), but still want to run some selected applications. It also allows several applications to be run all at once in the proper order after you've booted, without having to go through and pick them out one at a time.

RunSomeAppls has two features.

1. Run at startup

It starts specified applications and bcds when it is initially run. This will generally be used by putting RunSomeAppls on your CommandCentral run line. You specify the things to run in your WorkstationProfile as follows:

```
[RunSomeAppls]
Name: <application or bcd name>
Name: <application or bcd name>
Name: ...
```

The applications need not be in the proper order. RunSomeAppls...

---

4. SortMenu.bcd Harold J. Shinsato:OSBU North:Xerox 2-Aug-85 16:08:16
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.1>Tools>SortMenu.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.0>Doc>>SortMenu.doc of 12-Aug-85 9:45:50
Description:

```
-- File: SortMenu.doc - last edit:
-- Breisacher.ES 12-Aug-85 9:45:50

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.
```

SortMenuConfig.bcd EXPORTS SortMenu which will sort and keep a MenuData.MenuHandle sorted. Here is SortMenu.mesa:

```
SortMenu: DEFINITIONS = BEGIN

  Sort: PROCEDURE [menu: MenuData.MenuHandle];
  -- Whenever an item is added or swapped in 'menu',
  --  the menu is sorted.

  Unsort: PROCEDURE [menu: MenuData.MenuHandle];
  -- Sets the menu back to unsorted.

  END.
```

SortMenuConfig also calls SortMenu.Sort for the attention window menu, thus running SortMenuConfig will sort your attention window menu.

---

5. SortMenuConfig.bcd Harold J. Shinsato:OSBU North:Xerox 16-Apr-86 23:49:31
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.1>Tools>SortMenuConfig.bcd
  Documentation: none

Listing of hacks on [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>
Documentation from [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>
Listing created 16-Mar-89 17:38:16

	Hack Name	Last Writer	Create Date
1.	BootDMT.bcd	Seizo Umehara:OSBU South:Xerox	3-Oct-86 11:12:13
2.	BWSLassoOpen.bcd	Seizo Umehara:OSBU South:Xerox	5-May-87 16:55:22
3.	BWSSol.bcd	Gerard S. Zonus:OSBU South:Xerox	25-Jul-86 16:34:41
4.	ConvertBravoToVP.bcd	Harold J. Shinsato:OSBU North:Xerox	16-Jan-87 14:50:31
5.	CvTerm8.bcd	Jean-Marie R. de La Beaujardiere:OSBU North:Xerox	24-Jun-87 12:14:01
6.	DocumentStatistics.bcd	Bruce S. Lee:OSBU South:Xerox	22-May-87 14:16:07
7.	FormWindowLayoutTool.bcd	Perry A. Caro:OSBU North:Xerox	12-Aug-86 12:16:25
8.	Fractals.bcd	Stephen B. Tom:OSBU North:Xerox	6-Jun-86 17:33:27
9.	GetEnglishKeyboard.bcd	Nelson H. Ng:OSBU North:Xerox	23-Sep-86 10:00:15
10.	InvestigateGroups.bcd	Nanette E. Harter:OSBU North:Xerox	27-Jun-86 17:34:15
11.	Kaleidoscope.bcd	Stephen B. Tom:OSBU North:Xerox	5-Jun-86 14:11:20
12.	LoaderMenu.bcd	Lee F. Breisacher:OSBU South:Xerox	29-Apr-86 15:27:03
13.	MakeIPFileType.bcd	Frank H. Bowers:OSBU North:Xerox	28-Jan-87 17:11:25
14.	MessagingToolBWS.bcd	Lee F. Breisacher:OSBU South:Xerox	22-May-87 11:46:23
15.	RootPicture.bcd	Bruce K. Whittaker:sd:xerox	20-Sep-88 14:17:55
16.	ShowRules.bcd	Bruce S. Lee:OSBU South:Xerox	21-Aug-87 14:35:44
17.	ShowVM2.bcd	Martin F. N. Cooper:OSBU North:xerox	26-Apr-88 19:32:10
18.	SpaceOut.bcd	Stephen B. Tom:OSBU North:Xerox	5-Jun-86 18:16:02
19.	TableToFile.bcd	Larry Rosenberg:OSBU North:Xerox	18-May-88 13:09:57
20.	VolStat.bcd	Stephen B. Tom:OSBU North:Xerox	22-May 86 9:43:34
21.	VPActivity.bcd	Seizo Umehara:OSBU South:Xerox	30-Jan-88 0:23:25
22.	VPChat.bcd	Kenneth W. Lui:OSBU North:Xerox	16-Jul-86 15:35:00
23.	VPServer.bcd	Seizo Umehara:OSBU South:xerox	5-May-87 9:12:39
24.	VPTalk.bcd	Norin F. Saxe:OSBU North:Xerox	3-Jul-86 13:31:33

---

1. BootDMT.bcd Seizo Umehara:OSBU South:Xerox 3-Oct-86 11:12:13
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>BootDMT.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>BootDMT.doc of 3-Oct-86 11:34:03
Description:

```
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
--  BootDMT.doc of version #2: edited by S.Umehara, 86-Oct-03 11:34 am PDT
```

BootDMT works only in ViewPoint boot time. It displays a name of last loaded and started application and its load/start time in bouncing box as follows:

```
+---------------------------------+
|    Loaded: Reference Icons      |
|       time: 00:05:290           |
|   Started: Reference Icons      |
|       time: 00:05:290           |
+---------------------------------+
```

These information can be saved to a log file "BootDMT.log" in system-catalog (see Notes). Log-file shows # of disk IOs & page faults too (see attached example). At completion of boot, BootDMT generates a short beep just like old Star did in debug mode. Log and beep are controled by WorkstationProfile as:

```
[BootDMT]
Log: TRUE  -- TRUE|FALSE, default is FALSE
Beep: TRUE -- TRUE|FALSE, default is TRUE
```

Notes:
BootDMT.bcd must be renamed to BootDMT.autorun before use.
To take a log, boot switch 'd and 'D must be bo...

---

2. BWSLassoOpen.bcd Seizo Umehara:OSBU South:Xerox 5-May-87 16:55:22
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>BWSLassoOpen.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>BWSLassoOpen.doc of 5 May-87 18:25:37
Description:

```
-- Copyright (C) 1986, 1987 by Xerox Corporation. All rights reserved.
--  BWSLassoOpen.doc of version #2: edited by S.Umehara, 87-May-05  6:25 pm PDT
```

BWSLassoOpen.bcd changes icon's window opening behavior to minimize window repainting due to window overlapping. Before running BWSLassoOpen.bcd, add following parameter(s) to [Windows] section in UserProfile:

```
[Windows]
  SemiTiled: Both           -- Open|Props|Both|None default is None
  NoBoxForSemiTiled: MinBox -- AsIs|MinBox|MinHeihgt|DoLasso default is DoLasso
  DoLasso: Open             -- Open|Props|Both|None default is None
```

When "SemiTiled" is on (for Open and/or Props), BWSLassoOpen tries to find maximum box that does not overlap with any of currently opened windows, and open the window in it with possible maximum size. If no such box is available, "NoBoxForSemiTiled" specifies the action to be taken. "MinBox" opens with minimum dims [w: 161, h: 160]. "MinHeihgt" opens with minimum height (160). These can minimiz...

---

3. BWSSol.bcd Gerard S. Zonus:OSBU South:Xerox 25-Jul-86 16:34:41
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>BWSSol.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>BWSSol.doc of 25-Jul-86 15:54:31
Description:

```
-- File: BWSSol.doc - last edit:
-- Zonus.ES            25-Jul-86 15:54:31
```

-- Wagner 7-Jan-85 8:58:33

BWSSol is a the game of solitaire, Klondike to be precise.
Run BWSSol.bcd in the loader icon then select "Solitaire" from the herald menu

"Start" and "Quit" are self explanatory
"Game" allow you to choose between the normal version and the more difficult casino version in which you ante $50 to begin with. You then go through the stock a card at a time (as opposed to three at a time in the regular game). You are only allowed one pass through the stock. After finishing, you get $5 for each card promoted to the goals. Thus you need to promote ten cards to break even.

To pick up a card, point to it with the mouse and press Point. The card will be moved if there is only one place for it to go, otherwise the cursor will turn to mouse red and let you point to where it should go. To turn up a new card, just point to the s...

---

4. ConvertBravoToVP.bcd Harold J. Shinsato:OSBU North:Xerox 16-Jan-87 14:50:31
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>ConvertBravoToVP.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>ConvertBravoToVP.doc of 16-Jan-87 16:48:12
Description:
-- File: ConvertBravoToVP.doc
-- Last Edit: Shinsato 16-Jan-87 16:48:10

INTRODUCTION

Bravo is a document formatting and editting system which Xerox implemented on the Alto many years ago. An evolution of Bravo is BravoX. This conversion will not handle BravoX.

ConvertBravoToVP is an application which runs with the converter icon (Conversion Common Software) and the VP Document Editor (version 2.1) on ViewPoint 1.1. It converts documents from Bravo format to ViewPoint format. This software was originally part of Star. When Star was rewritten to give us ViewPoint, the software was moved to the application folder "VP File Conversion of Bravo Documents", but it was only available internally. This "hack" is the latest stage of the software which used to be a product.

INSTALLATION AND USE

Get ConvertBravoToVP.bcd, or the application folder "VP File Conversion of Bravo Documents". Run it in the Loader. Get...

---

5. CvTerm8.bcd Jean-Marie R. de La Beaujardiere:OSBU North:Xerox 24-Jun-87 12:14:01
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>CvTerm8.bcd
Documentation: none

---

6. DocumentStatistics.bcd Bruce S. Lee:OSBU South:Xerox 22-May-87 14:16:07
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>DocumentStatistics.bcd
Documentation: none

---

7. FormWindowLayoutTool.bcd Perry A. Caro:OSBU North:Xerox 12-Aug-86 12:16:25
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>FormWindowLayoutTool.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>FormWindowLayoutTool.doc of 29-Apr-85 15:27:06
Description:
-- File: FormWindowLayoutTool.doc - last edit:
-- Gobbel.pa 29-Apr-85 15:27:06
-- Diamond.PA 20-Aug-84 19:19:47
-- Breisacher.ES 19-Apr-84 14:52:04
-- Caro.pa 1-Apr-85 14:07:15 (but no fooling here!)

[4.0g Version: Please see Restrictions List]

The FormWindowLayoutTool allows a BWS programmer to graphically layout a FormWindow or PropertySheet. The tool automatically generates much of the Mesa source needed to produce a FormWindow or PropertySheet. These sources can then be compiled and executed and the resulting FormWindow will look like the one laid out earlier.

Description of the FormWindowLayoutTool:

After loading the tool in the Basic Workstation, a "FormWindow Layout Tool" menu item will be added to the Attention window menu. Bugging this brings up the layout tool, which has the following items:

ItemType: {Choice, Decimal, Integer, Boolean, Text, Command, TagOnly, Window}
...

---

8. Fractals.bcd Stephen B. Tom:OSBU North:Xerox 6-Jun-86 17:33:27
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>Fractals.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>Fractals.doc of 6-Jun-86 17:10:23
Description:
Document name: Fractals.doc
Home: (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools/
Last Revised by: SBT 6-Jun-86 17:10:22
Owner: SBT

OVERVIEW:
Fractals replaces the current logon DMT and is activated by the "Logoff" command in the Attention window. To deactivate press any key.

BACKGROUND:
Fractals, like its Tajo counterpart, generates fractal "landscapes". The basic algorithm, which follows, is recursive in nature (although the implementation is not). We start with a random triangle which is fairly large relative to the window:

1) For each unperturbed side of the triangle, displace its midpoint a random distance along a line perpendicular to the side, either inwards or outwards.
2) Connect the midpoints to each other, and repeat for each of the four new triangles thus created.

WHAT IS REQUIRED TO RUN:
Copy Fractals.bcd from (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools onto the loader if you are in ViewPoint o...

---

9. GetEnglishKeyboard.bcd Nelson H. Ng:OSBU North:Xerox 23-Sep-86 10:00:15
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>GetEnglishKeyboard.bcd
Documentation: none

---

10. InvestigateGroups.bcd Nanette E. Harter:OSBU North:Xerox 27-Jun-86 17:34:15
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>InvestigateGroups.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>InvestigateGroups.doc of 1-Jul-86 9:40:39
Description:
-- File: InvestigateGroups.doc - Last edit:
-- Harter.PA 1-Jul-86 9:40:39

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.

The Investigate Groups application lists the groups (NS distribution lists) a user is a member of in the Clearinghouse. The tool runs in the ViewPoint (BWS4.2) environment. Investigate Groups provides functionality similar to that of an XDE hack, DLDetective. After loading the Investigate Groups bcd, the tool Icon can be found in the Basic Icons folder of the Workstation divider in the Directory.

The tool will has the following functionality:
1) Lists the groups in a specified set of domains that the named user is a member of.
2) Lists all the groups in a specified set of domains (and indicates the ones the user is a member of).
3) Removes the named user from the groups in the domains, unconditionally or through confirmation.

Property Sheet

The property sheet fields are defined as follows:
-Icon Name allows the user to define the na...

---

11. Kaleidoscope.bcd Stephen B. Tom:OSBU North:Xerox 5-Jun-86 14:11:20
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>Kaleidoscope.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>Kaleidoscope.doc of 5-Jun-86 14:36:59
Description:
Document name: Kaleidoscope.doc
Home: (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools/
Last Revised by: SBT 5-Jun-86 14:26:55
Owner: SBT

Copyright (C) 1984, 1985, 1986 by Xerox Corporation. All rights reserved.

OVERVIEW:
Kaleidoscope replaces the current logon DMT and is activated by the "Logoff" command in the Attention window. To deactivate press any key.

WHAT IS REQUIRED TO RUN:
Copy Kaleidoscope.bcd from (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools onto the loader if you are in ViewPoint or run it from the Command Central from XDE. For the non-programmer review documentation on:
(Alt:OSBU North:Xerox)BWSHacks/HowToRunHacks.doc

RESTRICTIONS:
None. Runs on both Dandelions and Daybreaks and is compatible with BWS 4.0, 4.1, 4.2*.

---

12. LoaderMenu.bcd Lee F. Breisacher:OSBU South:Xerox 29-Apr-86 15:27:03
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>LoaderMenu.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>LoaderMenu.doc of 29-Apr 86 15:34:36
Description:
-- File: LoaderMenu.doc - last edit:
-- Breisacher.es 29-Apr-86 15:32:48

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.

LoaderMenu adds a popup menu to the loader icon with the single item "Run". Select a .bcd file anywhere on your desktop or in a folder, chord over the loader icon and bug Run.

LoaderMenuImpl.mesa also serves as an example client for the icon popup menu feature. Here's the relevant code:

```
    GenericProc: Containee.GenericProc = {
      SELECT atom FROM
        menu =>
          BEGIN
          run: XString.ReaderBody ← XString.FromSTRING ["Run"L];
          name: XString.ReaderBody ← XString.FromSTRING ["Loader"L];
          title: MenuData.ItemHandle ← MenuData.CreateItem[zone: NIL, name: @name, proc: NIL];
          items: ARRAY[0..1) OF MenuData.ItemHandle ← [
             MenuData.CreateItem[zone: NIL, name: @run, proc: Run]];
          menu: MenuData.MenuHandle ← MenuData.CreateMenu[zone: NIL, title: title, array: DESCRIPTOR[items]];
          RETURN [menu];
          END;
        freeMenu =>
          BEGIN
          menu:...
```

---

13. MakeIPFileType.bcd Frank H. Bowers:OSBU North:Xerox 28-Jan-87 17:11:25
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>MakeIPFileType.bcd
  Documentation: none

---

14. MessagingToolBWS.bcd Lee F. Breisacher:OSBU South:Xerox 22-May-87 11:46:23
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>MessagingToolBWS.bcd
  Documentation: none

---

15. RootPicture.bcd Bruce K. Whittaker:sd:xerox 20-Sep-88 14:17:55
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>RootPicture.bcd
  Documentation: none

---

16. ShowRules.bcd Bruce S. Lee:OSBU South:Xerox 21-Aug-87 14:35:44
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>ShowRules.bcd
  Documentation: none

---

17. ShowVM2.bcd Martin F. N. Cooper:OSBU North:xerox 26-Apr-88 19:32:10
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>ShowVM2.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>ShowVM2.doc of 18-Jan-88 21:05:46
  Description:
    -- File: ShowVM2.doc - last edit:
-- Cooper:OSBU North:Xerox 18-Jan-88 21:05:46

ShowVM2 is a tool for monitoring virtual memory usage in BWS. It periodically scans the VM map and reports various statistics in a window, and optionally in a log file.

Running the tool
================

To run ShowVM2, retrieve ShowVM2.bcd from your local BWS hacks directory and drop it onto the Loader icon on the desktop. When started, the item "Show VM Usage" is added to the Attention menu. Selecting this item will cause the VM Usage window to appear on the desktop. Note that the menu item is then removed from the Attention menu, so that only one such window may be open at any time. To stop running the tool, simply invoke the Close command on the VM Usage window header.

Window items
============

The items which appear in the VM Usage window, and their interpretation, are discussed in this section.

1. Show Map

This boolean item determ...

---

18. SpaceOut.bcd Stephen B. Tom:OSBU North:Xerox 5-Jun-86 18:16:02
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>SpaceOut.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>SpaceOut.doc of 5-Jun-86 18:45:17
  Description:
    Document name: SpaceOut.doc
Home: (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools/
Last Revised by: SBT 5-Jun-86 18:45:15
Owner: SBT

OVERVIEW:
SpaceOut replaces the current logon DMT and is activated by the "Logoff" command in the Attention window. To deactivate press any key.

WHAT IS REQUIRED TO RUN:

Copy SpaceOut.bcd from (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools onto the loader if you are in ViewPoint or run it from the Command Central from XDE. For the non-programmer review documentation on:
(Alt:OSBU North:Xerox)BWSHacks/HowToRunHacks.doc

RESTRICTIONS:
None. Runs on both Dandelions and Daybreaks and is compatible with BWS 4.0, 4.1, 4.2*.

---

19. TableToFile.bcd Larry Rosenberg:OSBU North:Xerox 18-May-88 13:09:57
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>TableToFile.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>TableToFile.doc of 2-Nov-88 16:09:03
Description:
-- File: TableToFile.doc - last edit:
-- Lrosenberg:OSBU North:Xerox 2-Nov-88 16:09:03

TableToFile is really two programs in one. When loaded, it registers two commands in the attention menu. The two commands are used together to allow editing ViewPoint Tables in a simplier fashion than editing the table directly.

o "Table => File" take as input the selection, which can be any number of documents or folder of documents, and enumerates the content of all folders it encounters. It converts each document it finds one at a time. All tables found in one document have their contents written to a file. Fonts from the table cells are preserved. Output of this program is a folder called "Table to File" which contains one document for every document it encountered. If an old output version of the same file already exists, it will be deleted.

WARNING: ALL DATA OTHER THAN TABLE CELLS' CONTENT IS LOST.

A single table is wr...

---

20. VolStat.bcd Stephen B. Tom:OSBU North:Xerox 22-May-86 9:43:34
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>VolStat.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>VolStat.doc of 22-May-86 11:30:46
Description:
Document name: VolStat.doc
Home: (Alt:OSBU North:Xerox)BWSHacks/4.2/Tools/
Last Revised by: SBT 22-May-86 11:30:44
Owner: SBT

OVERVIEW:

Essentially a mirror to its Tajo counterpart (by Nannette Simpson and Dan Conde) which emulates Othello's "Describe Volume" command. VolStat retrieves information about the state of the logical volumes on the disk. Specifically, it lists the processor ID (useful for remote debugging) along with the volume name, type, status, size in pages, number of free pages, relation of the volume to the system volume, microcode, germ, bootfile, and bootfile switches.

CURRENT FEATURES AND USER INTERFACE:

VolStat (an abbreviation for volume statistics) registers the menu command, "VolStat" in the Attention window. Invoking the menu command will post the message:

"Would you like to save the volume statistics in a file? [YES|NO]"

Selecting [NO] will display the status information in the Atten...

---

21. VPActivity.bcd Seizo Umehara:OSBU South:Xerox 30-Jan-88 0:23:25
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>VPActivity.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>VPActivity.doc of 30-Jan-88 0:46:08
Description:
-- Copyright (C) 1988 by Xerox Corporation. All rights reserved.
-- VPActivity.doc : edited by STU, 88-Jan-30 12:46 am PST

VPActivity.bcd is a BWSActivity.bcd with following changes:

o Smaller display window.
o Default bars are CPU & Disk IO.
o VM page was deleted.
o Popup menu is triggered by mouse chords on window.
o Window can be moved to any place by Point-Down & dragging.
o Window place and number of bars are remembered until next Logon.
o Automatic popup over the "Full screen" of Tajo in VP.

Note:
(1) It will be better to change suffix to ".autorun" before copy to the loader.
(2) VPActivity runs on BWS 4.0 (VP1.0) without Popupmenu.
(3) VPActivity runs on BWS 4.3 (VP2.0).

---

22. VPChat.bcd Kenneth W. Lui:OSBU North:Xerox 16-Jul-86 15:35:00
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>VPChat.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>VPChat.doc of 16-Jul-86 15:41:43

Description:
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
-- VPChat.doc
-- Lui 16-Jul-86 15:34:50

VPChat is a Viewpoint version of the XDE's Chat (see documentation for Chat in MUG). There is an application version of VPChat called Chat, and a plain bcd version called VPChat.bcd. They are functionally identical.

VPChat registers an item in the Attention menu called "VPChat," which when invoked creates a Chat window. A Chat window has a message window, a form window, and a tty windows. The form window has the following items:

Connect	Command item for opening a connection
Disconnect	Command item for closing a connection
Another	Command item for creating another Chat window
Host	Text item for specifying name or net address of remote workstation.
HostType	Chat has three modes of operation. (see documentation for Chat in MUG).

Addditional Chat windows can be created either by reselecting the item in the Attention menu, or by clicking over the "Another" command....

---

23. VPServer.bcd Seizo Umehara:OSBU South:xerox 5-May-87 9:12:39
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>VPServer.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>VPServer.doc of 5-May-87 11:09:03
Description:
-- Copyright (C) 1987 by Xerox Co., Ltd. All rights reserved.
-- VPServer.doc : last edited by Umehara:OSBU South, 87-May-05 11:09 am PDT

VPServer is a BWS counterpart of MFileServer. VPServer enables FileTool (from XDE) or WSIcon hack (from BWS) to access files in workstation running BWS.

[I] Directory rule

Any of following path-names can be used in directory field of FileTool & WSIcon.

empty
Desktop/<sub path name>
Help/<sub path name>
SystemFile/<sub path name>
<real path name>

<sub path name> is a name of folder(s) and can be empty.
<real path name> is a real name of BWS filing hierarchy. For example, real pathname of a adf file in application "VP Example" is:

Directory: CatalogFor10476B/VP Example/
File: example.adf

[II] Profile parameters

[II-1] WorkstationProfile

Values in WorkstationProfile are used when workstation is Logged off.

[System]
DefaultUser: <fullname>

[VPServer]
Running: TRUE|FALSE -- default is TRUE
ShowActivity:...

---

24. VPTalk.bcd Norin F. Saxe:OSBU North:Xerox 3-Jul-86 13:31:33
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.2>Tools>VPTalk.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.2>Doc>>VPTalk.doc of 2-Jul-86 14:06:55
Description:
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
-- VPTalk.doc
-- NFS 2-Jul-86 14:06:53

VPTalk is a Viewpoint version of the XDE hack Talk (see Talk.doc on the <Hacks>Doc> directory). VPTalk and Talk allow two people to have a conversation between workstations. VPTalk can connect to and receive connections from either the XDE Talk or VPTalk. Both parties must have one of the two versions running on their machines.

There is an application version of VPTalk called Talk, and a plain bcd version called VPTalk.bcd. They are functionally identical.

VPTalk registers an item in the Attention menu called "Talk," which when invoked creates a talk window. A talk window has a message window, a form window, and two tty windows. The form window has the following items:

Connect	Command item for opening a connection
Disconnect	Command item for closing a connection
Another	Command item for creating another talk window
Host	Text item for specifying name or net address ...

Listing of hacks on [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>
Documentation from [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>
Listing created 16-Mar-89 17:32:00

	Hack Name	Last Writer	Create Date
1.	AutoApplize.bcd	Deborah J. Lewis:OSBU South:Xerox	14-Sep-88 14:52:33
2.	BootDMT.bcd	Seizo Umehara:IWA:Fuji Xerox	10-Aug-88 2:41:42
3.	BootingMenu.bcd	Seizo Umehara:OSBU South:xerox	25-Nov-87 19:22:18
4.	BWSHeapCheckTool.bcd	Makoto Mita:OSBU South:Xerox	9-Oct-87 13:38:55
5.	BWSLassoOpen.bcd	Seizo Umehara:OSBU South:Xerox	26-Apr-88 11:40:40
6.	BWSNeverZoneCheckTool.bcd	Makoto Mita:OSBU South:Xerox	17-Nov-87 14:07:15
7.	ConvertBravoToVP.bcd	Harold J. Shinsato:OSBU North:Xerox	29-Jan-88 17:28:29
8.	DefaultIconMenu.bcd	Perry A. Caro:OSBU North:Xerox	30-Jan-87 15:25:30
9.	DocumentStatistics.bcd	Bruce S. Lee:OSBU South:Xerox	9-Sep-87 19:20:41
10.	FlushOut.bcd	J. Paul Holbrook:OSBU South:Xerox	7-Mar-85 13:30:26
11.	FormWindowLayoutTool.bcd	Frank H. Bowers:OSBU North:Xerox	22-Jun-87 13:27:12
12.	Fractals.bcd	Samuel C. Yang:OSBU South:xerox	10-Mar-89 14:28:08
13.	ListAppls.bcd	Tetsuo Seto:IWA:Fuji Xerox	20-Apr-88 18:30:19
14.	LogDisplayWindowConfig.bcd	Kenneth J. Guzik:OSBU North:Xerox	20-Apr-88 17:12:21
15.	LogFile.bcd	Deborah J. Lewis:OSBU South:Xerox	14-Sep-88 14:22:46
16.	LogFileImpl.bcd	Deborah J. Lewis:OSBU South:Xerox	14-Sep-88 14:31:36
17.	Maintain2ForXDEinVP.bcd	GOBIN Richard:CNADSI:RXF	12-May-88 21:48:06
18.	Music.bcd	Don Burrell:OSBU North:Xerox	3-Nov-86 17:02:13
19.	NumberingAssistantConfig.bcd	Akira Hirose:IWA:Fuji Xerox	21-Nov-88 20:46:21
20.	RootPicture.bcd	Bruce K. Whittaker:sd:xerox	1-Oct 88 15:59:03
21.	RunSomeAppls.bcd	William J. Maybury:OSBU South:Xerox	11-Mar-88 9:43:01
22.	SelectionQueryTool.bcd	Deborah J. Lewis:OSBU South:Xerox	16-Sep-88 14:34:18
23.	ShowPage.bcd	Yoshito Minamizaki:IWA:Fuji Xerox	30-Apr-88 0:47:05
24.	ShowRules.bcd	Bruce S. Lee:OSBU South:Xerox	5-Oct-87 10:26:32
25.	ShowVM.bcd	Makoto Mita:OSBU South:Xerox	10-Dec-86 9:37:30
26.	ShowVM2.bcd	Martin F. N. Cooper:OSBU North:xerox	26-Apr-88 18:58:46
27.	SpaceEater.bcd	Jeremy M. Goodell:OSBU South:Xerox	22-Jan-88 18:14:44
28.	SunTIPTest.bcd	Alex Dianysian:OSBU South:xerox	17-Feb-89 11:34:26
29.	SwatTool.bcd	Michael D. Kupfer:OSBU North:Xerox	1-Oct-87 22:30:16
30.	TableToFile2.bcd	Larry Rosenberg:OSBU North:Xerox	25-Oct-88 15:29:14
31.	VPActivity.bcd	Seizo Umehara:OSBU South:Xerox	10-May-88 14:13:03
32.	VPCHBrowser.bcd	Matt Thompson:OSBU North:Xerox	21-Nov-88 0:42:05

---

1. AutoApplize.bcd Deborah J. Lewis:OSBU South:Xerox 14-Sep-88 14:52:33
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>AutoApplize.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>AutoApplize.doc of 1-Aug-88 12:34:32
Description:
-- File: AutoApplize.doc - last edit:
-- Lewis:OSBU South:Xerox 1-Aug-88 12:34:32

OVERVIEW

AutoApplize is a tool which automatically creates applications according to the specifications in a BuildScript description file. The purpose of the tool is to eliminate repetitive manual steps involved in building an application. AutoApplize will fetch all the files required in the application and create the application with the requested properties, thus eliminating two time-consuming and error-prone manual steps in the applizing process.

The user must still build message files for the application manually prior to running AutoApplize. Similarly, the AutoApplize user must manually store message files and applications to the desired file directory. If you are interested in a tool which does either of these functions automatically, contact the Product Support group about their application-building tool.

R...

---

2. BootDMT.bcd Seizo Umehara:IWA:Fuji Xerox 10-Aug 88 2:41:42
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>BootDMT.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>BootDMT.doc of 10-Aug-88 3:08:05
Description:
-- Copyright (C) 1986, 1987, 1988 by Xerox Corporation. All rights reserved.
-- BootDMT.doc: last edited by STU, 88-Aug-10 7:08 pm JST

BootDMT is a DMT that works during ViewPoint is booting. BootDMT displays a name of loading and starting application with load/start time in bouncing box as follows:

```
+-------------------------------------------------+
|       Reference Icons started (0'05"9)          |
| Remote Printing: loaded (0'28"3) starting ...   |
+-------------------------------------------------+
```

1st line shows name of the last started application and time (m'ss) used for starting. 2nd line shows currently loading and/or starting application.

Installation note: BootDMT.bcd must be renamed to BootDMT.autorun.

<<< for developer: using boot-switch 'd (small d) >>>

If boot switch 'd is set ON, above application loading data are logged to a file "BootDMT.log". Log file is stored in system-catalog. Log-file shows loading and stating time, number of disk IOs, page faults and system b...

---

3. BootingMenu.bcd Seizo Umehara:OSBU South:xerox 25-Nov-87 19:22:18
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>BootingMenu.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>BootingMenu.doc of 16-May-85 16:55:25
Description:
-- File: BootingMenu.doc - last edit:
-- JGS 16-May-85 16:55:25

BootingMenu.bcd is a VP equivalent of booting menu in Tajo's HeraldWindow.

BootingMenu registers a the "Booting" command with the system menu.

When it is invoked, a popup menu appears with the boot choices. In addition to entries for each logical volume the menu contains the following choices:

File	Selection must be a bootfile on the local volume.
Boot Button	Boots from the boot button
Net Boot	Etherboots the bootfile specified by current selection which must be a string and a valid bootfile number. If the selection is not a string it defaults to booting othello.
Set Switches	Sets the boot switches to the current selection which must be a string.
Reset Switches	Resets the boot switches.

---

4. BWSHeapCheckTool.bcd Makoto Mita:OSBU South:Xerox 9-Oct-87 13:38:55
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>BWSHeapCheckTool.bcd
Documentation: none

---

5. BWSLassoOpen.bcd Seizo Umehara:OSBU South:Xerox 26-Apr-88 11:40:40
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>BWSLassoOpen.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>BWSLassoOpen.doc of 26-Apr-88 15:32:14
Description:
-- Copyright (C) 1986, 1987, 1988 by Xerox Corporation. All rights reserved.
-- BWSLassoOpen.doc of version #4: edited by STU, 88-Apr-26 3:32 pm PDT

BWSLassoOpen.bcd changes icon's window opening behavior to minimize window repainting due to window overlapping. Before running BWSLassoOpen.bcd, add following parameter(s) to [Windows] section in UserProfile:

```
[Windows]
  SemiTiled: Both           -- Open|Props|Both|None default is None
  NoBoxForSemiTiled: MinBox -- AsIs|MinBox|MinHeihgt|DoLasso default is DoLasso
  DoLasso: Open             -- Open|Props|Both|None default is None
```

When "SemiTiled" is ON (for Open and/or Props), BWSLassoOpen tries to find maximum width box that does not overlap with any of currently opened windows, and open new window in it with possible maximum size. If no such box is available, "NoBoxForSemiTiled" specifies the action to be taken. "MinBox" opens with minimum dims [w: 161, h: 160]. "MinHeihgt" opens with minimum height (160). These can m...

---

6. BWSNeverZoneCheckTool.bcd Makoto Mita:OSBU South:Xerox 17-Nov-87 14:07:15
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>BWSNeverZoneCheckTool.bcd
Documentation: none

---

7. ConvertBravoToVP.bcd Harold J. Shinsato:OSBU North:Xerox 29-Jan-88 17:28:29
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>ConvertBravoToVP.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>ConvertBravoToVP.doc of 1-Feb-88 12:04:04
Description:
-- File: ConvertBravoToVP.doc
-- Last Edit: Shinsato 1-Feb-88 12:04:03

INTRODUCTION

Bravo is a document formatting and editting system which Xerox implemented on the Alto many years ago. An evolution of Bravo is BravoX. This conversion will not handle BravoX.

ConvertBravoToVP is an application which runs with the converter icon (Conversion Common Software) and the VP Document Editor (version 3.0) on ViewPoint 2.0. It converts documents from Bravo format to ViewPoint format. This software was originally part of Star. When Star was rewritten to give us ViewPoint, the software was moved to the application folder "VP File Conversion of Bravo Documents", but it was only available internally. This "hack" is the latest stage of the software which used to be a product.

INSTALLATION AND USE

Get ConvertBravoToVP.bcd, or the application folder "VP File Conversion of Bravo Documents". Run it in the ...

---

8. DefaultIconMenu.bcd Perry A. Caro:OSBU North:Xerox 30-Jan-87 15:25:30
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>DefaultIconMenu.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>DefaultIconMenu.doc of 30-Jan-87 15:40:10
Description:

-- File: DefaultIconMenu.doc
-- Last Revised by: Caro 30-Jan-87 14:01:12

This little hack is a quick and handy way to look at the full name of an icon.

This hack replaces the default Containee.Implementation with one that has a new genericProc. The new genericProc provides facilities for handling the Menu and FreeMenu atoms. The menu created has the title "Icon Name" and the single element of the menu is the name of the icon. This hack is incremental to the other default Implementation items, including the genericProc (which calls the old genericProc with any other atom besides Menu or FreeMenu, as is customary), so everything else works the same as before.

Previously, the only way to look at an icon's name is to Props it or to put it in a container. This hack allows you to see the full name of the icon that's on the desktop much more quickly.

Bugs To: Caro:OSBU North:Xerox

DefaultIcon...

---

9. DocumentStatistics.bcd Bruce S. Lee:OSBU South:Xerox 9-Sep-87 19:20:41
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>DocumentStatistics.bcd
Documentation: none

---

10. FlushOut.bcd J. Paul Holbrook:OSBU South:Xerox 7-Mar-85 13:30:26
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>FlushOut.bcd
Documentation: none

---

11. FormWindowLayoutTool.bcd Frank H. Bowers:OSBU North:Xerox 22-Jun-87 13:27:12
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>FormWindowLayoutTool.bcd
Documentation: none

---

12. Fractals.bcd Samuel C. Yang:OSBU South:xerox 10-Mar-89 14:28:08
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>Fractals.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>Fractals.doc of 10-Mar-89 15:03:06
Description:
Document: [Alt:OSBU North:Xerox]BWSHacks/4.3/Doc/Fractals.doc
Tool: [Alt:OSBU North:Xerox]BWSHacks/4.3/Tools/Fractals.bcd

Last Revised by:
Yang 10-Mar-89 15:03:04
SBT 6-Jun-86 17:10:22

Owner: Yang:OSBU South:Xerox

OVERVIEW:
Fractals replaces the current logon DMT and is activated by the "Logoff" command in the Attention window. To deactivate press any key.

BACKGROUND:
Fractals, like its Tajo counterpart, generates fractal "landscapes". The basic algorithm, which follows, is recursive in nature (although the implementation is not). We start with a random triangle which is fairly large relative to the window:

(1) For each side of the triangle, displace its midpoint a random distance along a line perpendicular to the side, either inwards or outwards.
(2) Connect the midpoints to each other, and repeat [starting at step (1)] for each of the four new triangles thus created.

WHAT IS REQUIRED TO R...

---

13. ListAppls.bcd Tetsuo Seto:IWA:Fuji Xerox 20-Apr-88 18:30:19
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>ListAppls.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>ListAppls.doc of 20-Apr-88 19:05:13
Description:
-- File: ListAppls.doc - last edit:

-- Seto:IWA:Fuji Xerox 21-Apr-88 9:19:59

FUNCTION

Makes a list of
all files found in the system catalog with creation time(Greenwich mean time),
content files for application folders sorted on priority,
ADF file contents for application folders.
and check if the 'priority' and 'requires' fields of ADF file are consistent.

HOW TO USE

Run ListApplis.bcd which adds "List Appls" command to the attention menu.

You do NOT have to run applicaitons. ListApplis should also work on the system booted with 'N switch.

Log file named "ApplsList.log" is created on your desktop when finished.

Skelton of the log file is;
"System catalog as of 20-Apr-88 14:34:41",
Non-application folder files in alphabetical order,
"XXX appliction folders out of YYY files found in the system catalog.",
Invisible appl. folders in priority order,
Visible appl. folder...

---

14. LogDisplayWindowConfig.bcd Kenneth J. Guzik:OSBU North:Xerox 20-Apr-88 17:12:21
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>LogDisplayWindowConfig.bcd
Documentation: none

---

15. LogFile.bcd Deborah J. Lewis:OSBU South:Xerox 14-Sep-88 14:22:46
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>LogFile.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>LogFile.doc of 16-Sep-88 10:00:23
Description:

```
-- File: LogFile.doc - last edit:
-- Lewis:OSBU South:Xerox 16-Sep-88 10:00:23

-- Copyright (C) 1988 by Xerox Corporation. All rights reserved.
```

LogFile is a simple utility package which can be used by ViewPoint programmers to create SimpleText documents. It was invented to provide a log file capability for BWS hacks. Clients must bind a copy of LogFileImpl into their software or otherwise arrange for it to be available at runtime.

```
-- ========
-- TYPES ==
-- ========
```

A LogFile.Handle is a handle on an open log file.

```
-- ========================
-- FILE LEVEL OPERATIONS ==
-- ========================
```

The file-level operations are Create, Open, Close, and Destroy. A typical client will Create a new log file, append information using various content operations, and Close the file. The client can specify the directory in which the LogFile will be created; the default is to create it on the desktop.

Open allows a client to open an existing log file and append additional text to ...

---

16. LogFileImpl.bcd Deborah J. Lewis:OSBU South:Xerox 14-Sep-88 14:31:36
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>LogFileImpl.bcd
Documentation: none

---

17. Maintain2ForXDEinVP.bcd GOBIN Richard:CNADSI:RXF 12-May-88 21:48:06
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>Maintain2ForXDEinVP.bcd
Documentation: none

---

18. Music.bcd Don Burrell:OSBU North:Xerox 3-Nov-86 17:02:13
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>Music.bcd
Documentation: none

---

19. NumberingAssistantConfig.bcd Akira Hirose:IWA:Fuji Xerox 21-Nov-88 20:46:21
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>NumberingAssistantConfig.bcd
Documentation: none

---

20. RootPicture.bcd Bruce K. Whittaker:sd:xerox 1-Oct-88 15:59:03
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>RootPicture.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>RootPicture.doc of 13-Sep-88 14:40:05
Description:

```
-- File: RootPicture.doc - Last edit:
-- Whittaker:SD:Xerox 13-Sep-88 14:40:05
Document name: RootPicture.doc
NS home: [Alt:OSBU North:Xerox]BWSHacks/4.3
Last Revised by: Tom:PA 14-Jan-86 17:07:10
```

Owner: Tom:PA

RootPicture registers the menu command, "RootPicture" in the Attention window. By toggling the menu item, a tiny window located in the upper righthand side of the screen will be activated/deactivated. Chording over this window will invoke a menu listing all bitmap files in the system catalog. These include files with the name of either "*.press" or "*.res," or files of type VP canvases. Selecting a particular choice will display its corresponding rootpicture. Invoking the item "Desktop" will display the customary desktop gray.

What is Required:
  a) RootPicture.bcd
  b) One or more bitmap files in the system catalog. Note any file that worked with the Tajo 11.0 version of RootP...

---

21. RunSomeAppls.bcd William J. Maybury:OSBU South:Xerox 11-Mar-88 9:43:01
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>RunSomeAppls.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>RunSomeAppls.doc of 1-Aug-87 17:52:41
  Description:
    -- File: RunSomeAppls.doc - last edit:
-- Maybury:OSBU South:Xerox 1-Aug-87 17:52:41
-- Breisacher.es 7-Mar-86 14:27:42

RunSomeAppls is sorta like the InitialCommand in an XDE User.cm. It is handy for developers that like to boot with the 'N switch (which causes nothing to automatically run at boot time), but still want to run some selected applications. It also allows several applications to be run all at once in the proper order after you've booted, without having to go through and pick them out one at a time.

RunSomeAppls has three features.

  1. Run at startup

It starts specified applications and bcds when it is initially run. This will generally be used by putting RunSomeAppls on your CommandCentral run line. You specify the things to run in your WorkstationProfile as follows:

    [RunSomeAppls]
    Name: <application or bcd name>
    Name: <application or bcd name>
    Name: ...

The appl...

---

22. SelectionQueryTool.bcd Deborah J. Lewis:OSBU South:Xerox 16-Sep-88 14:34:18
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>SelectionQueryTool.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>SelectionQueryTool.doc of 16-Sep-88 16:13:53
  Description:
    -- File: SelectionQueryTool.doc - last edit:
-- Lewis:OSBU South:Xerox 16-Sep-88 16:13:53

The SelectionQueryTool provides a quick way to determine whether a selection supports some selection target type of interest. It was created so that its author would not have to engage in the onerous task of reading documents code in order to answer questions of the form "Does a document support selection target mumblefratz?".

To run the tool, retrieve SelectionQueryTool.bcd from the <BWSHacks> directory and run it using the ViewPoint Loader.

The tool registers a command "Selection Query Tool" in the desktop Attention window. Invoking the command brings up an option sheet containing a command item CanYouConvert! and a choice item named "Selection target". All of the predefined BWS Selection.Target types are supported. Choose the target type of interest (e.g., "string"), select any object (e.g., some text in a document), and b...

---

23. ShowPage.bcd Yoshito Minamizaki:IWA:Fuji Xerox 30-Apr-88 0:47:05
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>ShowPage.bcd
  Documentation: none

---

24. ShowRules.bcd Bruce S. Lee:OSBU South:Xerox 5-Oct-87 10:26:32
  Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>ShowRules.bcd
  Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>ShowRules.doc of 23-Nov-88 11:53:54
  Description:
    -- File: ShowRules.doc - last edit:
-- Lewis:OSBU South:Xerox 23-Nov-88 11:53:54
-- BLee:OSBU South:Xerox 6-Aug-87 17:27:19 PDT

OVERVIEW

The Show Rules tool allows you to extract properties of CUSP buttons, tables, and fields from a document into a separate report document which can be displayed and printed. In particular, CUSP programs and table/field fill-in rules can be extracted. The ability to dump this information is useful because it can only be accessed in the source document by opening a property sheet and viewing the information online.

This documentation describes the enhanced version of Show Rules available on <BWSHacks>4.3> for VP 2.0.

OPERATION

Retrieve a copy of the "Show Rules" application from <BWSHacks>4.3>Tools and copy it to the Loader icon on your desktop. The autorun property of the application can be turned on via the application's property sheet if you wish to...

---

25. ShowVM.bcd Makoto Mita:OSBU South:Xerox 10-Dec-86 9:37:30
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>ShowVM.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>ShowVM.doc of 10-Dec-86 9:42:00
Description:
-- File: ShowVM.doc - last edit:
-- Mita.ES 10-Dec-86 9:40:08

ShowVM is checking both VM and Backing it reports VM usage and Backing usage.

To use

Run ShowVm on BWS and select attention menu Show Virtual Memory. youcan tell how to use.

---

26. ShowVM2.bcd Martin F. N. Cooper:OSBU North:xerox 26-Apr-88 18:58:46
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>ShowVM2.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>ShowVM2.doc of 13-Jan-88 8:55:38
Description:
-- File: ShowVM2.doc - last edit:
-- Cooper:OSBU North:Xerox 13-Jan-88 8:55:38

ShowVM2 is a tool for monitoring virtual memory usage in BWS. It periodically scans the VM map and reports various statistics in a window, and optionally in a log file.

Running the tool
==========

To run ShowVM2, retrieve ShowVM2.bcd from your local BWS hacks directory and drop it onto the Loader icon on the desktop. When started, the item "Show VM Usage" is added to the Attention menu. Selecting this item will cause the VM Usage window to appear on the desktop. Note that the menu item is then removed from the Attention menu, so that only one such window may be open at any time. To stop running the tool, simply invoke the Close command on the VM Usage window header.

Window items
==========

The items which appear in the VM Usage window, and their interpretation, are discussed in this section.

1. Show Map

This boolean item determ...

---

27. SpaceEater.bcd Jeremy M. Goodell:OSBU South:Xerox 22-Jan-88 18:14:44
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>SpaceEater.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>SpaceEater.doc of 25-Jan-88 11:31:22
Description:
-- File: SpaceEater.doc - last edit:
-- Goodell.ES 25-Jan-88 11:31:22

Ever needed to use up all (or some of) the space on your ViewPoint volume for testing or just for fun? Well, now there's an easy, relatively quick way of doing so. Just run SpaceEater.bcd in ViewPoint. This registers a SpaceEater command in the Attention menu. Selecting this brings up a window which allows you to specify how many pages you want to use up (up to 10 pages less than the available pages). Bug Start and a file of that size is created and an icon is placed on your desktop. Delete the icon when you want the space back.

The file is created in the background, using the BWS background manager, but it doesn't matter much since the system doesn't allow you to do anything while a file of this size is being created.

Fine Notes:

Filing and Pilot apparently need some overhead space, so depending on the size of the file you create, 1...

---

28. SunTIPTest.bcd Alex Dianysian:OSBU South:xerox 17-Feb-89 11:34:26
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>SunTIPTest.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>SunTIPTest.doc of 17-Feb-89 11:39:39
Description:
-- File: SunTIPTest.doc - last edit:
-- Dionysian 17-Feb-89 11:39:39

SunTIPTest run's on 6085 on BWS 4.3. It compiles TIPTables for Sun4 BWS. It is useful for making sure the TIP files are syntactically correct.

Commands available:

Create - Calls SunTIP.CreateTable with the filename in the Name: field of the subwindow. If the parse is successful, nothing happens. If there is any problem, a message is posted in the Attention window, and an error log named "TIP.errors" is placed in the system folder.

---

29. SwatTool.bcd Michael D. Kupfer:OSBU North:Xerox 1-Oct-87 22:30:16
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>SwatTool.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>SwatTool.doc of 2-Oct-87 16:12:26
Description:
-- File: SwatTool.doc - last edit:
-- Kupfer 2-Oct-87 16:12:26

This hack registers an item in the Attention menu called "Swat Tool". Invoking this menu item brings up a small window shell containing 2 commands: Close and Swat. Close does the obvious thing. Swat raises an uncaught signal, sending you to wherever uncaught signals normally send you.

"OK," I hear you saying, "so what's the point? Have you been watching too many Mr. Tea commercials?" The point is that doing certain types of performance analysis becomes much less tedious. Suppose you want to find out why redisplaying a shared book takes so long. Well, you set up the Perf Tool, redisplay the shared book 10 times, and then (immediately) swat back to CoPilot to look at the Perf Tool numbers. Well, redisplaying a shared book takes long enough for you to get bored and inattentive, but not long enough that you can go do something else while y...

---

30. TableToFile2.bcd Larry Rosenberg:OSBU North:Xerox 25-Oct-88 15:29:14
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>TableToFile2.bcd
Documentation: none

---

31. VPActivity.bcd Seizo Umehara:OSBU South:Xerox 10-May-88 14:13:03
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>VPActivity.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>VPActivity.doc of 10-May-88 16:10:55
Description:
-- Copyright (C) 1988 by Xerox Corporation. All rights reserved.
-- VPActivity.doc : edited by STU, 88-May-10 4:10 pm PDT

VPActivity.bcd is a BWSActivity.bcd with following changes:

- o Smaller display window.
- o Default bars are CPU & Disk IO.
- o VM page was deleted.
- o Popup menu is triggered by mouse chords on window.
- o Window can be moved to any place by Point-Down & dragging.
- o Window place and number of bars are remembered until next Logon.
- o Automatic popup over the "Full screen" of Tajo in VP.

Note:
(1) It will be better to change suffix to ".autorun" before copy to the loader.
(3) This version of VPActivity runs on BWS 4.3 (VP2.0) only.

---

32. VPCHBrowser.bcd Matt Thompson:OSBU North:Xerox 21-Nov-88 0:42:05
Home: [Alt:OSBU North:Xerox]<BWSHacks>4.3>Tools>VPCHBrowser.bcd
Documentation: [Alt:OSBU North:Xerox]<<BWSHacks>4.3>Doc>>VPCHBrowser.doc of 17-Nov-88 13:01:12
Description:
-- File: VPCHBrowser.doc - created by MJT. Last edit:
-- MJT 17-Nov-88 9:39:20
***** needs to be edited!!!!!*******
-- Copyright (C) 1985, 1988 by Xerox Corporation. All rights reserved.
-- Borrowed heavily from CHBrowser.doc

VPCHBrowser is a Clearinghouse browser tool. The user can enumerate Clearinghouse entries of any given type (for example, of type Print Service). The user can also examine the property values for a particular entry (for example, the entry for Nevermore:OSBU North:Xerox).

Running VPCHBrowser.

Retrieve VPCHBrowser.bcd from the BWSHacks directory

Using VPCHBrowser.

Here is an explanation of the VPCHBrowser tool fields.

CH Entry Type:
The user specifies a specific kind of Clearinghouse entry to examine. The user can menu over CH Entry Type to select from some predefined entry types.

If the user wants to examine Clearinghouse entries for some entry type that is not predefined, then the user can type the n...

---

start
00903 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Getting started
To: NewUsers: ;

This module discusses how to "boot" a 6085 processor. (To boot a machine is to load and start a system on the machine.)

Before you start, you should make sure that you are reading the correct tutorial. There are currently two different machines that can run XDE software: the 8010 and the 6085. If you have a 6085, you should be reading this tutorial; if you have an 8010, you should be reading a separate tutorial, called Teach8010Booting.nsmail. If you don't know which kind of machine you have, you will have to ask someone.

You should also make sure that you have a printed copy of this tutorial before you continue, since the electronic copy will not always be available while you are practicing booting.

start
01647 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Physical and logical volumes
To: NewUsers: ;

In Xerox terminology, a hard disk ("physical volume") is divided into several "logical volumes." There can be up to 10 logical volumes on a physical volume, although there are not usually nearly so many. Obviously, the more logical volumes you have, the less space you have on each of those volumes.

There are three principle systems that you can have on your workstation: ViewPoint, XDE, and InterLisp. Most machines will have XDE and ViewPoint or XDE and InterLisp; unless you have a very large disk (80 Mbytes or more), you probably don't have room for all three.

To find out what logical volumes are on your disk, and how much space is on each of them, find the word Volume: at the right hand side of your Herald Window. This field gives the number of free pages on your Tajo volume.

Clicking Point over this field cycles through the logical volumes on the disk, and displays the number of free pages on each. Use Point to cycle through the volumes on your disk, so that you have some idea of the names of the volumes and how large they are. For machines running XDE and ViewPoint, the most common configuration is to have the following four volumes:

Tajo: XDE volume for development work
User: ViewPoint volume
Scavenger: ViewPoint data volume

For now, you don't need to know any more detail about any of these volumes. The next tutorial, TeachCompile-Bind-RunWithXXXX.nsmail discusses the different logical volumes in more detail.

start
01669 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Booting a logical volume
To: NewUsers: ;

Generally speaking, each logical volume has a bootfile; the bootfile is the file that gains control when you boot the volume. For example, there is a Tajo bootfile on the logical volume Tajo: when you boot the Tajo volume, this bootfile sets up the environment you know as XDE. The name of the bootfile or system that the bootfile creates does not have to correspond to the volume name; for example, the ViewPoint bootfile is generally stored on the User volume. It is also possible to use a logical volume just as a data volume, in which case you would not put a bootfile on it. You cannot boot a volume that does not have a logical volume bootfile on it.

Booting a volume will restore it to a pristine state, and will thus cure most software problems. Booting will not destroy any files or mail messages, but any text not saved in a file will be lost.

To boot a logical volume from XDE, you use the Boot From: menu, available from the Herald Window. Move your cursor into the Herald and chord to bring up the Boot From: menu. This menu contains the names of the volumes on your workstation, such as Scavenger, User and Tajo. This menu can be used to boot any of the volumes listed. For example, to reboot your Tajo volume, you would select Tajo from this list. Try it, and notice that the graphic mouse appears, asking for confirmation of the boot command. Confirm

the command with Point; this will reboot Tajo. (To abort the command, click Adjust instead of Point.)

start
01003 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Boot switches
To: NewUsers: ;

There are a number of different boot switches available to control certain aspects of booting. The person who set up your machine for you should have specified a set of default boot switches for each bootfile on the machine. Generally speaking, you need quite a bit of experience before you know which boot switches you should use; for now, you should always consult a more experienced user before changing boot switches.

You can also set boot switches for a particular boot by using the Set Switches: command in the Boot From: menu. For now, you shouldn't have to worry about boot switches; the person who set up your machine should have initialized the correct switches.

Consult your XDE User's Guide for a list of available switches and for a complete explanation of the other commands in the Boot From: menu.

start
03439 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Boot button
To: NewUsers: ;

Rebooting the appropriate logical volume will solve most software problems. However, there will be times when a software boot is not sufficient. For example, if your machine is completely "frozen," you will not be able to reboot from the Herald Window.

For those times when your machine freezes, you can try a special command, PROPS-STOP. This usually works when your cursor is an hour-glass, and one operation appears to be hung. It will not work when everything appears to have stopped (including your clock). This command was built into Tajo 12.3 bootfiles and later. If you are running CoPilot you will not have this feature.

When a PROPS-STOP does not work, you will need to do a "hard boot" of the physical volume instead. To do a hard boot, you will need to use the boot button on your machine. The boot button on a 6085 is a small red button labelled B Reset; it is near the floppy drive.

To do a hardware boot, push the boot button. The screen will go blank for a little while, and then display a series of icons across the bottom of your screen. These icons represent the different devices from which you can boot.
Going from left to right within the first group of icons, the icons represent a workstation, a floppy drive, and an Ethernet.

To do a particular boot, select the function key that corresponds to the icon on the screen. Thus, to perform a workstation boot, you press the F1 function key (the leftmost function key); to perform a floppy boot you press the F2 function key, and so on.

The second group of icons represent "diagnostic boots." Each of these icons is the same as the icon in the first group, with the addition of a wrench. We discuss these diagnostic boots later in the tutorial.

Select the F1 key now to do a workstation boot. This will reboot your workstation, using the microcode, germ, and bootfile currently installed on the workstation.

(Note: If you do not select an icon within 30 seconds of pressing the boot button, the machine will automatically boot by itself. The default boot is either a standard workstation boot (F1) or a diagnostic workstation boot (F5), depending on how your machine is set up. For now, we want you to try a workstation boot, so if you waited too long and let the default boot occur, then you should press the Reset button and try again.)

Doing a workstation boot should return you to the XDE volume. (It is possible to set up workstations so that a workstation boot returns control to a volume other than XDE, but for programmers it is rare. On most programmers' workstations, you will end up in XDE when you do a workstation reboot.)

During the boot, there is a series of numbers in the upper left hand corner of your screen. The numbers start at 100, and gradually cycle through until they reach 990, which is the normal readout for the XDE volume. 8000 is the normal readout for a ViewPoint volume. If something goes wrong during the boot, the numbers will stop at a number other than 990; the number indicates the type of error.

(Note: you will sometimes hear people call these numbers "maintenance panel codes," because they are displayed on the maintenance panel of the 8010.)

Booting does take a few minutes, so don't panic if a code remains for a little while.

start
01594 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Booting from the net
To: NewUsers: ;

The other possible devices that you can use to boot are a floppy and the net. Booting from the net is very common; booting from floppy is generally done only for initial installation. If you happen to have a bootable floppy around, you are welcome to try an F2 boot, but in general you just need to know that it is possible to boot from floppy.

Booting "from the net" means booting from software that resides on a Boot Service on the Ethernet. Boot Services typically offer utilities for performing operations such as configuring your workstation and installing software.

To boot from the net, you select the F3 option (the icon that shows an Ethernet.) This will produce a menu that lists the available bootfiles.

The system administrator for a given network sets up the bootfiles that are associated with a particular number, so the bootfiles you see listed on one net may be slightly different from the bootfiles you see on another net. In general, however, you will always have at least the Installer, which is a utiltity that simplifies intalling and upgrading your system, and a diagnostic Tajo bootfile. If you aren't sure what some of the choices on your menu are, ask someone nearby for clarification.

To practice booting from the net, try booting the Installer. Experiment with the Installer a bit if you like, and then reboot (any way you like) to reach your Tajo volume again.

start
00986 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Diagnostic boots
To: NewUsers: ;

Earlier, we mentioned the diagnostic boots, which are represented by a picture of a standard booting device (workstation, floppy, net), with the addition of a wrench. The wrench indicates a diagnostic boot; that is, a boot whose purpose is to test the hardware. In general, you will not have to do these boots. If you think there might be something wrong with your machine, however, you might want to try running diagnostics to get some indication of what is wrong. If the boot does not complete normally, the machine will "hang" with a particular code on the cursor; this number indicates the nature of the problem. (For a list of "maintenance panel" codes, see the XDE User's Guide.)

You can do a diagnostic boot from the workstation, from a floppy or from the net, just like standard boots.

start
04072 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: When not to boot: moving between booted volumes
To: NewUsers: ;

Although you can always use booting as a means to get from one volume to another, you don't have to boot every time you want to reach another volume. There is a shorthand way of calling the "debugger" volume which allows you to easily move from one previously booted volume to another. The advantage of using this method is that you will not have to reconstruct your screen each time that you reach a volume; you will be able to leave your desktop as is, much as you do when you invoke DMT.

The XDE volume that you are in is called a "debugger" volume; it can serve as a debugger for either other XDE volumes or the User (ViewPoint) volume. Depending on what kind of development work you will be doing, you may not have any other XDE volumes, since much of XDE application development can be done in the same world (with the Sword debugger). To do Viewpoint application development, though, you will need to establish the User volume as the "client volume" for the debugger.

You can establish a User-Tajo relationship by booting User from the Herald window to invoke the ViewPoint software. To return to Tajo once User has been booted, simply type SHIFT-STOP (hold down the

SHIFT and STOP keys simultaneously.) This is known as an "interrupt." As you can see, interrupting is not the same thing as booting; it does not affect the state of your Tajo volume. In particular, you do not have to reconstruct the layout or state of any of your tools. (Booting User takes about 15 minutes, so you might not want to experiment with it now.)

Once you have booted User for the first time and interrupted, you do not have to reboot to return to User. This is because the state in which you left your User volume was saved in an "Outload file". To reach your previously booted User volume from Tajo without rebooting, you will need use its outload file. To do this, create a debugger window in one of two methods. Look at the left side of the herald window.

1) If it says "Tajo of 12.3", then type "Interpreter" in the executive.
2) If it says "Tajo of 14.0", then type "Sword" in the executive.

Since Sword can be the debugger for many different clients, you will need to specify what client we wish to operate on. To do this, chord over the client: field and select "outload". This action should cause a new subwindow to be displayed (you may need to select "outload" twice). This subwindow prompts you for the name of the outload file that describes your User volume. The default name for an outload file is "Debuggee.outload", but this may vary depending on your user.cm entries. You should check with your mentor to find out how your machine was set up.

When the name of the outload file is determined, and entered in the appropriate field, hit Apply! This causes the subwindow to go away, and when the specified outfile is read the debugger will display what is called a "swap-reason" in the debugger window. It should read "*** Interrupt ***", since that is what you did.

To use this window to reach User again, you need to type "P" (for Proceed,) and follow with a carriage return to confirm the command. Hit the DELETE key to abort the command. (Note that you must type only P, and not more of the word.) You can Proceed into a volume once you have booted it and interrupted (SHIFT-STOP) out of it. Or you can execute the proceed command in the go: field of the form subwindow. The cursor will change to the shape of a mouse, waiting for you to confirm with "Point". ("Adjust" will abort the command.)

Proceed now to the user volume (with whichever method you like). This will return you to User; once you are in User, SHIFT-STOP back to XDE.

The above method can be used to move about between volumes during your normal operation. You do not normally need to boot the new volume each time that you want to go from one volume to another.

start
01005 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: For more information
To: NewUsers: ;

You should now feel comfortable with booting a volume from the herald window of a particular volume, and with booting your machine from boot buttons. You should also have a general idea of when you will need to boot a volume and when you can just call the debugger volume.

If you do not feel comfortable with booting, you should go back and reread the appropriate sections, and experiment some more. It is important for you to be comfortable with the various ways of booting your machine.

Hopefully, you now have enough information to enable you to survive the first few occasions on which you will need to boot, but remember that there is more information contained in the XDE User's Guide which you will later need to know.

When you are ready to go on, choose TeachMailSystem.nsmail from the Files: menu.

start
00904 00071 US
@00045 01536 ftffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Getting started
To: NewUsers: ;

This module discusses how to "boot" an 8010 processor. (To boot a machine is to load and start a system on the machine.)

Before you start, you should make sure that you are reading the correct tutorial. There are currently two different machines that can run XDE software: the 8010 and the 6085. If you have an 8010, you should be reading this tutorial; if you have a 6085, you should be reading a separate tutorial, called Teach6085Booting.nsmail. If you don't know which kind of machine you have, you will have to ask someone.

You should also make sure that you have a printed copy of this tutorial before you continue, since the electronic copy will not always be available while you are practicing booting.

start
01648 00071 US
@00045 01536 ftffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Physical and logical volumes
To: NewUsers: ;

In Xerox terminology, a hard disk ("physical volume") is divided into several "logical volumes." There can be up to 10 logical volumes on a physical volume, although there are not usually nearly so many. Obviously, the more logical volumes you have, the less space you have on each of those volumes.

There are three principle systems that you can have on your workstation: ViewPoint, XDE, and InterLisp. Most machines will have XDE and ViewPoint or XDE and InterLisp; unless you have a very large disk (80 Mbytes or more), you probably don't have room for all three.

To find out what logical volumes are on your disk, and how much space is on each of them, find the word Volume: at the right hand side of your Herald Window. This field gives the number of free pages on your Tajo volume.

Clicking Point over this field cycles through the logical volumes on the disk, and displays the number of free pages on each. Use Point to cycle through the volumes on your disk, so that you have some idea of the names of the volumes and how large they are. For machines running XDE and ViewPoint, the most common configuration is to have the following three volumes:

Tajo: XDE volume for development work
User: ViewPoint volume
Scavenger: ViewPoint data volume

For now, you don't need to know any more detail about any of these volumes. The next tutorial, TeachCompile-Bind-RunWithXXXX.nsmail discusses the different logical volumes in more detail.

start
01669 00071 US
@00045 01536 ftffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Booting a logical volume
To: NewUsers: ;

Generally speaking, each logical volume has a bootfile; the bootfile is the file that gains control when you boot the volume. For example, there is a Tajo bootfile on the logical volume Tajo: when you boot the Tajo volume, this bootfile sets up the environment you know as XDE. The name of the bootfile or system that the bootfile creates does not have to correspond to the volume name; for example, the ViewPoint bootfile is generally stored on the User volume. It is also possible to use a logical volume just as a data volume, in which case you would not put a bootfile on it. You cannot boot a volume that does not have a logical volume bootfile on it.

Booting a volume will restore it to a pristine state, and will thus cure most software problems. Booting will not destroy any files or mail messages, but any text not saved in a file will be lost.

To boot a logical volume from XDE, you use the Boot From: menu, available from the Herald Window. Move your cursor into the Herald and chord to bring up the Boot From: menu. This menu contains the names of the volumes on your workstation, such as Scavenger, User and Tajo. This menu can be used to boot any of the volumes listed. For example, to reboot your Tajo volume, you would select Tajo from this list. Try it, and notice that the graphic mouse appears, asking for confirmation of the boot command. Confirm

the command with Point; this will reboot Tajo. (To abort the command, click Adjust instead of Point.)

start
01003 00071 US
@00045 01536 ftffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Boot switches
To: NewUsers: ;

There are a number of different boot switches available to control certain aspects of booting. The person who set up your machine for you should have specified a set of default boot switches for each bootfile on the machine. Generally speaking, you need quite a bit of experience before you know which boot switches you should use; for now, you should always consult a more experienced user before changing boot switches.

You can also set boot switches for a particular boot by using the Set Switches: command in the Boot From: menu. For now, you shouldn't have to worry about boot switches; the person who set up your machine should have initialized the correct switches.

Consult your XDE User's Guide for a list of available switches and for a complete explanation of the other commands in the Boot From: menu.

start
03181 00071 US
@00045 01536 ftffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Boot buttons
To: NewUsers: ;

Rebooting the appropriate logical volume will solve most software problems. However, there will be times when a software boot is not sufficient. For example, if your machine is completely "frozen," you will not be able to reboot from the Herald Window.

For those times when your machine freezes, you can try a special command, PROPS-STOP. This usually works when your cursor is an hour-glass, and one operation appears to be hung. It will not work when everything appears to have stopped (including your clock). This command was built into Tajo 12.3 bootfiles and later. If you are running CoPilot you will not have this feature.

When a PROPS-STOP does not work, you will need to do a "hard boot" of the physical volume instead. To do a hard boot, you will need to use boot buttons. The boot buttons on an 8010 are the two buttons beside the maintenance panel on your machine. Locate them. (Note: there is a small door (more like a flap) that covers the maintenance panel. If you do not see the boot buttons, then your door is closed; you will have to find this door and pull it down.) The left button is labelled "B RESET" and the right button is labelled "ALT B."

To do a hardware boot, press in both buttons, and then release the left one. If you continue to hold in the right button, the maintenance panel numbers should cycle gradually from 0000 to 0010. Each of these numbers represents a different kind of boot. The mapping from number to type of boot varies occasionally from release to release and location to location, so you should check with someone local to find out the mapping. The various boots are referred to as 0 boot, 1 boot, and the like.

When the maintenance panel codes reach the number of the boot that you want, you release the right button, and the boot will be performed. If you make a mistake and pass the number that you want, don't worry about it; the numbers will continue to cycle and you will get another chance.

A 0 boot is the basic boot. It will run hardware diagnostics to make sure your disk is all right, and then it will boot a logical volume using the microcode, germ, and bootfile currently installed on the workstation. Depending on how your machine is set up, this should return you to the XDE volume. (It is possible to set up workstations so that a hard boot returns control to a volume other than XDE, but for programmers it is rare. On most programmers' workstations, you will end up in XDE when you do a hard boot.)

The numbers that display on the maintenance panel during the boot signify that diagnostics are being executed. If there is a difficulty with the boot, these numbers will stop at a number that indicates the nature of the problem. If this should happen during a boot, you will need help.

990 is the normal readout for XDE volumes; 8000 is the normal readout for ViewPoint. Any other number usually indicates that something is wrong. (Booting does take a few minutes, so don't panic if a code remains for a little while.)

start
00840 00071 US
@00045 01536 ftffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)

From: XDE-Training:OSBU North:Xerox
Subject: Alternative boots
To: NewUsers: ;

As mentioned above, each of the numbers from 0 to 9 represents a different kind of boot. These are called "alternate" boots, since the 0 boot is considered the standard boot.

The most common alternate boot is the 1 boot, which is just like the 0 boot except that it does not run hardware diagnostics. Thus, you can use this boot when you want to save time by not running the diagnostics. However, you should not always use a 1 boot; the diagnostics performed in the standard boot are a valuable method of checking to make sure that your machine is healthy. For now, since there is nothing wrong with your machine (hopefully), try doing a 1 boot.

start
00808 00071 US
@00045 01536 ftfffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Another Alternative: Booting from the net
To: NewUsers: ;

Another important alternate boot is booting the Installer "from the net". Booting "from the net" means booting from software that resides on a Boot Service on the Ethernet. Boot Services typically offer utilities for performing operations such as configuring your workstation, installing and upgrading software, diagnostics, etc. The Installer is the utility that is used to install/upgrade software.

Find out from someone how to reach the Installer from the net, and then try booting it.

Experiment with the Installer a bit if you like, and then reboot (try a 1 boot) to reach Tajo again.

start
04072 00071 US
@00045 01536 ftfffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: When not to boot: moving between booted volumes
To: NewUsers: ;

Although you can always use booting as a means to get from one volume to another, you don't have to boot every time you want to reach another volume. There is a shorthand way of calling the "debugger" volume which allows you to easily move from one previously booted volume to another. The advantage of using this method is that you will not have to reconstruct your screen each time that you reach a volume; you will be able to leave your desktop as is, much as you do when you invoke DMT.

The XDE volume that you are in is called a "debugger" volume; it can serve as a debugger for either other XDE volumes or the User (ViewPoint) volume. Depending on what kind of development work you will be doing, you may not have any other XDE volumes, since much of XDE application development can be done in the same world (with the Sword debugger). To do Viewpoint application development, though, you will need to establish the User volume as the "client volume" for the debugger.

You can establish a User-Tajo relationship by booting User from the Herald window to invoke the ViewPoint software. To return to Tajo once User has been booted, simply type SHIFT-STOP (hold down the SHIFT and STOP keys simultaneously.) This is known as an "interrupt." As you can see, interrupting is not the same thing as booting; it does not affect the state of your Tajo volume. In particular, you do not have to reconstruct the layout or state of any of your tools. (Booting User takes about 15 minutes, so you might not want to experiment with it now.)

Once you have booted User for the first time and interrupted, you do not have to reboot to return to User. This is because the state in which you left your User volume was saved in an "Outload file". To reach your previously booted User volume from Tajo without rebooting, you will need use its outload file. To do this, create a debugger window in one of two methods. Look at the left side of the herald window.

1) If it says "Tajo of 12.3", then type "Interpreter" in the executive.
2) If it says "Tajo of 14.0", then type "Sword" in the executive.

Since Sword can be the debugger for many different clients, you will need to specify what client we wish to operate on. To do this, chord over the client: field and select "outload". This action should cause a new subwindow to be displayed (you may need to select "outload" twice). This subwindow prompts you for the name of the outload file that describes your User volume. The default name for an outload file is "Debuggee.outload", but this may vary depending on your user.cm entries. You should check with your mentor to find out how your machine was set up.

When the name of the outfile is determined, and entered in the appropriate field, hit Apply! This causes the subwindow to go away, and when the specified outload file is read the debugger will display what is called a "swap-reason" in the debugger window. It should read "*** Interrupt ***", since that

is what you did.

To use this window to reach User again, you need to type "P" (for Proceed,) and follow with a carriage return to confirm the command. Hit the DELETE key to abort the command. (Note that you must type only P, and not more of the word.) You can Proceed into a volume once you have booted it and interrupted (SHIFT-STOP) out of it. Or you can execute the proceed command in the go: field of the form subwindow. The cursor will change to the shape of a mouse, waiting for you to confirm with "Point". ("Adjust" will abort the command.)

Proceed now to the user volume (with whichever method you like). This will return you to User; once you are in User, SHIFT-STOP back to XDE.

The above method can be used to move about between volumes during your normal operation. You do not normally need to boot the new volume each time that you want to go from one volume to another.

start
01006 00071 US
@00045 01536 ftffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: For more information
To: NewUsers: ;

You should now feel comfortable with booting a volume from the herald window of a particular volume, and with booting your machine from boot buttons. You should also have a general idea of when you will need to boot a volume and when you can just call the debugger volume.

If you do not feel comfortable with booting, you should go back and reread the appropriate sections, and experiment some more. It is important for you to be comfortable with the various ways of booting your machine.

Hopefully, you now have enough information to enable you to survive the first few occasions on which you will need to boot, but remember that there is more information contained in the XDE User's Guide which you will later need to know.

When you are ready to go on, choose TeachMailSystem.nsmail from the Files: menu.

start
01162 00071 US
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Click at 1, then Display!
To: NewUsers: ;

Welcome to the Xerox Development Environment! You are reading the first message of a tutorial that will give you "hands-on" experience with the XDE user interface.

The first thing that you will need to become friends with is your mouse, which is used to direct attention to a particular area of the screen. The standard mouse has two buttons. The left button is called "Point"; the right button is "Adjust." Pressing and immediately releasing a mouse button is called "clicking" the button. Pressing down both buttons simultaneously is called "Chording" the mouse.

Mouse movements are tracked on the screen by a small black arrow called the "cursor". Try using your mouse to point at various places on the screen. You can move the mouse in any direction, or pick it up and move it when you reach the edge of the mouse pad. When you are good at using the mouse, move the cursor over to the word Display! (which you used to view this message) and click Point.

start
01316 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Commands
To: NewUsers: ;

As you can see, clicking Point over the word Display! allows you to read a new message. Display! is a command associated with the MailTool, which is the name of the tool that you are using to read these messages. The XDE user interface is largely based on visual imagery: you can usually see the commands associated with a given tool without having to ask for them or memorize them. A word followed by an exclamation point, such as Display!, Delete!, or Undelete!, indicates that the word is a command. (This is true throughout the environment; it is not unique to the MailTool.)

Clicking Point over a command invokes that command. When you invoke a command, it is best to set just the tip of the cursor over the command; placing the entire cursor on top of a command may not activate it.

Also, if you move away from a command while the button is still down, the command will not be invoked when you release the button. Try this: press and hold in Point over Display!, and then move the cursor away from the command before releasing Point. The command will not be invoked.

Now invoke Display! correctly to view the next message.

start
01497 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Windows
To: NewUsers: ;

You are reading this message in what is called the "text subwindow" of your mail "window". Each XDE "tool", or applications program, communicates with you through one or more windows. A window is just a partition of the screen in which text or graphics can be displayed. Generally, each tool owns a window, although a tool does not have to have any window or it can have several windows. At the top of each window is a herald, or name stripe, that tells you the name of the tool that the window is associated with.

Windows are composed of subwindows, which are separated by horizontal black lines. The MailTool window has four subwindows. The uppermost subwindow is a message subwindow used for posting messages from the tool to the user. The second subwindow, which contains an ordered list of all the messages available to you, is called the Table of Contents.

The third subwindow is a command subwindow. All the available commands specific to the MailTool are grouped together in this command subwindow. The items followed by ! are all commands; the items followed by : are called fields, and are used for collecting arguments to commands. Fields are discussed more fully later in this tutorial.

The fourth subwindow is a text subwindow, which is used for displaying the messages.
Display the next message.

start

01578 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Displaying your messages
To: NewUsers: ;

Display! allows you to read through a group of messages in the order in which they are listed in the Table of Contents. Thus, you do not have to explicitly specify an argument for Display!; the default argument is the next message in the mail file.

However, if you would like to read your messages in some other order (for example, if you want to read an earlier message again), you can explicitly specify any message in the file as the argument to Display!. To do this, just click Point over any character in the message title (in the Table of Contents subwindow). (Notice that the character becomes highlighted; that is, black characters become white or vice-versa.) When you have explicitly selected a message, invoking Display! will cause that message to be displayed in the bottom subwindow, regardless of whether or not you have read the messages that precede it.

Notice that there is an asterisk in front of the Table of Contents entry for each unread message. When you read your mail in order, the title of the message that you are currently reading is moved to the top of your Table of Contents. There is also a small arrowhead in the Table of Contents window that points to the title of the message currently being displayed. Find the title of this message in the Table of Contents.

Try explicitly selecting the title of the next message, and then invoke Display!

start
00817 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Marking messages
To: NewUsers: ;

You can mark specific messages in the Table of Contents if you like. Try selecting the blank space at the far left of a Table of Contents entry (there are three spaces to the left of the number; select the leftmost). Once you have selected the space, type any letter, and it will appear there. Thus, for example, you can type an ! if you have a message that you are particularly interested in, or an I by messages that you thought were incomplete, or any other marking system you like. Markings may consist of only one letter.

To remove a marking, just select the letter and type a space (the delete key does not work here).

start
01563 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Your scrollbar
To: NewUsers: ;

When you view text in a window, you may actually be looking at only a small portion of the available information. A window can be thought of as an opening through which you can view a potentially infinite scroll of text; the amount of text that you see at one time is limited by the size of your window, and not by the amount of material in the text. To view text not currently visible, you can use a window feature called the scrollbar to "scroll" the text in a window up or down.

The scrollbar for a window is a narrow transparent rectangle found at the far left of the window. Move your cursor into the scrollbar for the Table of Contents subwindow of the MailTool window. (Each subwindow is scrolled separately.) When the cursor is in the correct position for scrolling, it will change into a double-headed arrow. Try moving your cursor into the scrollbar.

When the cursor is in the scrollbar, notice that the scrollbar has a dark grey region and a light grey region. These grey regions, combined, represent the entire length of the scroll behind the window; the dark grey region represents the percentage currently visible through the window. Thus, a dark small grey region means that you are viewing a small part of the file, and a large dark grey region means that you are viewing most of the file.

Display the next message to find out how to use your scrollbar.

start
02112 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox

Subject: Scrolling
To: NewUsers: ;

Mouse buttons direct the scrolling operation. The cursor changes when one of the buttons is pressed in the scrollbar: Point scrolls the document up (the double-headed arrow changes to point up) and Adjust scrolls it down (the double-headed arrow changes to point down). You can think of scrolling as moving the file that is behind the window so that you can see a different part of the file. The window itself remains the same; you are effectively just putting a different part of the file "in" the window.

Now try scrolling: position your cursor in the scrollbar for the Table of Contents beside message 10 and click Point. Notice that this line is now at the top of the window. Depending on how your machine is set up, you may be able to obtain continuous scrolling by holding down Point in the scrollbar region (rather than just clicking it.)

You can use Adjust to scroll back down in the file to view previous text. To practice this, put your cursor in the scrollbar somewhere in the middle of the Table of Contents and click Adjust. Message 10, or the message that was at the top of your Table of Contents before you clicked Adjust, will now be located at the position of the cursor. Practice scrolling the Table of Contents up and down.

When you are experimenting with scrolling, notice that the position of the dark grey region changes as you scroll the window up and down. This signifies that you are viewing a different part of the file; the dark grey region always shows the portion and position of the text that you are viewing with respect to the entire text.

You may have to scroll many of the later messages in this file in order to read the end of the message; if you are ever not sure whether or not there is more of a message, you should scroll up to check.

Since you are now familiar with the Display! command, you will no longer be reminded to invoke it when you want to advance to the next message.

start
01382 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Thumbing
To: NewUsers: ;

The scrollbar can also be used to "thumb" a file. Thumbing is analogous to opening a book by placing your thumb at the approximate position of the section you want to start reading, and pulling the book open at that point (as you might do with a dictionary).

To thumb a file, press Chord (both buttons simultaneously) while the cursor is in the scrollbar. The cursor should change to a left-pointing arrow that can be moved up or down in the scollbar. If the cursor is in the middle of the scrollbar, releasing Chord will move you to the middle of the file; if the cursor is at the top end of the scrollbar, releasing Chord will scroll you back to the beginning of the file. With practice, you will be able to reach the particular part of the file that you want to look at without having to scroll through the entire file. This is especially useful with large files.

Note: to release a Chord, always release the left button first. You don't have to release the buttons simultaneously, and you won't get the result you expect if you release the right button first.

Practice thumbing the Table of Contents. Notice that releasing Chord when the cursor is not in the scrollbar aborts the operation.

start
01089 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Adjusting subwindows
To: NewUsers: ;

In addition to scrolling or thumbing the text in a file, you may also wish to change the shape, size or position of the windows on your screen. The next series of messages will discuss the available commands for manipulating your windows.

Look at the right edge of the lines that divide this window into subwindows, and note the small boxes. These boxes are used to guide the dividing lines when you move them up or down within the window. To see how they work, choose one of the boundary lines on your screen and place the point of the cursor on the box that lies on it, then press and hold down Point. Move the cursor up and down, and note how the boundary moves with the cursor. Now move the line to the position you wish the boundary to be in and release the button. The line will be moved to that point. This feature is available for every subwindow in the Xerox Development Environment.

start

01821 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Window Manager menu
To: NewUsers: ;

In addition to adjusting subwindow divisions, you may also wish to make an entire window larger or smaller, or move it to a different location altogether. The Xerox Development Environment has a feature called the Window Manager menu, available for every window, which is used to control the size and position of your windows.

During the exercises on window manipulation, you should practice on your Empty Window, and not your MailTool window.

You can obtain the stack of possible "menus" for any window by placing the cursor anywhere in that window and pressing Chord. Try this now in the Empty Window. Continue to hold down Chord. You should see a stack of menus, with one menu fully displayed at the top of the stack. The title of the menu that is at the top of the stack should be highlighted; that is, it should appear as white letters on a black background instead of black letters on a white background. Menus are a convenient way of letting you see the available commands when you want to use them, while still conserving "real estate" on the screen the rest of the time. (Note, however, that commands in menus are not designated with !.)

If the Window Manager menu ("Window Mgr") is not already at the top of the stack of menus, you will have to bring it to the top. To do this, point the cursor at "Window Mgr" (it will highlight), continue to hold down Adjust while you release Point and then click it again. The Window Manager menu should now be fully displayed at the top of the stack. Bringing a menu to the top of the stack may be a bit difficult at first; you should practice until you are good at it.

start
01409 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Moving a window
To: NewUsers: ;

The Move command is used to move a window about the display screen without changing its size.

Bring the Window Manager menu to the top of the stack, continue to hold down Chord, and select "Move" from the menu by moving the cursor on top of it. ("Move" will be highlighted when the cursor is in the correct position.) Release Chord (both buttons at once), and the cursor will change into the shape of a corner with an "M" in it.

Practice moving the cursor in and out of the various boundaries of the Empty Window, and watch the corner change shape to represent different corners of the window. Using Move is like picking up a piece of paper by one corner. Thus, the corner represented by the cursor is basically the argument to the Move command; it specifies the corner by which you are "picking up" the window.

When you are ready to move the window, move the cursor to the desired position, and click Point. The corner represented by the cursor will be moved to that location.

Move allows you to move a window about the display area, but doesn't let you change its size or shape. Try moving the Empty Window around, using different corners as your anchor. Experiment until you are comfortable with this command.

start
01185 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Changing the size of a window: Drag
To: NewUsers: ;

The Drag and Grow commands allow you to change the size of a window on the screen. Bring up the Window Manager menu by chording just as you did to invoke the Move command, but this time select "Drag" instead. The cursor will change to look like an arrow that points to a line. The line represents a border of the window. The Drag command moves (drags) one border of the window either outward (to make the window bigger) or inward (to make the window smaller).

Move the cursor in and out of the window and notice how the cursor changes to represent the different borders of the window. When you click Point, the specified border of the window will stretch or shrink to the position of the cursor, and the rest of the window will remain the same. Practice moving around the borders of the Empty Window with the Drag command, but try to make sure that the Empty Window does not overlap this message. Notice that you can use Drag to shrink the window as well as to enlarge it.

start
00843 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Changing the size of a window: Grow
To: NewUsers: ;

Drag allows you to adjust the position of one window border at a time. The Grow command, on the other hand, allows you to adjust length and height simultaneously. Select Grow in the Window Manager menu. The cursor should look like a corner with a "G" in it. Move this corner in and out of the window and watch it change to the shape of the corner closest to where you exit the window, as it did with the Move command. Position the cursor and click Point.

Grow allows you to pull a corner of the window in any direction, enlarging or shrinking the window along its width and height. Experiment with this command in the Empty Window.

start
01365 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Top and Bottom
To: NewUsers: ;

In the XDE, you can overlap and stack windows, just as you can stack pieces of paper on a desk. If one window either partially or completely covers another window, you may wish to change the order of the stack (much as you would shuffle the stack of papers). The "Top" and "Bottom" commands in the Window Manager menu are used to control the position of a window in the stack: Top places a window on top of all others; Bottom places it beneath them.

To practice these commands, you must first have some overlapping windows. Move your windows around until you have a group of windows in a stack. (The Herald window, which is the wide rectangle at the top of your screen can be moved just like any other window. You can use it in your stack of windows if you like.)

Now try the Top and Bottom commands. Invoking these commands occasionally causes one or more windows to be completely obscured, and it is not uncommon to forget a window that is invisible. You may therefore wish to manipulate your windows so that a tiny portion of each window is showing. You can invoke the menu for any window as long as the visible portion of the window is at least the size of the cursor.

start
01594 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Size and Zoom
To: NewUsers: ;;

The Size command on the Window Manager menu reduces a window to a "tiny" rectangle of fixed size. Tiny windows can appear anywhere on the screen. (You will learn how to control the position of a tiny window in a later message.)

Try invoking Size on the Empty Window, and notice that the name of the window remains visible. (Depending on how your machine is set up, the tiny window will probably appear somewhere near the top of your screen; you may have to use Top or Bottom to find the tiny window.)

When a window is tiny, you can call up a menu in the same way as when it is normal size; however, Move, Grow, and Drag do not work for tiny windows. Since the size of a tiny window is fixed, there is no way to use Drag or Grow; we discuss how to move a tiny window in a later lesson.

Invoking Size on a tiny window puts it back to its original size and position, and places it on top of any other window at that location. Invoke Size again on the Empty Window.

As its name implies, the Zoom command causes a window to increase in size dramatically, so that it takes up all the available room on the screen. Like Size, Zoom can be reversed by invoking it a second time. Invoke Zoom twice on the Empty Window and watch it zoom up and then back down. A "zoomed" window is just like any other; you can use the Window Manager commands to put it under other windows, or to change its size if you like.

start
01828 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Deactivating a window

To: NewUsers: ;

The last command on the Window Manager menu is "Deactivate," which removes a window from the display and makes it inactive. Windows can be in one of two basic states: active or inactive. Active windows are those that are open on your screen and that you are currently able to read. An active window may be tiny; tiny windows are essentially active windows that have been temporarily moved out of the way. The contents of a window are not altered when it is reduced to the tiny state.

Deactivation, on the other hand, causes a window to lose any contents that you have typed into it. Deactivating a window destroys the tool window, but the tool itself is still available for future use.

Thus, deactivating a window only affects the information that you have typed into that window; it does not affect information that is stored in files. For example, if you deactivate an Empty Window, you will lose any text that you have typed into that window. However, if you deactivate the MailTool window, you will not lose any text; the table of contents and the associated messages are stored in files and are not affected by deactivation.

Deactivate the Empty Window. When a window is deactivated, the name of the associated tool is added to the Inactive menu. To bring up this menu, Chord in the grey bit area (any area not covered by a window) and select the Inactive menu from the list of available menus. This menu contains the names of all the tools that have been deactivated since the last time that you booted. Select Empty Window from the list of inactive tools, and the Empty Window will once again be active on your screen.

start
02306 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Accelerators for Move, Grow and Drag
To: NewUsers: ;

Up until now, you have been using the commands on the Window Manager menu to manipulate your windows. The next few messages introduce faster ways to invoke these commands. In the XDE, such shortcuts are called "accelerators."

Try moving the cursor to the herald of the Empty Window (the white-on-black "label" at the top of the window). As the cursor enters the herald, it changes to a bulls-eye shape, and sections of the herald video-invert. Move the cursor from the left side of the herald to the right side, and notice that the herald is divided into three sections.
The left and right sections, which are equivalent, offer quick ways to invoke the Move, Grow, Drag, Top, and Bottom commands. Position the cursor in an outer section of the herald of the Empty Window. Press and hold down Adjust, and the cursor will assume the "M in a corner" shape. When you release Adjust, the window will be moved to the place where the cursor is. In other words, to apply the Move command to a window, you can either:

(i) Get the Window Manager menu and select Move,
Position the cursor and click Point;

OR

(ii) Move the cursor into an outside section of the herald
Press and hold down Adjust
Position the cursor and release.

The Grow and Drag commands can be executed in a similar accelerated manner. Move the cursor back into an outside section of the herald, and hold down Adjust. Continue to hold it down and click Point; the cursor will change into the "G" for the Grow command. Still holding down Adjust, click Point again, and the cursor will change into the shape of an arrow pointing at a border, ready for you to execute the Drag command. Click Point a few more times to cycle through these three commands, then practice using these accelerators to adjust your windows.

(Note: if you select Move, Grow, or Drag from the Window Manager menu, you can hold down Adjust and click Point to cycle through to reach any of the others. Thus, if you accidentally select the wrong command from the menu, you can easily enough reach another command.)

start
00779 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Accelerators for Top and Bottom
To: NewUsers: ;

Move the cursor back into the left or right section of the herald of the Empty Window. If a window is already on top of all other windows on the screen, clicking Point in an outer section of its herald

will invoke the Bottom command. If the window is underneath any other window, clicking Point will invoke the Top command. Each time that you click Point, you will invoke the inverse of whichever command you invoked last time. Try positioning the Empty Window so that it overlaps this window slightly, then practice the accelerated Top and Bottom commands on both windows.

start
02066 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Accelerators for Size and Zoom
To: NewUsers: ;

Position the cursor in the center section of the Empty Window's herald. Click Adjust. You have just invoked the Size command. Clicking Point will invoke the Zoom command. Experiment a little with the accelerated Zoom and Size commands.

Invoke the Size command to shrink the Empty Window down to a tiny rectangle. Try out the accelerated commands on this small window. You will see that all of the commands function as they do on a full-sized window, with the exception of Move, Grow, and Drag. The Grow and Drag commands cannot be used to change the shape of a tiny window; the size of such a small window is fixed. The Move command works a little differently than it does on a full-sized window. Try it and see how it is different.

In a tiny window, the Move command can only be invoked with the accelerator (clicking Adjust in the right or left herald), and not with the Window Manager menu. Furthermore, no "M in a corner" appears when the command is invoked; instead, the cursor is tracked by the tiny window itself. Practice using the Move command to move the tiny Empty Window around.

Notice that the Herald Window, which is the long banner at the top of your screen, does not have a window herald when it is in the active state. You can still use the accelerated commands on this window, but you will have to move it from its position at the top of the screen. That is, the top edge of the window serves as its window herald, but you can't access that edge when the window is at the very top of your screen.

Each time you make a window tiny, it will return to the position from which it was sized. Thus, you can arrange the tiny windows on your screen any way that you like; the organization will not be lost unless you reboot your volume. (In a later tutorial, you will learn how to specify the way that your windows are set up after you boot.)

start
00744 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Summary of accelerators
To: NewUsers: ;

Here is a summary of the window manager accelerators available through the herald of a window:

The left and right sections of the herald are equivalent. Clicking Point in either of these two sections invokes Top and Bottom; holding down Adjust and clicking Point makes Move, Grow, and Drag available.

The center section of the herald is used to invoke Size and Zoom; Point invokes Zoom and Adjust Sizes the window. If you have trouble remembering these accelerators, you might want to write them down until you become more familiar with them.

start
01620 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Entering text
To: NewUsers: ;

Now that you know how to manipulate windows on your screen, you are ready to find out how to enter text in a window, how to edit existing text, and how to control the font in which characters on your screen are displayed.

Click Point anywhere in your Empty Window, and notice that a blinking caret appears. This is your type-in point.

When you type, the text will appear at the type-in point. In the XDE, only one window can have an active type-in point at any given time. Thus, when you want to enter text in a window, you need to first click a mouse button over that window to activate its type-in point.

Set your type-in point in the Empty Window and start typing. Type several words, and then hit the backspace key (a backward arrow, above the carriage return). The last letter in your text will be deleted. SHIFT and backspace together will delete the last word that you typed. Try it. These methods of editing are helpful if you realize that you have made a typing mistake while you are still typing. Now continue typing to the end of the line.

When you reach the end of the line, do not enter a carriage return. Continue typing, and notice that the system automatically breaks your text at the edge of a window and sends the overflow to the next line. Type several lines of text. Now change the shape of your window using the Grow or Drag command; the text will be reformatted to conform to the new shape of the window.

start
01636 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Selecting text
To: NewUsers: ;

Pick any letter in the text you have just typed and point the cursor at it. (Remember to use just the tip of the cursor.) Click Point once, and the character should video-invert; you have "selected" that letter. Now choose another letter and click Point twice in rapid succession to select an entire word. Three clicks will select the whole paragraph, four clicks the complete textual entity, and five clicks will return you to a single character. Try cycling through this sequence. Notice that the selected material is always video-inverted.

Another method of selecting text is extension with Adjust. Select the first character of this sentence with Point, then select the last character with Adjust; you will have selected the entire sentence. With that sentence selected, move your cursor to the first sentence in this paragraph and click Adjust over a letter in the word "text". The selected text will be extended backward to the new position. Thus, you can extend a selection either forward or backward using the extension technique.

Extension of selected text operates in the same units as the original selection: if you select a character with Point, the extension will be by characters; if you initially select a word, the extension will be by words, and so on. Practice selecting text until you are comfortable with both methods; the next few lessons will show you how you can manipulate selections of text within and across windows.

start
01704 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Adding text
To: NewUsers: ;

You can easily add text to any position in an existing editable document. For example, to add text to your Empty window, position the cursor in front of the character where you would like to insert your additions. Click Point to set your type-in point. Now start typing; the text will be inserted at the type-in point. Experiment with adding to the text in your Empty window, and notice that you can insert text at any position in a file, including the middle of an existing word. (Note that you always have to set a type-in point before you can enter text in a file; this is how you specify the location in the file where you want to put the new text.)

To set a type-in point at the beginning or end of a word, you may find it easiest to select the entire word with two clicks rather than trying to select the space that separates two words. Select a word by clicking twice at any letter in the first half of the word, and your type-in point will be at the beginning of that word. Select it by clicking near the end of the word, and your insertion point will be at the end of the word. Try positioning your type-in point using this method.

Note, however, that there are some windows that won't allow you to insert text into them. These windows are "read-only" windows, because you can read what they say, but you can't change it. For example, make a selection in your Table of Contents subwindow and try to type in some new text. The screen will blink at you to tell you that you are trying to do something illegal.

start
02474 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Moving and Copying text
To: NewUsers: ;

The next few messages discuss how to copy or move text from one position to another within a window or across windows using the special keys DELETE, COPY, STUFF, PASTE, and MOVE. These keys are all located in the cluster on the left side of your keyboard.

If you find that your keyboard does not have some of the keys described in this tutorial, refer to the one page keyboard summary document. This shows the mapping between the keycap names and functions; you will often have to refer to this document to find out which key or combination of keys performs a certain function. This document is commonly kept beneath the mouse pad for easy reference. Find your copy of it, and familiarize yourself with it. If you do not have one, you can refer to page I-18 in the General Tools chapter of the XDE User's Guide.

Set a type-in point anywhere in your Empty window and type in the sentences "Copying text is very easy. Moving text is also easy." Now suppose that you would like to move the word "very" to the second sentence. To do this, you first set a type-in point just before the "e" that starts the last word. (Click twice at the letter "e" to select the word and position the type-in in front of the word.) This is the new location to which you are going to move the text. Now press and hold down the MOVE key. While holding this key down, select the word "very". Release the MOVE key.

Your sentences probably now read "Copying text is easy. Moving text is also veryeasy." When you move or copy text, it is important that you are aware of whether or not you are moving the spaces that separate words as well as the letters themselves. Selecting a word with multiple clicking does not also select the spaces that precede and follow it. However, you can use the extension technique to extend a selection to include the surrounding spaces.

Now try copying text from one place to another. This command works in the same way as the MOVE command, except that it preserves the original text in addition to copying it to a new location. To copy text, first set a type-in point where you would like the new text to appear. Then hold down COPY, select the text to be copied, and release COPY. Try COPYing some text, and try to make sure that the spacing comes out right.

start
01433 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Stuffing text
To: NewUsers: ;

You can also use the STUFF key to copy text from one window into another or within a single window. To use STUFF, first make sure that you have a type-in point set in your Empty window. Now select some text from this message, move your cursor back into the Empty window and click Adjust. Clicking Adjust over a window resets the type-in point to the place where it was last set in that window, but does not change the current selection. Had you tried to use Point instead of Adjust to re-establish the type-in point in the Empty window, the current selection would no longer be the sentence above; it would be a character in your empty window. Now press STUFF and the text will be copied to the new location.

Note that if you are trying to use STUFF to copy text within a single window, you will have to use PROPS-Point instead of Adjust. (If you try to use Adjust, you will just extend the selection.) Try stuffing text within a window: select some text in your Empty window, set another type-in point in the same window by pressing the PROPS key and clicking Point, and then press the STUFF key. (Remember that your PROPS key probably has a different keycap name.)

Experiment with MOVE, COPY and STUFF until you are comfortable with them.

start
02124 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Deleting text
To: NewUsers: ;

To delete text from a window, select the text and press the DELETE key. As with moving or copying text, you will need to pay attention to whether you also select the spaces and the punctuation. For example, when deleting a sentence, you will want to also delete the spaces either preceding it or following it to avoid having extra spaces separating the remaining sentences.

Text that is deleted from the screen is not immediately destroyed. Instead, it is held in storage (called the "trash bin") until other text is deleted. One advantage of this feature is that you always have a chance to change your mind about the last section of text that you have deleted. The trash bin can hold an vast amount of text, but it is reset each time that you press the DELETE key. For example, if you have DELETEd an entire file, the contents of the file will all be stored in the trash bin. However, if you then DELETE an extra space, your existing trash bin (the contents of the file) will be removed and replaced with only a space character. Text deleted with either BS or BW is not inserted in the trash bin.

If you decide that you would like to re-insert the text that is stored in the trash bin, press PASTE and the text will be inserted at your type-in point. (This process is sometimes called "cutting and pasting".)

DELETE a piece of text, type a few words if you wish, then press and release the PASTE key. You will have moved the previously deleted text to a new place. (Note that deleting text automatically sets the type-in point, so you can just insert new text immediately after deleting old text.) Set a type-in point in another place and press PASTE again. The text will be pasted at that point as well.

Remember that you can perform a PASTE command on the text in the trash bin as many times as you like, but that the bin only holds one segment of deleted text at a time, and any previously deleted text is really gone.

start
02724 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Local files
To: NewUsers: ;

When you enter text into an Empty Window, the text is not automatically stored in a file. To avoid losing the text when you deactivate the window or restart your system, you need to store it in a local file. To store text in a local file, first type or copy the contents into an Empty Window. The subject of the text could be your comments on this tutorial, the names of the people that you have met today, or anything else that you can think of.

When you are finished entering text, select the name that you would like the file to have (if it is not in the text, you can type it anywhere on the screen and then select it. One convenient place to type is the blank space following RS! in the lower subwindow. To do this, first select the colon (←:) with Point to set a type-in point.) A file name is limited to 100 characters, and cannot contain any spaces, or question marks. Plus sign, minus sign, period, and dollar sign are the only special characters that are acceptable for a file name. (The screen will flash if you attempt to name a file with an illegal name.)

When you have SELECTED the name that you wish your file to have, you need to invoke the "Store" command. To do this, Chord anywhere in the Empty window, bring the File Window menu to the top of the stack, and select Store. Just typing the name on the screen is not enough; you must have the name selected (highlighted) when you invoke the Store command. If you don't have a selection, there is no way for Store to know what you want the name of the file to be.

When you have invoked Store, a message will appear in the herald that gives the name of the file, followed by a parenthetical expression telling you whether it is an old file name or a new file name. If it is an old file name, confirming the command (explained below) will cause the current file to be rewritten on the old file, and you will lose the contents of the old file. You will always be provided with the information on whether it is a new or an old file name, so that you will not inadvertently rewrite an existing file.

The cursor now looks like a mouse. This image is asking you for confirmation of your command. Click Point; this will confirm the command. (To abort, click Adjust.) When you have confirmed the command, the new name of the file will appear in the herald of the window, and the file will be stored on your local disk.

(Note that you will not be able to set a type-in point when you have stored a file in a window. A later message will discuss how to edit an existing file.)

start
01974 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: EM Symbiote
To: NewUsers: ;

Look at the top of your Empty window, below the name stripe (window herald). There are two subwindows at the top of it; the uppermost of these is called the Editable Menu (EM) Symbiote. This name indicates that the subwindow can be edited to contain any collection of commands. The word "symbiote" signifies that the subwindow functions independently of the window itself; you can attach or detach a symbiote menu without changing the properties of the window itself. Symbiotes are added to the window to make some of the more frequently used commands easily accessible without having to use pop-up menus.

For example, your symbiote should contain the word Store. This is the same Store command that you just invoked from the File Window menu. Thus, the accelerated way to store a file is to select the name of the file and click Point over the word Store in the symbiote menu.

You can edit the list of available commands. However, when editing the list of commands, you will not be able to use Point to select the text. The reason for this restriction is simple; clicking Point at a word in this symbiote will invoke the command.

To set a type-in point in the EM, you will have to use PROPS-Point, rather than just Point. For example, to delete a command, hold down the PROPS key and click Point following the command to be deleted. Now backspace over it. Enter a new command, if you like, or retype the old one. For now, don't worry about it if you don't recognize all of the commands listed in your EM symbiote; you will learn about most of them later. The purpose of this lesson is simply to make you aware of the EM symbiote as an alternative to your pop-up File Window menu. You will also learn later how to permanently set the list of commands that appear in your EM symbiote.

start
01648 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Loading a file
To: NewUsers: ;

Deactivating the window in which a file was stored does not destroy the contents of a file, although it does remove the text from the File window. Deactivate your File window (the one with a file loaded in), and then select EmptyWindow from the Inactive menu to bring it back on your screen. It will once again be an Empty window.

To load your file into the Empty window so that you can again view its contents, type the name of the file, select it, and invoke "Load" from the File Window menu or the EM symbiote. When you have loaded a file, the name in the herald will change from "Empty Window" to the name of the file that you have loaded in.

Now invoke the Create command, either from the EM symbiote or the File Window menu, to get another Empty window on your screen. You will be asked to confirm the command with Point. (The location of the cursor when you click Point will be the location of the new Empty window.)

You can use this window to practice another way of loading a file. Type the name of a file into the Empty Window (make sure you type directly into the file subwindow and not into any of the other subwindows), and then press the DOIT key (this key is labelled MARGINS on your keyboard.) This will load the specified file into that window.
Now invoke Destroy, either from the EM symbiote or from the File Window menu, to get rid of one of your File windows. This command destroys only the window; it does not affect the file itself.

start
01377 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Editing a file
To: NewUsers: ;

To make changes in your file, you must invoke the Edit command. Try it. Notice that the herald of the window changes to indicate that you are editing the file. You can now edit the file in any way that you like; however, if you attempt to edit a file without invoking the Edit command, the screen will flash and your keystrokes will be ignored.

When you have finished editing a file, you will have to invoke Save to save your changes. Changes that are made to a file while it is being edited are not actually made to the file until you have invoked this command; if you deactivate a window that is being edited you will lose all of your edits.

Be careful to distinguish between the Save command and the Store command. Save is used to save the contents of a File window under the name that appears in the herald of that window; it is used when you are changing an existing file and wish to save the new version.

Invoking Store in a File window asks the window to take any selected text as its argument and store the contents of the File window under the selected name; this command can be used to initially name the contents of an Empty window or to change the name of an existing file.

start
02415 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: DMT
To: NewUsers: ;

You should now be familiar with the basics of window and text manipulation in the Xerox Development Environment. If you had difficulty with any of the things covered in this tutorial, you should go back and practice them until you are really comfortable. The last thing that this tutorial will teach you is how to leave your screen when you leave your office for a while. When you are ready to go home for the night, or when you know that you will be away from your machine for a long period of time, you should

"cover" your screen with the DMT tool. (The reason for this name is purely historical; the letters do not relate to its current function.) DMT allows you to turn your screen black when you are not working so that the phosphor on your screen will not wear out.

You can return from DMT at any time by pressing the STOP key.

DMT may be listed in your Inactive menu. If you select it from this menu, your screen will go entirely black, except for a small square that flashes the day and time. Another way to invoke DMT is to move your cursor to the executive window and type DMT, followed by a carriage return.

To deactivate DMT, you can press the STOP key or invoke the Deactivate command in the Window Manager menu. (You can manipulate the DMT pattern just as you can any other window.) Your screen will reappear just as it was before you invoked DMT. If you are through for the day, you can invoke DMT now and go home. If you are not through, you should try DMT anyway, just so that you are familiar with it.

There are many alternatives to DMT available, if you would like a more interesting pattern to display on their screen. Most of these alternatives are classed as unsupported utilities, which means that they are not part of the standard released system. These tutorials in general discuss only the released tools, since there are a large number of "hacks", and it is difficult to single out "good" ones from "bad" ones. However, once you are familiar with the basics of the system, you should take a look at the unsupported utilities that are available.

Some examples of such programs are BrushDMT.bcd, Poly.bcd, KineticFractal.bcd and SpaceOut.bcd, all of which are DMT alternatives. Try them out if you like.

start
00813 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Reaching the next tutorial
To: NewUsers: ;

When you are ready to go on to another tutorial, Chord over the word File: in your command subwindow. A menu will appear with a list of all possible tutorial or mail files. When you select the next tutorial, called TeachFiles.nsmail, from this list, this mail file will be put away, and TeachFiles.nsmail will appear in your MailTool window, ready for you to learn about the file systems of the Xerox Development Environment.

(Note: there are a total of 6 tutorials for non-programmers, and a total of 9 tutorials for programmers. You should expect to take about two full days to read through all the tutorials.)

start
01291 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Make sure you are reading the right tutorial
To: NewUsers: ;

There are two tutorials for learning about program development in the Xerox Development Environment, TeachCompile-Bind-RunWithCoPilot,nsmail and TeachCompile-Bind-RunWithSword.nsmail.

For the most part these files contain much of the same information, but they are each slanted toward a different debugger (CoPilot and Sword, respectively). You do not need to read through both tutorials. Which you use depends on the version of software that you are running. CoPilot is the debugger that is "built-in" to 12.3 and earlier CoPilot bootfiles. Sword is a new debugger that will only run on 12.3 Tajo bootfiles. With the next release of the Xerox Development Environment, Sword will replace CoPilot as the built-in debugger.

To find out what version of the bootfile you are currently running, look at the left side of the Herald window. It should say something like "CoPilot 12.0 of ..." or "CoPilot 12.3 of ...". If it says "Tajo 12.3 of ..." or "Tajo 14.0 of ..." you should be reading the TeachCompile-Bind-RunWithSword tutorial. Otherwise, continue to the next message.

start
01545 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Programming in the Xerox Development Environment
To: NewUsers: ;

This tutorial will introduce you to the steps involved in turning a newly written source file into a working program. It is purely "how-to"; that is, it will teach you the mechanics of using programming tools, but will not teach you how to actually write programs in Mesa. When you are through with this tutorial, you probably won't be ready to write your first program in the Xerox Development Environment, but you should have a general idea of the steps and tools involved, so that you will be ready to go as soon as you learn about the Mesa language.

Before starting the next lesson, make sure that your Herald window and your Executive are active. You will also need the files:

```
FactorialToolDefs.bcd
FactorialToolAImpl.mesa
FactorialToolBImpl.mesa
FactorialTool.config
```

as well as these system files:

```
Exec.bcd
Format.bcd
FormSW.bcd
Heap.bcd
Process.bcd
Put.bcd
Tool.bcd
ToolWindow.bcd
Window.bcd
```

on your local disk. Use your File Tool or your Executive to verify that you have these files and that they are on your search path. If you are missing one or more of them, you will need to ask for assistance.

We recommend that you have completed Teach6085Booting.nsmail or Teach8010Booting.nsmail before starting this tutorial.

start
01956 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Compiling
To: NewUsers: ;

The Mesa compiler does not have a window of its own. Rather, it runs from the Executive, or from a tool called Command Central.

Running the Compiler from the Executive is just like running any other tool from the Executive: you just type "Compiler foo" (where 'foo' is the name of the program to be compiled). You can compile a group of files by separating the names with spaces, as in "Compiler MyProgram MyOtherProgram Fred". Capitalization of the file name does not matter, and you can abbreviate "Compiler" if you like. You don't have to include the .mesa extension when you type the file name - it is assumed.

The Mesa compiler is a six pass compiler. Try compiling FactorialToolAImpl.mesa from the Executive. While the compiler is running, look at your Herald window. A small cube appears during compilation. This cube has one side for each pass of the compiler. As the compiler enters a new pass, the cube turns to show the corresponding face. In the Executive window, the passes are represented by dots following the command. When errors are found, the number of dots tells you the number of the pass during which the errors were found. (Of course, the last shown side of the die that is shown in the Hearld window also represents the pass during which the errors were found.)

In this case, the compilation will be successful, and you will see a message that gives the number of lines of code, and the number of seconds that the compilation took. When there are errors, you will see how many errors there were, and how long the compiler ran until it encountered them. To see the full compiler log (i.e., any error messages), you will have to load the file compiler.log into an Empty window.

The output of the compiler is a file called FactorialToolAImpl.bcd.

```
*start*
01464 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Compiling from Command Central
To: NewUsers: ;
```

You will often run the Compiler from Command Central instead of from the Executive. Command Central is a tool specifically designed to simplify the processes of compiling, binding, and running a program.

Bring up Command Central on your screen. (You may have to run CommandCentral.bcd from the Executive if you can't find the CommandCentral window on your screen.) The command subwindow has both a Compile! command and a Compile: field. To compile a file (or files) from Command Central, just fill in the name of the file(s) to be compiled in the Compile: field, and invoke Compile!. Compile FactorialToolBImpl.mesa. (Remember you don't have to include the .mesa extension. It is assumed.)

When the compilation has finished, the complete log file will be displayed in the bottom subwindow of the Command Central window. For your convenience, this file is automatically loaded into the Command Central text subwindow when the Compiler has finished.

When you compile from Command Central, the cube will appear in the Herald Window just as it did when you compiled from the Executive. Other information, such as the number of passes completed, and the code size (for successful compilations) will appear in the message subwindow of the Command Central window.

```
*start*
01890 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Compiler switches
To: NewUsers: ;
```

There are a number of compiler switches that allow you to specify details of compiler operation. For example, the /b switch tells the Compiler to turn on bounds checking; /e tells it to include error messages; /u to give warnings about uninitialized variables. The complete list of compiler switches and their default values is in the Compiler chapter of your XDE User's Guide.

You can set default compiler switches in your user.cm file. The entry CompilerSwitches: can be placed in either your [Executive] section or your [CommandCentral] section, or both. (If you do not have a section for these tools in your user.cm, you can add one.) For now, we suggest that you set your switches to beu-j (the j switch activates cross jumping; a - preceding a switch turns it off).

As with other tools, you can override the default values in your user.cm if you like. To specify a different set of switches when running the Compiler from the Executive, just type the list of switches (preceded by a /) on the command line. Compiler switches take the same form as other Executive switches: that is, they can be either global or local. For example, to turn off bounds checking and uninitialized variable checking, you would use the command "Compile/-b-u Myprogram" (global switches) or, equivalently, "Compile MyProgram/-b-u" (local switches).

To set compiler switches in Command Central, just invoke Options! to bring up the options window, and edit the switches that are listed under Compiler Switches:. These switches will override the settings in your user.cm; they will be reset to the default user.cm values each time that you deactivate and

reactivate the tool (and each time you boot your system).

start
01508 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Binding
To: NewUsers: ;

Compiling a program generates an executable object file (if there are no errors). However, in most cases you will not immediately run the object files generated by the compiler. Instead, you will usually want to use the Binder.

The Binder is a tool that groups individual object files together into a single large object file. (The Binder is thus similar to a linker.) Both the Compiler and the Binder produce object files; the Compiler produces simple object files from source files, and the Binder produces complex object files from simple ones.

For example, if you want to use procedures or types that are found in another module, you will need to use the Binder to associate your files with the files from which you would like to use symbols. The Binder also allows you to group files together in a specific order, so that you can load a single file instead of a large group of files.

The Binder takes as input a file with the extension .config (short for configuration). A .config file is essentially a list of individual object files that you want to group together. You will have to wait until you take the Mesa Course or until you attend a lecture on the Mesa language before you are ready to write a .config file; for now, you will just have to settle for knowing what to do with a config file once you have one.

start
01195 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Using the Binder
To: NewUsers: ;

Running the Binder is much like running the Compiler. To use the Binder from the Executive window, just type "Binder foo". (Where 'foo' is the name of your .config file.) You don't need to include the .config extension; this will be assumed. Similarly, to run the Binder from Command Central, fill in the name of the file in the Bind: field, and invoke the Bind! command. Bind FactorialTool.config now, either from the Executive or from Command Central. This will produce an object file called FactorialTool.bcd.

The file binder.log contains the log file for the most recent binder command. This file is automatically loaded into the Command Central text subwindow (as was compiler.log); if you bind from the Executive, you will have to load it into an empty window manually.

You can also set Binder switches, just as you did for the Compiler. The default Binder switches are set in your user.cm file; the complete list of available switches can be found in the Binder chapter of your XDE User's Guide.

start
01250 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: File naming conventions
To: NewUsers: ;

Since the Compiler and the Binder both produce files with the .bcd extension, it is important for all of the modules that make up a program to have distinct root names. For example, if you compiled a file called FactorialTool.mesa, you would then have a file called FactorialTool.bcd. If you then tried to bind a file called FactorialTool.config, you would run into trouble, because the binder would try to create a file called FactorialTool.bcd and it would find that such a file already existed. Depending on which files conflict, the Binder may give you an error message or it may just overwrite the earlier file.

The best way to avoid mistakes is to adopt a consistent naming convention. You should make sure that all .mesa and .config files have distinct root names. Thus, program modules that implement a particular interface should be named MumbleImpl.mesa, or MumbleAImpl.mesa and MumbleBImpl.mesa when there are multiple implementation files.

You also need to be sure that you include the extension (.mesa or .config) when you name a file.

start
02224 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Logical volumes
To: NewUsers: ;

Once you have an object file, either a simple one produced by the Compiler or a more complex one produced by the Binder, you are ready to run it and see if it works. Surprisingly enough, this step is probably the one in which the Xerox Development Environment differs most from the systems that you are accustomed to. The philosophy of the Xerox Development Environment is that programs should be written, compiled, and debugged in a separate logical volume from where you run them.

In the XDE, the word "volume" can mean either a physical volume or a logical volume. A physical volume corresponds to a storage device, typically your hard disk. A logical volume is usually a subset of a physical volume; there are usually several logical volumes on a single physical volume. (However, a large logical volume could span several physical volumes. There can be no more than 10 logical volumes on a particular physical volume.) Each logical volume is largely protected from actions in other logical volumes. Thus, having a separate logical volume for running your test programs ensures that they have a fresh and clear test area in which to run.

This is called the "world-swap" approach to debugging. When you are ready to run a test program, you start from a "debugger" volume (the one in which you wrote and compiled your program.) Before a test program can be run, the object version must be copied to the test volume, the current contents of real memory must be swapped out and written to a file, and finally the test volume booted (replacing the contents of real memory). This is called a "world swap" to the client volume. When you are done testing your program in the client volume, you can "world swap" back to the debugger via a similar process.

We discuss world swapping in greater detail in debugger.nsmail, so you don't need to worry about it too much now. For now, you just need to be familiar with the theory that you will be executing your programs in an entirely different logical volume from where you develop them.

start
01472 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Tajo volume
To: NewUsers: ;

The name of the volume in which you will be running your test programs is Tajo. Some machines will have a Tajo volume; other machines will have the Tajo boot file stored on the another volume. The first thing that you should do is find out which way your machine is set up. To find this out, Chord in the Herald window and bring up the Boot From: menu. If Tajo is listed on this menu, then Tajo is the name of your test volume. Otherwise, the Tajo boot file is on another volume. Don't boot either volume now; just figure out the anme of the volume on which your Tajo boot file is stored..

The rest of this tutorial refers to the test volume as Tajo, but you should substitute another volume name if appropriate. You should also edit your user.cm file to include the information about which way your machine is set up. Load your user.cm into a file window, and find the [Executive] section. Make sure that there is a ClientVolume: entry there; it should have the name of the volume where your Tajo boot file is stored.

(If your user.cm file is wrong, the changes you make will not take effect until you next boot. If you need to change your ClientVolume: entry on the short term (without booting), you can just edit the ClientVolume: entry in the Command Central options window.)

start
02165 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Running the program
To: NewUsers: ;

You will be running your programs in Tajo, but the object file that you want to run is on the CoPilot volume. Thus, you will have to copy the object file from CoPilot to Tajo, and then boot Tajo, and finally run the program from the Tajo Executive. If you have never booted Tajo, you will have to do so before you can run a program. (Boot Tajo from the Herald Window of your CoPilot volume, wait until you reach Tajo, then press SHIFT-STOP together to return to CoPilot.)

There are two ways to accomplish these steps: an easy way, and a hard way. In this lesson, you will learn the hard way; that is, you will perform all of the steps manually. You will rarely have to do this process manually, but it is important that you do it at least once so that you understand exactly

what is happening. The next lesson covers the easy way.

Type "open tajo/w" (or "open user/w") in your Executive. The /w switch specifies that the volume is being opened for writing.

Now copy the file onto the Tajo volume, with the Executive command line "Copy <Tajo>ToolFactorial.bcd ← ToolFactorial.bcd". After copying the file, type "close tajo" to close the Tajo volume.

Now that you have stored the file onto Tajo, you just have to get there and run the file. When you reach Tajo, you will need to load and run the file ToolFactorial from the Tajo Executive. This program creates a tool window, with three fields in its form subwindow: Number=, Format: and CalculateFactorial!. The number field is used to specify the input to the command; format is used to specify the format (base) of the answer, and CalculateFactorial! actually performs the command. You can experiment with this test program if you like. (The input must be between 1 and 12.)

Remember: When you are through running the test program, you can return to CoPilot by pressing SHIFT-STOP.

Boot Tajo from the Herald window Boot from: menu now. Remember to run ToolFactorial from the Executive window when you get there.

start
01150 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Command Central's Run! command
To: NewUsers: ;

The easy way to run a program is via Command Central. To run a program from Command Central, all you have to do is fill in the name of the program in the Run: field, and invoke the Run! command. When you invoke Run!, the object file is copied to the Tajo volume, Tajo is booted, and finally the program is loaded from the Tajo Executive. Try filling in ToolFactorial in the Run: field and then invoking the Run! command, if you like.

There are also switches that you can specify in the Run: field. For example, if you enter filename/d in this field, the debugger will be called immediately after the tool is loaded (so that you can set breakpoints, or look at the state of things). There are several other switches that can appear here; check the Command Central chapter of your XDE User's Guide for a full list.

In general, you should always run test programs in the Tajo volume, either from CommandCentral or by copying them over manually.

start
00878 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Command Central's Go! command

You have now had a chance to compile, bind, and run a program in the Xerox Development Environment. The easy way to sequence these three steps is with Command Central's Go! command.

The Go! command invokes the Compile!, Bind! and Run! commands sequentially. Thus, if you fill in the name of a file or the Compile: field, the Bind: field, and the Run: field, and then invoke Go!, you will not have to do anything else until you end up in Tajo. Of course, if the compiler or the binder finds errors, the other commands will not be invoked. Thus, for example, if the compilation is not successful, the Go! command will abort after the compilation, and neither Bind! nor Run! will be invoked.
start
00532 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The debugger, CoPilot
To: NewUsers: ;

You now know how to compile, bind, and run a test program in the Xerox Development Environment. However, there's a catch to all this. As you know, no test program really runs perfectly the first time. So, to prepare you for the first time you really test a program, you should work through the next tutorial, called TeachDebugger.nsmail.

start
01293 00071 US
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Make sure you are reading the right tutorial
To: NewUsers: ;

There are two tutorials for learning about program development in the Xerox Development Environment, TeachCompile-Bind-RunWithCoPilot.nsmail and TeachCompile-Bind-RunWithSword.nsmail.

For the most part these files contain much of the same information, but they are each slanted toward a different debugger (CoPilot and Sword, respectively). You do not need to read through both tutorials. Which you use depends on the version of software that you are running. CoPilot is the debugger that is "built-in" to 12.3 and earlier CoPilot bootfiles. Sword is a new debugger that will only run on 12.3 Tajo bootfiles. With the next release of the Xerox Development Environment, Sword will replace CoPilot as the built-in debugger.

To find out what version of the bootfile you are currently running, look at the left side of the Herald window. It should say something like "Tajo 12.3 of ..." or "Tajo 14.0 of ...". If it says "CoPilot 12.0 of ..." or "CoPilot 12.3 of ..." you should be reading the TeachCompile-Bind-RunWithCoPilot tutorial. Otherwise, continue to the next message.

start
01575 00071 US
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Programming in the Xerox Development Environment
To: NewUsers: ;

This tutorial will introduce you to the steps involved in turning a newly written source file into a working program. It is purely "how-to"; that is, it will teach you the mechanics of using programming tools, but will not teach you how to actually write programs in Mesa. When you are through with this tutorial, you probably won't be ready to write your first program in the Xerox Development Environment, but you should have a general idea of the steps and tools involved, so that you will be ready to go as soon as you learn about the Mesa language.

Before starting the next lesson, make sure that your Herald window and your Executive are active. You will also need the files:

```
FactorialToolDefs.bcd
FactorialToolAImpl.mesa
FactorialToolBImpl.mesa
FactorialTool.config
BWSCalculator.bcd
```

as well as these system files:

```
Exec.bcd
Format.bcd
FormSW.bcd
Heap.bcd
Process.bcd
Put.bcd
Tool.bcd
ToolWindow.bcd
Window.bcd
```

on your local disk. Use your File Tool or your Executive to verify that you have these files and that they are on your search path. If you are missing one or more of them, you will need to ask for assistance.

We recommend that you have completed Teach6085Booting.nsmail or Teach8010Booting.nsmail before starting this tutorial.

start
01964 00071 US
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Compiling
To: NewUsers: ;

The Mesa compiler does not have a window of its own. Rather, it runs from the Executive, or from a tool called Command Central.

Running the Compiler from the Executive is just like running any other tool from the Executive: you just type "Compiler foo" (where 'foo' is the name of the program to be compiled). You can compile a group of files by separating the names with spaces, as in "Compiler MyProgram MyOtherProgram Fred". Capitalization of the file name does not matter, and you can usually abbreviate "Compiler" if you like. You don't have to include the .mesa extension when you type the file name - it is assumed.

The Mesa compiler is a six pass compiler. Try compiling FactorialToolAImpl.mesa from the Executive. While the compiler is running, look at your Herald window. A small cube appears during compilation. This cube has one side for each pass of the compiler. As the compiler enters a new pass, the cube turns to show the corresponding face. In the Executive window, the passes are represented by dots following the command. When errors are found, the number of dots tells you the number of the pass during which the errors were found. (Of course, the last shown side of the die that is shown in the Hearld window also represents the pass during which the errors were found.)

In this case, the compilation will be successful, and you will see a message that gives the number of lines of code, and the number of seconds that the compilation took. When there are errors, you will see how many errors there were, and how long the compiler ran until it encountered them. To see the full compiler log (i.e., any error messages), you will have to load the file Compiler.log into an Empty window.

The output of the compiler is a file called FactorialToolAImpl.bcd.

```
*start*
01472 00071 US
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Compiling from Command Central
To: NewUsers: ;
```

You may run the Compiler from Command Central instead of from the Executive. Command Central is a tool specifically designed to simplify the processes of compiling, binding, and running a program.

Bring up Command Central on your screen. (You may have to run CommandCentral.bcd from the Executive if you can't find the CommandCentral window on your screen.) The command subwindow has both a Compile! command and a Compile: field. To compile a file (or files) from Command Central, just fill in the name(s) of the file(s) to be compiled in the Compile: field, and invoke the Compile! command. Compile FactorialToolBImpl.mesa. (Remember you don't have to include the .mesa extension. It is assumed.)

When the compilation has finished, the complete log file will be displayed in the bottom subwindow of the Command Central window. For your convenience, this file is automatically loaded into the Command Central text subwindow when the Compiler has finished.

When you compile from Command Central, the cube will appear in the Herald Window just as it did when you compiled from the Executive. Other information, such as the number of passes completed, and the code size (for successful compilations) will appear in the message subwindow of the Command Central window.

```
*start*
01892 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Compiler switches
To: NewUsers: ;
```

There are a number of compiler switches that allow you to specify details of compiler operation. For example, the /b switch tells the Compiler to turn on bounds checking; /e tells it to include error messages; /u to give warnings about uninitialized variables. The complete list of compiler switches and their default values is in the Compiler chapter of your XDE User's Guide.

You can set default compiler switches in your user.cm file. The entry CompilerSwitches: can be placed in either your [Executive] section or your [CommandCentral] section, or both. (If you do not have a section for these tools in your user.cm, you can add one.) For now, we suggest that you set your switches to beu-j (the j switch activates cross jumping; a - preceding a switch turns it off).

As with other tools, you can override the default values in your user.cm if you like. To specify a different set of switches when running the Compiler from the Executive, just type the list of switches (preceded by a /) on the command line. Compiler switches take the same form as other Executive switches: that is, they can be either global or local. For example, to turn off bounds checking and uninitialized variable checking, you would use the command "Compiler/-b-u Myprogram" (global switches) or, equivalently, "Compiler MyProgram/-b-u" (local switches).

To set compiler switches in Command Central, just invoke Options! to bring up the options window, and edit the switches that are listed under Compiler Switches:. These switches will override the settings

in your user.cm; they will be reset to the default user.cm values each time that you deactivate and reactivate the tool (and each time you boot your system).

start
01508 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Binding
To: NewUsers: ;

Compiling a program generates an executable object file (if there are no errors). However, in most cases you will not immediately run the object files generated by the compiler. Instead, you will usually want to use the Binder.

The Binder is a tool that groups individual object files together into a single large object file. (The Binder is thus similar to a linker.) Both the Compiler and the Binder produce object files; the Compiler produces simple object files from source files, and the Binder produces complex object files from simple ones.

For example, if you want to use procedures or types that are found in another module, you will need to use the Binder to associate your files with the files from which you would like to use symbols. The Binder also allows you to group files together in a specific order, so that you can load a single file instead of a large group of files.

The Binder takes as input a file with the extension .config (short for configuration). A .config file is essentially a list of individual object files that you want to group together. You will have to wait until you take the Mesa Course or until you attend a lecture on the Mesa language before you are ready to write a .config file; for now, you will just have to settle for knowing what to do with a config file once you have one.

start
01195 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Using the Binder
To: NewUsers: ;

Running the Binder is much like running the Compiler. To use the Binder from the Executive window, just type "Binder foo". (Where 'foo' is the name of your .config file.) You don't need to include the .config extension; this will be assumed. Similarly, to run the Binder from Command Central, fill in the name of the file in the Bind: field, and invoke the Bind! command. Bind FactorialTool.config now, either from the Executive or from Command Central. This will produce an object file called FactorialTool.bcd.

The file binder.log contains the log file for the most recent binder command. This file is automatically loaded into the Command Central text subwindow (as was compiler.log); if you bind from the Executive, you will have to load it into an empty window manually.

You can also set Binder switches, just as you did for the Compiler. The default Binder switches are set in your user.cm file; the complete list of available switches can be found in the Binder chapter of your XDE User's Guide.

start
01250 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: File naming conventions
To: NewUsers: ;

Since the Compiler and the Binder both produce files with the .bcd extension, it is important for all of the modules that make up a program to have distinct root names. For example, if you compiled a file called FactorialTool.mesa, you would then have a file called FactorialTool.bcd. If you then tried to bind a file called FactorialTool.config, you would run into trouble, because the binder would try to create a file called FactorialTool.bcd and it would find that such a file already existed. Depending on which files conflict, the Binder may give you an error message or it may just overwrite the earlier file.

The best way to avoid mistakes is to adopt a consistent naming convention. You should make sure that all .mesa and .config files have distinct root names. Thus, program modules that implement a particular interface should be named MumbleImpl.mesa, or MumbleAImpl.mesa and MumbleBImpl.mesa when there are multiple implementation files.

You also need to be sure that you include the extension (.mesa or .config) when you name a file.

start
02006 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Logical volumes
To: NewUsers: ;

Once you have an object file, either a simple one produced by the Compiler or a more complex one produced by the Binder, you are ready to run it and see if it works. How you run the program depends on the type of application it is. If you are developing an application for the Xerox Development Environment, then you can test it either on the logical volume in which it was developed (this one) or on a separate logical volume that is set up for testing, depending on the nature of the application. If you are developing an application for the Viewpoint environment, then that application must be tested in that environment on a BWS bootfile (BWS stands for Basic Workstation). That is, it will be written, compiled, and debugged in a separate logical volume from where you run them.

In the XDE, the word "volume" can mean either a physical volume or a logical volume. A physical volume corresponds to a storage device, typically your hard disk. A logical volume is usually a subset of a physical volume; there are usually several logical volumes on a single physical volume. (However, a large logical volume could span several physical volumes. There can be no more than 10 logical volumes on a particular physical volume.) Each logical volume is largely protected from actions in other logical volumes.

In general, most XDE applications can be tested in the same volume that they are debugged. An exception to this would be if you were testing/debugging anything in the Bootfile or below (because Sword depends on those same calls). (The bootfile is made up of the program implementations of the System Interfaces defined in the Mesa Programmer's Manual and the Pilot Programmer's Manual.) If, in the future, you find yourself working on those implementations, then you would need to do "world-swap" debugging.

start
00910 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Testing in XDE
To: NewUsers: ;

When testing applications for the XDE volume, all you need to do is run the program from the Executive. To do this type "FactorialTool" in the executive. Do this now.

This program creates a tool window, with three fields in its form subwindow: Number=, Format: and CalculateFactorial!. The number field is used to specify the input to the command; format is used to specify the format (base) of the answer, and CalculateFactorial! actually performs the command. You can experiment with this test program if you like. (The input must be between 1 and 12.)

(Note: You should only use the Run: line of Command Central if you plan to do world-swap debugging. To debug in the same world, you should run from the Executive.)

start
00865 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Testing in the BWS Environment
To: NewUsers: ;

The method of testing programs on separate volumes is called the "world swap" approach to debugging. When you are ready to run a test program in BWS you start from a "debugger" volume (the one in which you wrote and compiled your program.) Before a test program can be run, the object version must be copied to the test volume, the current contents of real memory must be swapped out and written to a file, and finally the test volume booted (replacing the contents of real memory). This is called a "world swap" to the client volume. When you are done testing your program in the client volume, you can "world swap" back to the debugger via a similar process.

start
01641 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The User volume
To: NewUsers: ;

The name of the volume in which you will be running your BWS test programs is usually called User. The User volume has a BWS boot file stored on it. Some machines will have a User volume; other machines

may have the BWS boot file stored on another volume. The first thing that you should do is find out how your machine is set up. To find this out, Chord in the Herald window and bring up the Boot From: menu. If User is listed on this menu, then User is the name of your BWS volume. Otherwise, the BWS boot file is on another volume. You will need to ask the person who set up your machine what the name of the volume is that contains the BWS boot file. Don't boot any volume now; just figure out the name of the volume on which your BWS boot file is stored.

The rest of this tutorial refers to the test volume as User, but you should substitute another volume name if appropriate. You should also edit your user.cm file to include the information about which way your machine is set up. Load your user.cm into a file window, and find the [Executive] section. Make sure that there is a ClientVolume: entry there; it should have the name of the volume where your BWS boot file is stored.

(If your user.cm file is wrong, the changes you make will not take effect until you next boot. If you need to change your ClientVolume: entry on the short term (without booting), you can just edit the ClientVolume: entry in the Command Central options window.)

```
*start*
02651 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Running a program in the User volume
To: NewUsers: ;
```

You will be running your programs on User, but the object file that you want to run is on the XDE volume. Thus, the object file will have to be copied from this volume to the User volume, the User volume booted, and finally program will be run.

One method of doing this entire process is by using the CommandCentral tool. To run a program from Command Central, you should first look at the Command Central options window. Make sure that the ClientVolume: entry is set to User, and the ClientSwitches: entry are "ONdy\365" (no quotes). [Make sure you invoke Apply! to close the options window.] Then all you have to do is fill in the name of the program in the Run: field, and invoke the Run! command. When you invoke Run!, the object file is copied to the User volume, User is booted, and finally the program is loaded. All by invoking one command!

The program that you are going to run in Viewpoint is called BWSCalculator.bcd. (We have made it simpler for you by doing all of the compiling and binding for this tool since it involves knowing more about System interfaces and compatability issues that you have not learned, yet.) Put BWSCalculator.bcd on the Run: field of CommandCentral. To boot the User volume and run this program, invoke the Run! command.

There are also switches that you can specify in the Run: field. For example, if you enter filename/d in this field, the debugger will be called immediately after the tool is loaded (so that you can set breakpoints, or look at the state of things). There are several other switches that can appear here; check the Command Central chapter of your XDE User's Guide for a full list.

Once you are in Viewpoint, you must put an icon on the desktop to test the program. To get an icon, Follow these steps:

  Open the Directory folder
  Open the Workstation folder
  Open the Basic Icons folder
  Copy the Calculator icon to the desktop

Select the Calculator icon on the desktop and hit the OPEN key. This will bring up a tool window that represents a post-fix notation calculator which maintains a stack of ten elements. To put numbers on the stack, you should click a number button and then click enter. When you have at least one number on the stack and one in the display, an operation such as "+" can be performed. You can experiment with this test program if you like.

Remember: When you are through running the test program, you can return to XDE by pressing SHIFT-STOP.

```
*start*
00888 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Command Central's Go! command
```

You have now had a chance to compile, bind, and run a program in another volume. The easy way to sequence these three steps is with Command Central's Go! command.

The Go! command invokes the Compile!, Bind! and Run! commands sequentially. Thus, if you fill in the names of the appropriate files on the Compile: field, the Bind: field, and the Run: field, and then

invoke Go!, you will not have to do anything else until you end up in User. Of course, if the compiler or the binder finds errors or warnings, the other commands will not be invoked. Thus, for example, if the compilation is not successful, the Go! command will abort after the compilation, and neither Bind! nor Run! will be invoked.

```
*start*
00548 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The debugger, Sword
To: NewUsers: ;
```

You now know how to compile, bind, and run a test program in the Xerox Development and possibly in Viewpoint as well. However, there's a catch to all this. As you know, no test program really runs perfectly the first time. So, to prepare you for the first time you really test a program, you should work through the next tutorial, called TeachSword.nsmail.

start
01025 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The debugger: CoPilot
To: NewUsers: ;

This tutorial introduces the Xerox Development Environment debugger, called CoPilot.

Before you start, you should make sure that you are reading the correct tutorial. There are currently two different debuggers. Which you use depends on the version of software that you are running. CoPilot is the debugger that is "built-in" to 12.3 and earlier CoPilot bootfiles. Sword is a new debugger that will only run on 12.3 Tajo bootfiles. With the next release of the Xerox Development Environment, Sword will replace CoPilot as the built-in debugger.

To find out what version of the bootfile you are currently running, look at the left side of the Herald window. It should say something like "CoPilot 12.0 of ..." or CoPilot 12.3 of ...". If it says "Tajo 12.3 of ..." or "Tajo 14.0 of ..." you should be reading the TeachSword tutorial.

start
01835 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The debugger: CoPilot
To: NewUsers: ;

CoPilot is a source level debugger: it allows you to debug with the same language constructs and concepts you used in writing the original source program. CoPilot also allows you to make procedure calls from the debugger, to assign values to variables during program execution, and to evaluate expressions.

This tutorial contains two different kinds of messages. The first 14 messages discuss the debugger user interface and introduce many of the common debugger commands. You should read through these messages to get a general idea of the kinds of commands that are available, but don't worry about remembering all of the details.

The remaining messages provide some debugging examples, and some suggestions on general debugging techniques. While you are working through the debugging examples, you should use the earlier messages as a reference if you want more complete information on any of the commands.

When you are through with this tutorial, you will not know everything that there is to know about CoPilot, but you will hopefully have some idea of how much it can help you debug your programs. CoPilot is a good debugger; learning to use it well can save you a lot of time.

You should start this tutorial in the CoPilot volume. Bring up your CoPilot window. (If it is on the Inactive menu or tiny it is called CoPilot 12.0; if it is active, the name in the herald of the window is Debug.log.) You will also need the files called Function.mesa, MiscProcs.mesa, Heap.bcd, and String.bcd. Use your Executive or your File Tool to verify that you have these three files. If you don't have them, have someone else help you locate them.

start
01059 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The world-swap approach to debugging
To: NewUsers: ;

By now, you should be familiar with the basic idea of world-swap debugging: the debugger is in the CoPilot volume, but your test programs execute in the Tajo volume. Thus, if a test program does not work, you can debug in in CoPilot, make some changes, and then rerun it in Tajo again.

When you are executing a client program, there are basically three ways that you can return to the debugger. You can interrupt (SHIFT-STOP); this is a request to the system to stop what you are doing and return to the debugger. You can then later return to the Tajo volume and keep going, if you like.

You can also reach the debugger via a breakpoint in your code. This is essentially the same as an interrupt.

Finally, you can reach the debugger via a program error. We will discuss some of the more common errors and how to deal with them later in this tutorial.

start
01963 00071 UU

@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The debugger user interface
To: NewUsers: ;

The debugger works in command completion mode; that is, you type only as much of a command as is needed to uniquely identify it, no more and no less. The debugger fills in the rest of the command.

Set a type-in point in the debugger window and type a question mark. You will see a list of possible commands. The capitalized letters tell you how much of the command you need to type. For example, the list tells you that B stands for Break. Notice, though, that the letter "A" appears by itself. This means that there is more than one command that starts with the letter A. To see what those options are, type "A?" (Note that the letter is automatically capitalized, regardless of whether you type it in lower or upper case.)

The options that start with the letter A are AScii and ATtach. Thus, AS is ascii, and AT is attach. Try typing "at" after the command prompt. As soon as you enter the "t", the debugger fills in the rest of the command for you. If you try to type more, the debugger will interpret the remaining letters as the argument to Attach. To see that this is so, clear out the existing command line by hitting the DELETE key, and then try typing "att". The debugger will not recognize the second t, and will just kill the command.

To see the possible arguments to Attach, type "at?" on a command line. You will see that you can ATtach Condition, ATtach Keystrokes, or ATtach Symbols. (Don't worry; you aren't expected to know what any of these commands means -- yet.)

If you type something that you don't mean, you can always use the DELETE key to abort anything you have typed and return you to the top level command processor. As usual, when you give a command or provide input, you will have to enter a carriage return at the end of each line.

start
00810 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Debugger output
To: NewUsers: ;

You can also control the format (octal or decimal) for debugger output. Invoke the Options command from the CoPilot menu (which you can see by Chording in the debugger window). The values in this window are all enumerated types: that is, you can see all of the options available to you, and the value currently in effect is highlighted. You can change these values by selecting the desired format and invoking Apply!

The values that you specify in this window will apply to all output. However, you can force a particular interpretation of a number by suffixing it: a suffix of D or d forces decimal; b or B forces octal.

start
01170 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Commands for leaving the debugger
To: NewUsers: ;

Once you are in the debugger, there are a number of commands that you can use to exit the debugger.

K (ill session) kills the current debugging session; in other words, it boots the physical volume. To prevent you from accidentally booting when you didn't really mean it, this command asks for confirmation before it actually executes.

Q (uit) aborts the process that was executing when you entered the debugger; this usually deletes that process. Quit also requires confirmation.

P (roceed) resumes execution of the client program. For example, suppose that you have set a breakpoint, and returned to CoPilot to look at the state of things. When you are ready to return to the client world and continue execution of your program, you can Proceed out of CoPilot into Tajo. If you haven't booted your client volume (Tajo), proceeding will not do anything.

W (orry) is used primarily for debugging the operating system. You don't need to worry about this for now.

start
00760 00071 UU

@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at the client world: Userscreen
To: NewUsers: ;

If you are debugging in CoPilot, and decide that you need to take another look at something in Tajo, you don't have to Proceed back just to take a look at the screen. Instead, the debugger Userscreen (u) command will return you to Tajo for 20 seconds and allow you to look at the current state of the Tajo screen.

When you use the Userscreen command, control will automatically return to CoPilot when your 20 seconds are up. (You can return to the debugger before the 20 seconds are up by pressing STOP, or keep the Userscreen longer by holding STOP.)

start
00557 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: --: inserting comments in the debug log
To: NewUsers: ;

One of the possible debugger commands is --, which is used to insert a comment into the debugger log. When it sees --, the debugger will ignore all input until you enter a carriage return. Thus, you can insert comments amongst your debugging to serve as reminders of what you were doing.

Try inserting some comments in the debug log, if you like.

start
01964 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The current context
To: NewUsers: ;

The current context is the domain for symbol lookup: it consists of the current frame and its corresponding process, module, and configuration. For example, when you ask for information on a variable, the debugger will look only in the current context; if it doesn't find the specified variable within the current context, it will look no further.

Try typing "cu" for current context in the debugger window. This command will give you the name of a module and configuration, as well as the global and local frame number (G and L), and the Process State Block (PSB). (You aren't expected to know what all these things are right now: if you do know, that's great; if you don't know, you will learn about them later in your Mesa training.)

When a program crashes, the debugger is quite good at figuring out where the crash occurred, and setting the current context accordingly. Sometimes, however, such as when you interrupt from the client world to the debugger, you will have to set the context manually.

If the current context isn't the one that you are interested in, there are several commands that let you change it to something that you are interested in. In particular, SEt Configuration, SEt Module context, SEt Octal context, SEt Process context, and SEt Root configuration all allow you to control the current context. You can use any of these commands to change the context, depending on whether you are interested in a module, a process, or a configuration.

Try typing "lp" in the debug.log window, to produce a list of currently active processes. Pick any one, and use the SEt Process context command ("sep") to set the current context to that process. (Use the process number, as in "SEt Process context: 76B".) Now take another look at the current context.

start
01905 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting breakpoints
To: NewUsers: ;

Like all good debuggers, CoPilot allows you to set breakpoints so that you can check up on your program. In the Xerox Development Environment, breakpoints apply to modules that are known within the current context. Thus, you cannot set a breakpoint in your module unless you have first run the program and set the current context so that the debugger knows where to look for your code.

You can set breakpoints either from the DebugOps menu or via commands in the CoPilot window. Through the CoPilot window, you can Break Xit procedure (bx), Break Entry procedure (be), Break All Xits

module (bax), or Break All Entries module (bae). These commands allow you to set a breakpoint at entry or exit (or both) to a specified procedure, or to set breaks at entry or exit to all procedures in a specified module.

If you want to set a breakpoint on a location other than the entry or exit of a procedure, you will have to use the DebugOps menu of a file window. The DebugOps Break command uses the current selection to set a breakpoint at the closest statement enclosing the selection. Thus, if you load your source file into an Empty window, you can select the line where you would like to set a breakpoint and invoke Break, either from the DebugOps menu or from the EM Symbiote. (Note: remember, you shouldn't edit your source file while you are debugging, unless you are ready to recompile it and try again. You don't have to edit the source file to set breakpoints.)

Breakpoints are numbered sequentially throughout a debugging session. Breakpoint numbers are never reused within a session: that is, if you set five breakpoints and then remove them all, the next breakpoint that you set will be #6, and not #1 again.

```
*start*
00830 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Conditional breakpoints
To: NewUsers: ;
```

Breakpoints may also be conditional; that is, you can attach a condition so that the breakpoint is only taken when the specified condition holds.

The ATtach Condition command in the debugger takes two arguments: a breakpoint number, and a condition for that breakpoint. (If you don't know the number of the breakpoint that you're interested in, you can find it out with the List Breaks command.)

You can use any of the relational operators (< , >, =, <=, =>) to specify your conditions. You can specify the number of times that a break will be bypassed before the debugger is called. For example, "ATtach Condition #: 1, condition: 3".

```
*start*
00558 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Clearing breakpoints
To: NewUsers: ;
```

There are several commands that allow you to remove a breakpoint. CLear All Breaks, CLear All Entries, and CLear All Xits allow you to remove more than one breakpoint at once.

You can clear a specific break with the Clear Break [#] debugger command, or with the Clear command in the DebugOps menu. You can also CLear Condition, CLear Entry Break, or Clear Xit Break.

```
*start*
00633 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at breakpoints
To: NewUsers: ;
```

You can use the List Breaks command or the Display Break command to take a look at your currently active breakpoints. Display Break takes a breakpoint number as argument and lists its type, the procedure or module name in which it is found (for source breakpoints, the source text is also displayed.) Any conditions attached to the breakpoint are also displayed.

List Breaks lists the above information for all currently active breakpoints.

```
*start*
02239 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Display Stack
To: NewUsers: ;
```

Once you have returned to the debugger, either via a program error or a breakpoint, you can start actually probing around and looking at values within the current context. Display Stack is one of the most common debugger commands; it allows you to look at the runtime state.

Display Stack shows the procedure call stack of the current process. Try typing "ds" in the debugger window now. Notice that the debugger returns with some information (most likely that it doesn't have any symbols for the current module), and then gives another >. Type another question mark, and you will notice that you have a different set of commands available. The Display Stack command gets you into a different command mode, where you have a different group of commands available:

```
g       displays the global variables of the module
j(n)    jumps down the stack n levels
l       lists the source line that invoked the debugger
n       moves to the next frame
b       reverse of n; shows the previous frame
p       displays the formal parameters of the current procedure
q       quits out of Display Stack mode and returns you to the top level of the debugger.(DELETE will
also do this.)
r       displays the return values
s       loads the source file into a window (if it isn't already), and scrolls the source line that
invoked the debugger to the top of the window. The source line is also displayed in the debugger
window.
v       displays the frame's variables
```

Note: the full set of Display Stack subcommands is not always available. For example, when the debugger cannot find a necessary symbol table, the commands that allow you to look at variables and parameters are all disabled. After all, if the debugger can't find the symbols, how can it let you look at them?

Therefore, you will have to wait until the debugging session at the end of this module before you get to practice Display Stack. When you want to try it, you can look back at this message to see the possible options. Remember that you will have to Quit out of Display Stack mode before you will have access to any of the top-level debugger commands.

```
*start*
01012 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The interpreter
To: NewUsers: ;
```

CoPilot contains an interpreter that allows you to perform operations such as assigning new values to variables, dereferencing, making procedure calls, indexing, field access, displaying variables and types, and simple type conversion.

If you type a question mark in the debugger again, you will see that the first command choice listed is a space (" "). Typing a space gets you into interpreter mode. (You can also type a space to get into the interpreter from within Display Stack mode.)

For example, let the interpreter evaluate an expression for you: try typing " 2 + 6 MOD 4" in the debugger (don't forget the initial space); the debugger will do the calculation and provide the answer.

The rest of this tutorial consists of two practice debugging sessions that will allow you exercise the debugger and its interpreter.

```
*start*
01036 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Example: Function
To: NewUsers: ;
```

The first example is a program called Function, which does not execute properly. To get started, compile and run the program from CommandCentral. When you reach Tajo, type "help function" in the Tajo Executive to see how the program works, and then type the following into the Executive:

```
Function s/20
Function c/5
Function s/6 s/10 c/4
Function q/
```

At this point, the program will crash, and you will return to the CoPilot volume.

To find out what happened, bring up your debugger window. The debug log should look something like this:

```
*** Address fault, PSB: 137B, at 0, in L: 3170B, PC: 4660B (in StringsImplB, G: 37154B) ***
```

An address fault is one of the most common errors that you will run into. Basically, an address fault occurs when a program tries to access an invalid region in memory. This is generally the result of a NIL pointer.

```
*start*
01898 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The call stack
To: NewUsers: ;
```

You should almost always start a debugging session with the Display Stack command, which lets you take a look at the call stack. Execute this command now by typing DS into the debugger window. This will produce a line that looks something like this:

```
No symbols for L: 3170B, PC: 4660B (in StringsImplB, G: 37154B)  >
```

This tells you that the error occurred in the module StringsImplB, but that the module StringsImplB.bcd is not on the CoPilot volume, so the debugger cannot deduce exactly what went wrong. One way to find out more information is to retrieve the module StringsImplB.bcd; once you have this module, you can determine the line of code where the error occurred.

However, in general you will not have to do this. If you continue down the call stack with the next command (just type n), you will see the entire call stack, one module at a time. In general, you should just look down the call stack until you see your own code. There will be times when your code is not on the stack, in which case you will have to work a little harder to figure out what went wrong, but in general you will see your code somewhere on the stack.

In this case, the original cause of the error is in the module Function, but Function is not on the top of the stack when the program crashed. Thus, you should continue down the stack until you find the module Function. Your debug log should then look something like this:

```
*** Address fault, PSB: 137B, at 0, in L: 3100B, PC: 4660B (in StringsImplB, G: 37154B) ***
>Display Stack
No symbols for L: 3100B, PC: 4660B (in StringsImplB, G: 37154B)  >n
No symbols for L: 54540B, PC: 4427B (in StringsImplB, G: 37154B)  >n
Main, L: 10034B, PC: 307B (in Function, G: 106754B)  >

*start*
01019 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at the line of code
To: NewUsers: ;
```

Once you have found the correct module on the stack, you can look at the line of code where it died with the Source command or the List command. (Remember, these commands are subcommands within Display Stack mode.) Try doing one of these commands. Your debugger window should look something like this:

```
>Display Stack
No symbols for L: 3170B, PC: 4660B (in StringsImplB, G: 37154B)  >n
No symbols for L: 54754B, PC: 4427B (in StringsImplB, G: 37154B)  >n
Main, L: 55310B, PC: 301B (in Function, G: 106754B)  >s    <>cardinal ← String.StringToNumber[number,
10];
```

Thus, the program died while making the call to the procedure String.StringToNumber. Your next question might be to ask what the procedure StringToNumber is supposed to do. To find this out, you can use the Show Type command, as described in the next message.

```
*start*
01283 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: ShowType
To: NewUsers: ;
```

You can use the debugger to find out the type of various procedures and variables, provided that you have the appropriate file on your local disk. You will learn more about this later; for now, you just need to worry about the process of finding out the type of a procedure, and not about how you can tell when this will work.

To find out the type of the procedure String.StringToNumber, select the name of the procedure anywhere on your screen (this message is probably the easiest place), chord in the debugger window, and select

the Show Type command from the CoPilot menu. This will display the type declaration of the procedure in the debug log, like this:

String.StringToNumber: PROCEDURE [s: LONG STRING, radix: CARDINAL ← 10] RETURNS [UNSPECIFIED];

Thus, this procedure takes two arguments, a long string and a cardinal, and returns a result of unspecified type.

(You can also execute the Show Type command by typing SH and then the name of the procedure, but not from within Display Stack mode.)

The next step is to take a look at the values that our program is passing to this procedure.

```
*start*
02302 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at variables
To: NewUsers: ;
```

Once you know the line of code on which the error occurred, the next step is typically to start looking at the values of various variables at that point. To look at the local variables for the procedure, use the v command; to look at the global variables, use the g command, and to look at the procedure parameters (if there are any), use the p command.

Try using these three commands. You should get something like this:

```
>v
h = 410720B↑
clientData = NIL
outcome = normal
OutputProc = PROCEDURE (indirect, 10756B) [10756B] (in module ExecImpl, G: 34004B)
operation = 3447042B↑(1,100)"q"
number = NIL
cardinal = 0
answer = 0

>g
func = PROCEDURE CubeInput (in module Function, G: 106754B)
z = ?[4141422B]

>p
h = 410720B↑
clientData = NIL
```

You can also use the interpreter to look at a specific variable. For example, in this case, you we know that we are interested in the variable number, so you could look at just that variable by typing a space to enter interpreter mode, and then typing number. Your debug log would look like this:

```
> number
number = NIL
```

When you type the name of an identifier (variable, module, etc.) to the debugger, you must capitalize it correctly (i.e., the way that it is declared). In Mesa, gamma, GamMa, and GAMMA are three different variables. Remember, you can also control the format of the output using the debugger's Option command. (See the earlier message entitled Debugger output for details.)

Thus, in this case, we are making a call to String.StringToNumber, and passing in a NIL pointer (number.) Remember, address faults are generally caused by NIL pointers. To be really thorough, you could go read the code for the string procedure and find out what happens when you pass in a NIL pointer, but in general you don't need to be that thorough. Address faults almost always mean a NIL pointer, and you have a NIL pointer on the line of code where the program died, so it is a fairly safe bet that this NIL pointer is what caused the problem.

Thus, the next step is figuring out how you should go about fixing the program.

```
*start*
00745 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Fixing the code
To: NewUsers: ;
```

Look at the line of code directly preceding the line of code where the error occurred. This is the line where we do our error checking:

```
    IF (operation = NIL) AND (number = NIL) THEN EXIT;
```

Thus, if both operation and number are NIL, we exit gracefully. However, in this case number is NIL but operation is not; thus, we get through the error checking, but the program still causes an error. To fix this program, all you have to do is change the word AND to the word OR. Try making this change, and rerunning the program. This time, it should work just fine.

```
*start*
01106 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: MiscProcs
To: NewUsers: ;
```

The second debugging example is a "program" called MiscProcs. This program consists of three miscellaneous procedures grouped together in one file: there is no mainline code, so there will be no visible results when you run the program.

Load the file MiscProcs.mesa into a file window, and take a look at the code if you like. (You aren't supposed to be familiar with Mesa syntax yet, but you should be able to get a general idea of what's going on.) There are some global declarations at the beginning of the file, and then four procedures, called Factorial, FreeOldNodes, Interchange, and MakeLinkedList.

You will need to compile this program in CoPilot and then run it in Tajo. Since this program doesn't have any mainline code, you will have to hit SHIFT-STOP to return to CoPilot.

Now use Command Central to Compile and Run MiscProcs. You don't need to bind anything, since there is only one module in this program.

```
*start*
01223 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting the module context
To: NewUsers: ;
```

You should now be back in CoPilot; the debug.log will tell you that the reason you are here is that you interrupted (from Tajo). Type "cu" in the debugger to take a look at the current context.

Since MiscProcs doesn't execute any mainline code, the current context will not be the context that you are interested in (look at the module name). So, the first thing that you need to do is set the current context. In general, you will have to set the current context when you reach the debugger by interrupting, but not when you reach it via a program error.

To do this, use the SEt Module context ("sem") command, with MiscProcs as argument. You must type the name just as it appears in the internal declaration (MiscProcs: PROGRAM). (I.E., capitalization matters, and the extension should be left off, as it is not part of the internal name.)

(Remember, there are a lot of ways to specify the current context, such as by process number, module name, or configuration name.)

Try CUrrent context again to see the new context.

```
*start*
01378 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Making a procedure call from the interpreter
To: NewUsers: ;
```

You are now going to make a call to the recursive Factorial procedure. As you can see, there is no global code to make a call to Factorial; this module by itself contains no way to make calls to Factorial.

In the debug.log window, type " Factorial" (don't forget the space, which invokes the interpreter). This doesn't invoke the procedure; it just displays information about it so that you can be sure that you and the debugger are talking about the same procedure Factorial. (Again, the name of the procedure must match the internal declaration of the procedure.)

Now, to actually make a call to Factorial, type " Factorial [#]", where # is any number between 1 and

12. This number will be used as the procedure argument. The procedure will only accept input in this range; it will return 0 for the factorial of any number not between 1 and 12. (You might want to read the next paragraph before you hit a carriage return, so that you will know what to expect.)

Your screen will go grey as control returns to Tajo to execute the procedure call. When the call has finished, control will return to CoPilot, and the result will be printed in the debug.log window.

```
*start*
01721 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting breakpoints
To: NewUsers: ;
```

Now suppose that you would like to get a closer look at this recursive procedure. By setting a breakpoint as you exit Factorial, you can watch the recursion unwind. Use the debugger "bx" command to set a breakpoint at the exit of Factorial ("Break Xit procedure: Factorial".) This will be Breakpoint #1.

Call Factorial through the interpreter again (" Factorial [4]"). When you return to CoPilot, type " j" in the interpreter to look at the value of the variable j, which is the value returned by Factorial. (Don't forget the carriage return after the j.) Since this is the first time through, j will be 1, the factorial of 0. Now type p to Proceed back to Tajo and continue execution. Remember, Proceed just continues execution where you left off.

The second time that you hit your breakpoint, the value of j will still be 1 (the factorial of 1). Use the interpreter to check this. Now try the Display Stack command. This command will tell you that you are in the procedure Factorial, which is in the module MiscProcs. When you are in Display Stack mode, type a question mark to see the commands that are available to you within this mode. Try L, P, R, S, V, and G. When you are through experimenting with the Display Stack subcommands, use Q(uit) or DELETE to return to the upper level command processor.

If you are now convinced that the program works, use Clear All Breaks, or Clear Break #1 to remove the breakpoint. Then Proceed back to Tajo; the procedure will finish executing and tell you that the factorial of 4 is 24.

```
*start*
00916 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Procedure Interchange
To: NewUsers: ;
```

Take a look at the procedure Interchange in MiscProcs. To see that the array (A) is currently empty, type " A" in the debugger; you will see that the array has 13 elements, and that they are all currently 0.

Suppose that you want to change some of the values in this array. For example, you can set the first element to 7 by typing " A[0] ← 7". Change the values of the first three elements of this array to be 7, 8 and 9. The array should now look like this: A = (13)[7, 8, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Now call Interchange, and interchange the values of the first and last elements (" Interchange [0, 12]"). When you return to CoPilot, take another look at the array to verify that your switch has in fact been made.

```
*start*
01516 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: MakeLinkedList
To: NewUsers: ;
```

The last procedure in MiscProcs is called MakeLinkedList. This procedure builds a simple linked list. For example, if you specify a list of 5 elements, headNode will point to an element that contains the letter E; headNode.next to the letter D, and so on. To see the first element, type " headNode↑, to see the second, type " headNode↑.next↑", etc.

You are undoubtedly familiar with linked lists, so we will leave you to experiment with the debugger as much as you like. Set some breakpoints in MakeLinkedList; display the stack; change some values if you like; make a breakpoint conditional. Remember, the fact that your program runs in Tajo and your development environment runs in CoPilot means that you cannot harm any of your development tools by experimenting.

If you run into trouble, you can always look back at earlier messages in this file, or reference the debugger chapter of the XDE User's Guide. And take heart: all of the material in this tutorial will be covered in the first training unit, and you will have a second chance to learn anything that you have had trouble with or don't remember how to do.

When you are through experimenting, there is one additional tutorial that you might want to look at. This final tutorial, called TeachToolBuilding.nsmail, that discusses how to build new tools for the XDE.

start
01052 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: File systems: local and remote
To: NewUsers: ;

This tutorial discusses how to store, list, delete, retrieve, and organize files, both on your local disk and on remote file servers. A remote file server is a large capacity file storage device that is shared among many users. Thus, you will use remote file servers as a place to store files that others will need access to, and as a place from which to obtain copies of files created by others. Because remote servers have larger capacity and better reliability than your local disk, you should also back up important local files on your remote server.

The early lessons in this module discuss local files; the later lessons discuss remote files. You should create a practice local file to use for this tutorial, if you don't already have one around. (Creating a local file is covered in TeachBasics.nsmail, if you don't remember how to do it.)

start
01411 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Naming conventions
To: NewUsers: ;

By convention, a file name has two parts, the main name and the extension, which are separated by a period. For example, the file name "Introduction.doc" has main name "Introduction" and extension "doc". A file name can contain more than one period.

Generally, the main name identifies the file and the extension identifies the type of the file. Although you are free to use any extension that you like when naming files, there are several common extensions that are used throughout the Xerox Development Environment. Standard extensions include:

```
.bcd            --Mesa object file
.cm             --command file
.doc            --documentation file
.ip             --interpress format file
.mesa           --Mesa language source file
.nsmail         --mail file
.txt            --text file
```

The Xerox Development Environment is generally insensitive to case in file names. Thus, ALPHA.mesa, alPHa.mESa, and alphA.mesa all refer to the same file. However, capitalization is often used as visual punctuation, especially when a file name consists of more than one word, as in TripReport or MasterList. Since most tools display file names the way they are defined, regardless of how you refer to them, you need only be careful with capitalization when naming a file for the first time.

start
01705 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Executive window
To: NewUsers: ;

There are two ways to list the files that are stored on your local disk: with the File Tool and with the Executive. This lesson covers the basics of using the Executive. Find your Executive window; if it is not active, look through your tiny windows and on the inactive list.

Bring up your Executive window. ">" is the symbol which the Executive uses to signify that it is ready to accept your input; this symbol (called a command prompt) should appear in the window.

The Executive has a standard teletype interface, which means that many of the editing functions commonly available in the Xerox Development Environment do not work in this window. For example, set a type-in point and type something after the command prompt (>), but do not type a carriage return. Now select the word and attempt to delete it. XXX will appear to indicate that the command has been aborted, but the letters will not be erased from the screen.

In the Executive window, the type-in point is always at the end of the existing text in the window. Thus, you cannot use any editing technique that requires text selection. However, you can use backspace (BS) and backword (BW) to change the end of a line, and you can copy text into the Executive window from other places on the screen. Try using BS and BW, but try not to enter a carriage return:

you might do something unexpected.

In the Executive window, pressing DELETE at any time prior to pressing RETURN will cancel the entire command line and return a blank command line.

start
01613 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Using the Executive to list files
To: NewUsers: ;

Use Delete to clear out any text you may have entered into the Executive, and type a question mark. This will give you a list of all the commands and all the files that are available to you. This will probably be a long list: when it has finished, scroll up (if necessary) so that you can see the beginning of the list.

The beginning of the list is a group of words followed by ".~". The "dot tilde" extension is a convention that is used to designate a command name. You don't have to type the extension when you invoke the command; the purpose of the extension is to enable you to distinguish command names from file names. For now, you don't need to know the meanings of all of the commands; you just need to realize that they are commands. Many of them will be discussed later in this tutorial.

The rest of the names in the list are file names. Each group of files is headed by a name of the format <Tajo> or <Tajo>tools> or the like. These headings represent local directories that are on the search path. You may only have one group, or you may have several. The next few lessons discuss what a search path is, and how to create and manipulate local directories.

Note: You may see <Copilot> instead of <Tajo>. This is derived from the name of your volume. This tutorial will continue to refer to <Tajo>, but you should make the appropriate substitution if your XDE volume is named something else.

start
01973 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Local directories
To: NewUsers: ;

<Tajo> is the name of a logical volume, which you can think of as a partition of your hard disk. Thus, the list of files that you see when you list the files "on your local disk" is in fact the list of files stored on the logical volume Tajo. There may be as many as ten logical volumes on a single physical volume, and the files stored on each are entirely separate. Thus, Tajo is the name of the root directory for all of the files stored on the Tajo logical volume.

The fully-qualified name of a file describes the path from the machine on which the file is located down to the file itself. The general form for a file name is
[MachineName]<directory>subdirectories>name.extension.

For example, the simple file name user.cm could be more completely named as <Tajo>user.cm (the file named user.cm on the root directory Tajo. (The fully-qualified name of this file would also include the name of your own machine; however, you do not need to include the name of your machine when you are referring to a local file, just as you do not need to include the area code when making a local phone call.)

You can also sub-divide your local disk into a group of local directories, which are basically just file groupings. For example, you could create a local directory called "mail", one called "programs", and one called "text". You can create as many local directories as you like. You can also subdivide them as many times as you like: you might want to have the local directory "programs" further divided into individual project names, for example.

The complete name of a file thus includes any local directories when applicable. For example, the file <Tajo>mail>schedule.nsmail would be the complete name of a file stored on the local directory called "mail" on the Tajo volume.

start
02179 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The search path
To: NewUsers: ;

Since the complete name of a file gives the path to finding that file, you might think that you need to

give the complete path name of a file each time you refer to it. Fortunately, however, this is not the case. For example, when referring to a local file name you never have to include the machine name: there is no question as to the machine being referred to; it is defaulted to your machine. You can explicitly specify a machine name other than your own, but you don't need to worry about that for now.

You can also specify a "search path", which is just a list of local directories, and the order in which they are to be searched. Creation and manipulation of search paths is done with a special tool called the Search Path tool. Find this tool and activate it.

The first line in the Search Path tool shows the current search path. When you give the simple name of a file, the system will start looking for it in the first subdirectory in the list, and will continue the search until it finds it or until the search path is exhausted.

The search path does not necessarily contain all of the local directories in existence. If a directory is not listed on the search path, you effectively cannot see the files contained in that directory unless you refer to them by their fully-qualified name. Thus, if the directory <>yes is on the search path, and the directory <>no is not; then you can refer to the file <>yes>fred as just fred, but you would have to refer to the file <>no>sam as <>no>sam in order to be able to see it. Note: <> is shorthand for <Tajo>.

Any new files that you create will automatically be stored in the directory at the head of the search path unless you specifically designate another directory.

To see all the directories on the logical volume, Chord anywhere in the Search Path window and bring the menu labelled All Directories to the top of the stack. This lists all directories on the disk, regardless of whether or not they are on the search path.

start
01583 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Adding new directories to the search path
To: NewUsers: ;

You can create new local directories with the Search Path tool. Type in "<Tajo>practice" in the field labelled Directories:. (Select the colon following the word, and then type in Practice, or copy it in from this message.) Now invoke the Create Dir! command. You should see feedback in the bottom subwindow of the Search Path tool window telling you that the Practice directory has been created. Now Chord again, and notice that the name of your new directory has been added to the All Directories menu.

To add an existing to the search path, just select the name of the directory from the All Directories menu. This will push the selected subdirectory to be at the top of the search path. Add the directory Practice to the search path.

If you had entered just "practice" in the form field, rather than "<Tajo>practice", the new directory would have been created as a subdirectory of the directory currently on the top of the search path, rather than as a subdirectory of the root directory itself. The easiest way to make sense out of this is to try it: enter "practice2" into the Directory: field and invoke CreateDir!. Bring up the All Directories menu, and notice that your new directory is <Tajo>Practice>Practice2" rather than "<Tajo>Practice2."

You can also add a directory to the search path by typing its name in the Directory: field and invoking Push!

start
01566 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Removing directories from the search path
To: NewUsers: ;

You can remove directories from the search path in a similar manner. Just bring up the Search Path menu, and select the name of the directory that you wish to remove from the search path. Remove the Practice directory, and use the Destroy Dir! command to destroy the directory. As a protective measure, the system will not allow you to destroy a directory that contains any files, or a directory that is on the search path. Thus, if you have created a <Tajo>Practice>Practice2 directory, you will have to delete the Practice2 directory before you can delete the Practice directory.

If you want to completely change the search path, you can type in your desired search path in the Directories: field and then invoke Set!

You should consult the Search Path tool chapter of your XDE User's Guide if you would like more information on the operation of the Search Path tool.

Most of the Search Path tool commands are also available from the Executive window. Check your list of Executive commands, and you should see CreateDir, PopWorkingDirectory, PushWorkingDirectory, SetSearchPath, and ShowSearchPath. You can manipulate your search path from either the Search Path Tool or the Executive or both, depending on what is most convenient for you.

For more information on the Executive commands, consult the Executive chapter of your XDE User's Guide.

start
01719 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Renaming files
To: NewUsers: ;

The Executive Rename command is used to rename local files. Rename can be used to rename the simple name of a file, as in renaming a file called Conference.temp to Conference.txt. Rename can also be used to rename the path name associated with a file; that is, to change the local directory that it is found on.

As an example of this, assume that you have a local directory named "draft", and there is a file on that local directory named StatusReport.txt. When you have finished your status report, you decide to move it from the directory "draft" to the main directory Tajo. To do this, you would enter the following line in your Executive window:

Rename <>StatusReport.txt ← <>Draft>StatusReport.txt

As you can see, the syntax of rename requires the destination file name, followed by a ←, followed by the source file name. The ← must be surrounded by spaces. (The ← character is not on your keyboard; on a 8010 you get one by typing the key labelled with a left apostrophe; on a 6085 type the key labelled with a bullet (or dot). If you can't find it, you will have to check your keyboard mapping chart.)

You can also abbreviate the Rename command if you like; ren will work just as well. Capitalization is unimportant: tajo is as good as Tajo or taJo. However, the new file name will be capitalized exactly as you type it; in fact, you can use Rename.~ to change the capitalization of a filename.

Experiment with Rename a bit. If you have trouble, you can always refer to the Executive chapter of your XDE User's Guide.

start
01891 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Deleting files
To: NewUsers: ;

The Executive Delete command is used to delete files from the local disk. Delete really does delete a file: once you have used delete, there is not usually any way to recover your file.

You are going to use scratch files to experiment with the Delete command. When you edit a source file, the edits are not immediately made to the actual file. Instead, the system creates a temporary copy of the original file, and the edits are made to that file. When you invoke Save, the edited version becomes the "real" file, and the unedited version is saved as a backup. These backup versions are labelled with a single $ following the filename. The $$ versions are also backup files; they contain all characters entered into the file during the actual editing process. Thus, a $ file is a copy of the complete, unedited file, and a $$ file is a log of the changes made during the last editing session.

To find out if you have any existing scratch files, type *$? in the Executive window. The * character is a wildcard, used to match any or all characters. Thus, this string asks the Executive to list all files that end in the character $. If you don't have a scratch file, generate one by editing a source file. (You should have a practice file around by now.)

Now try deleting the scratch file. Since this is practice, it doesn't matter whether you use the $ version or the $$ version. Generally speaking, however, the $$ versions are less useful to keep around than $ versions. You should also note that scratch files are never deleted automatically, and that you should occasionally clean up your disk by deleting scratch files that you don't need.

You can also use the * "wildcard" in local deletions.

start
02635 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)

From: XDE-Training:OSBU North:Xerox
Subject: The File Tool
To: NewUsers: ;

This tutorial has covered most of the file manipulation commands available from the Executive window. The rest of the lessons in this tutorial will use the File Tool. Therefore, you should put your Executive tool away for a while (either deactivate it or Size it), and bring up your File Tool window.

The File Tool has four subwindows. The uppermost subwindow is a message subwindow; the second subwindow is a form subwindow; the third is a command subwindow, and the fourth is a log subwindow. Take a look at the command subwindow.

At the far right of the command subwindow there is a command called List-Options! Invoke this command; a small options window should appear on top of the command subwindow. This window contains various options that you can ask for when listing information about a file. These are boolean options: that is, they are either on or off. Highlighted items are on; these items represent the information that is currently provided by a list command. To change the setting of a boolean, you need only click over it with Point. Try turning some booleans on and off. When you have set the options that you like, invoke the Apply! command in the Options subwindow. This will put your choices into effect and remove the options window.

The Local-List command lists the specified information about files on the local disk. This command operates on the files listed in the Source: field of the form subwindow. (Throughout XDE, items followed by a colon indicate that you are to enter a value following the colon. The word preceding the colon is usually a hint as to what information is required.)

Thus, for example, if you want to see all of the files on your local disk, set a type-in point and enter an asterisk (the wild card) in the Source field. (* matches zero or more characters; i.e., *abc matches abc and xabc and xyzabc.) You can request information about one file, many files, or all files. You can use the wild card anywhere in the file name; for example, if you would like to list all files that have the extension .nsmail, you could enter *.nsmail in the Source field and then invoke Local-List!

The LocalDir: field specifies which local directory is to be searched. If this field is blank, Local-List will list all appropriate files on the search path. If there is a directory in this field, the search will be restricted to the files on that local directory.

Experiment with Local-List! and List-Options!

start
01988 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Naming conventions: Domains and Organizations
To: NewUsers: ;

The rest of the messages in this tutorial discuss manipulating files that are stored on remote file servers. To access such files, you first need to understand the conventions by which they are named, and you need to be able to identify yourself to show that you have access to the specified file.

In the XDE, all objects (machines and users) are named according to a hierarchical naming system. The world of objects is divided into organizations, and the organizations are further subdivided into domains. These divisions are logical rather than physical; an organization is typically a corporation (e.g. Xerox), and its domains reflect administrative, physical, or functional divisions within that corporation. Names are of the form <simple name>:<domain>:<organization>.

The simple name of a user is just his or her legal name, such as Mark K. Hahn. Within a particular domain, a user name must be unique: thus, there can be only one Mark K. Hahn in the domain OSBU North, but there can be another in the domain OSBU South.

When referring to a user name, you need only give as much of the name as is required to uniquely identify it. Thus, if you are within the domain OSBU North, you can refer to a printer as just "Pegasus", but if you are outside of that domain you must refer to it as "Pegasus:OSBU North", and if you are outside of the organization Xerox, you must refer to it as "Pegasus:OSBU North:Xerox".

For simplicity, user names can also be aliased. Thus, Hahn might be an acceptable alias for Mark K. Hahn. Aliases are generally created at the same time as the user name; you must use the registered alias or aliases for a name. Aliases must be unique; your system administrator is responsible for ensuring that user names and aliases are unique.

start
01145 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox

Subject: The Clearinghouse command
To: NewUsers: ;

Each user has an account on a remote file server, which is a separate machine with a large file storage capacity. You should check with someone to find out the name of your server.

To be able to communicate with remote file servers, you must first be logged in. You do not have to be logged in to manipulate files on your local disk, since all files stored there belong to you, but you do need to log in to identify yourself to the remote server.

The first step is to use the Executive's Clearinghouse command to set the domain and organization.

In the Executive, type Clearinghouse, followed by a carriage return. You will be prompted for the name of a domain. Fill in your domain, followed by another carrriage return. Next, you will be prompted for an organization. Fill in the appropriate organization and then another carriage return.

Note: You must include the carriage returns. You cannot type "Clearinghouse osbu north" on one line.

start
01964 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Logging in
To: NewUsers: ;

To login, type Login (or just log) to the Executive, and follow with a carriage return. The system will prompt you for your user name (your last name; this is usually an acceptable alias.) If you have already logged in once, your name will automatically appear in the User: field. If your name is already there, just confirm it with a carriage return. If it is not there, type it in, and follow with either a space or a carriage return. If you make a mistake while typing your user name, you can use the backspace key to fix the mistake.

You will now be prompted for a password. You should have been set up with a temporary password when you were given your account. Enter the password; case does not matter. If you make a mistake, you can backspace over it, or hit the DELETE key to start over. (You will learn how to change your password later.)

Type a carriage return after you have finished typing in your password. If you have entered an incorrect password you will not immediately be informed of that fact, so be careful not to type your password wrong. If you think you have made a mistake, just login again right away; the new login will automatically override the old one.

Since each machine only has a single user, there is no need for a logout command. However, if you would like to destroy your login so that no one else can use your account to read or send mail or to retrieve files, you can do so by logging in without a password or with an incorrect one. (Note: this will not prevent you (or anyone else sitting at your machine) from looking at the files on your local disk, including your mail files. Therefore, if you are concerned about security, you will have to learn to store your files on the remote server without keeping copies on your local disk.)

start
01766 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Storing files on a remote file server
To: NewUsers: ;

Once you are logged in, you are ready to learn about remote filing operations using the File Tool.

To store a file on your remote directory, you have to tell the File Tool where that directory is. Therefore, you should set a type-in point in the Host: field and type in the name of your remote host (file server). This field tells the File Tool the name of the machine on which you want to store your file. (If you don't know the name of this server, you will have to ask someone.)

Now press the NEXT key. Your type-in point should now be in the field following the Directory: field. The NEXT key is an accelerator for moving among fields in order.

Fill in the Directory: field with your directory, usually your last name. (Your system administrator is responsible for creating and naming directories; check with him or her if you don't know the name of your directory.)

Note: You can use either / or > to separate the names of directories in the File Tool. Thus, the direcory Documentation/Tutorials is equivalent to Documentation>Tutorials.

If you like, you may also divide the remote directory into subdirectories, just as you can on the local

disk. For example, if you are working on a program and would like to have all of your work on it grouped together, you might want to create a subdirectory that is the name of the program. The name of the directory and its subdirectories must be separated by the angle bracket sign. For example, your Directory: entry might be "Miller>Current". You may use as many levels of subdirectories as you like.

start
01494 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Source field
To: NewUsers: ;

When you have filled in the name of the directory (and optional subdirectories), use NEXT to advance to the Source field. "Source" is the local name of the file or files that you are interested in; type this information into the Source field. If you wish to operate on more than one file, separate the file names with spaces or carriage returns (or use the * wildcard character, if applicable.) You should not need to include local directories, if the file is on the search path. You can specify a local directory in the LocalDir: field, much as you did with Local-List! and Local-Delete!

Now invoke Store! in the command subwindow, and your file will be stored on the remote file server under the specified directory. While the File Tool is actually performing an operation, the command subwindow is cleared except for a pair of black boxes, which "dance" to show the rate of data transfer. When the operation finishes, the command subwindow will reappear.

You can later store different versions of the file under the same name; they will automatically be given sequential version numbers. The earliest version of a file is version filename!1. Normally, when you retrieve a file from a remote server, you get the most recent version of the file unless you specifically designate an earlier version.

start
01629 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Verify option
To: NewUsers: ;

Using the File Tool to retrieve, list and delete remote files is very much like using it to store a file. You may not need to practice each of these commands, but feel free to experiment.

When you are experimenting with the Delete commands, you can use the Verify boolean of the File Tool to protect yourself from deleting something that you didn't really mean to delete. Find the word Verify near the right edge of the File Tool command subwindow. This word represents a boolean option: when it is highlighted, the option is on, when it is not highlighted, the option is off. To turn on Verify, simply click over the word (the default value is usually off).

When Verify is turned on, invoking a Delete command will not actually delete the file. Instead, the command subwindow will change, and offer you the choices of Confirm! Deny! and Stop!. Invoking Confirm! proceeds with the deletion; invoking Deny! denies it. Confirm! and Deny! appear on a file-by-file basis: if you specify the deletion of a large list of files, you will be asked to confirm or deny each one.

Stop! aborts the entire command, rather than just denying the deletion of a single file. For example, you would want to use Stop! if you acccidentally did a Local-Delete of * instead of a Local List of *. You don't want to confirm or deny each file individually; you just want to stop the command altogether.

Whether or not you use Verify is entirely up to you.

start
01493 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Deleting files with the File Tool
To: NewUsers: ;

Local-Delete! and Remote-Delete! are used to delete files from your local workstation or from a remote file server, respectively. Local-Delete! can only be used to delete files from your own local workstation; hence, you need only fill in the Source field when using this command. Using this command is equivalent to using the Executive to delete files. You can also specify a local directory if you like; if you don't specify one, the File Tool will look for the file via the current search path.

Local-Delete! allows you to delete files either one at a time or by using expansion characters (such as *) to encompass a group of files. With Remote-Delete!, however, you can only delete one version of a

file at a time; you cannot delete all versions of a file simultaneously. For example, try storing your practice file several more times so that you have three or four versions of it stored on your directory. (You will have to make changes in your file before you can store it. Thus, store a copy, make a change, store another copy, and so on.)

Fill in the fields with the correct information, and invoke Remote-Delete!. Notice that only the EARLIEST version of the file has been deleted. In general, you will only be allowed to use Remote-Delete! on files from your own directory.

```
*start*
01458 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Listing and retrieving remote files
To: NewUsers: ;
```

Retrieve! allows you to copy a file from a remote directory onto your local disk. To use this command, just fill in the Host:, Directory: and Source: fields, and then invoke Retrieve! You may retrieve a file from any remote directory to which you have access; you are not restricted to your own directory, or even to your own host. You can use the LocalDir: field to specify the local directory that you want the file to be stored in. Note: you will be asked for confirmation on file retrieval when you have the Verify option turned on.

The Dest'n: field is used when you want to rename a file. Thus, if you are retrieving a file called ThisFileHasANameThatIsTooLong, and you want to call it just LongFileName on your own machine, you would put ThisFileHasANameThatIsTooLong in the Source: field, and LongFileName in the Dest'n: field.

Leaving the Dest'n: field blank will cause the file to be stored under its original name; that is, the same name as in the source field.

The Dest'n: field is also used with the Copy! command, which copies a local file into another local file. When using Copy!, the name of the file that you are copying from goes in the Source: field; the name of the file that you wish to copy into goes in the Dest'n: field.

```
*start*
01067 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Close!
To: NewUsers: ;
```

When you are using your file tool to communicate with a remote file server, there is an open connection between your machine and the file server. If you Size or Deactivate the File Tool window when you are through using it, the connection will automatically be closed. However, if you like to leave your File Tool window active on your screen, you should use the Close! command to close your connection (and free any resources needed to maintain it) while you are not actually using the tool. This is good policy; get in the habit of using Close!.

At the far right of the second subwindow of the File Tool is a collection of mysterious-looking symbols that have been ignored in this tutorial. These symbols are boolean options controlling fine points of the operation of the File Tool. If you are interested, they are fully documented in the File Tool chapter of your XDE User's Guide.

```
*start*
01223 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The File Transfer Program
To: NewUsers: ;
```

The File Transfer Program (FTP) is a predecessor of the File Tool, and thus the two share many functions. You will want to use the File Tool for most of your filing operations; however, FTP is more useful when you need to operate on a large number of files, such as when you execute a transfer of an entire group of files from one machine to another. The FTP program runs from the Executive window, and FTP commands can thus be included in an Executive command file.

An Executive command file is just a list of Executive commands that are executed in order without requiring any interaction from you. Thus, for example, if you want to back up a group of files onto a remote file server every night before you leave work, you can write an Executive command file that contains the commands to do so. You can then run that command file without having to interact with the File Tool directly.

If you are interested in writing command files that use the FTP program, you should read the FTP

chapter of your XDE User's Guide.

start
01535 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: FSWindowTool
To: NewUsers: ;

FSWindowTool is a tool that allows you to access either local files or files on remote file servers. This is a complex tool that provides a great deal of functionality: when you want to do a filing operation and you can't figure out how to do it--try this tool. Note, however, that this tool is a Prototype, and not part of the standard XDE Desktop Product.

To use this tool, you need to run the file NSFilingConfig.bcd, and then FSWindowTool.bcd from the Executive window. When the window first appears, you have a choice of three options: local, mesa, and remote. Reasonably enough, choosing local allows you to operate on local logical volumes, remote allows you to operate on files stored on a remote file server, and Mesa allows you to operate on files in the "Mesa world." The Mesa world is essentially your XDE volume (this volume); the Mesa development environment is the old name for the environment itself. In general, you will probably use this tool to operate on files that are stored on your ViewPoint volume or on a remote file server.

For now, you don't really need to understand what logical volumes are; you only need to become familiar enough with the FSWindowTool that you can use it when you have to. The following lessons will provide a brief overview of how to use this tool. You should experiment with it as you go along.

start
02130 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Changing directory protection
To: NewUsers: ;

As an example of using this tool, assume that you want to change the access rights on your directory on your remote file server. To do this, first select the remote option on the logon window. (To do this operation, you will have to be logged in; if you aren't currently logged in, bring up your Executive window and do so. You can also use the SetDomain/Organization! command to specify the domain and organization of the file server that you are interested in. If the file server that you are interested in is not in your domain, you will have to use this command.)

Selecting the remote option will provide a list of remote file servers in the specified domain and organization. Find the name of your server, select it, and invoke the Open&List! command. This will open a window that lists the file drawers on that particular file server. (The list always contains 16 items; if you can't find the name of your file drawer, invoke the Next Page! command.)

When you have found your file drawer, invoke Open&List! again. This will open another window, this time containing a list of the files and or subdirectories on your file drawer.

To change the protection on a specific folder or file, select the name of thaXDE-Training:OSBU North:Xerox file from the list, and invoke the Attributes! command. This will open another window that contains the attributes of the specified file. One of the attributes in this list is the Access List:, which specifies the current access rights for the file.

To change this list, just fill in the name of a person or group in the Access Entry: Name: field, and select the desired attributes (Read, Write, Add, Remove, or Owner) from the form subwindow. This items are booleans; if an access is selected, invoking Change Access List! will give the specified person that access; if the access is not highlighted, invoking Change Access List! will remove that access for the specified person.

start
00874 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Experiment
To: NewUsers: ;

As you can see, this is a very complicated tool, and we haven't discussed all of its capabilities. Experiment with it now as long as you like, and make sure that you read the documentation on this tool. It is a very useful tool. You can use the Close! command in the form subwindow to close each of the windows. When you are through experimenting with this tool, you should now be familiar with most basic filing operations. If you are unsure about any of the information presented here, you should go back and review the material until you are fully comfortable with it. When you are ready for more, chord over the File: field in the MailTool command subwindow and select TeachText.nsmail.

start
00673 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The mail system
To: @NewUsers

This module focusses on the electronic mail system used in the Xerox Development Environment. You will learn how to read your mail and how to send mail to others, how to print a mail message, and how to fix a damaged mail file. This tutorial assumes that you have completed TeachBasics.nsmail, TeachFiles.nsmail, and TeachText.nsmail.

MailTool is the Xerox Development Environment tool designed to aid you in your electronic mail activities. You are using MailTool to read these messages.

start
02171 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The Mail Send Tool
To: @NewUsers

The mail system has two basic functions: sending mail and retrieving mail that others have sent to you. You send mail with a special tool called the Mail Send tool. To get this window on your screen, invoke New Form! in the command subwindow of the MailTool window. If your Mail Send window obscures any of the text in this window when it opens, adjust your windows until it no longer does so.

The Mail Send window has three subwindows: a message subwindow, a form subwindow, and a text subwindow. The text subwindow also contains some fields, such as Subject:.

To send a message, you must first specify a subject and at least one recipient for it. The Subject: and To: fields of the text subwindow are provided as a reminder that you need to provide this information. The words enclosed in brackets indicate that there are fields in the subwindow. Thus, to enter a subject for your message, click Point somewhere over the word Subject:, and then press the NEXT key. The word Topic will disappear.

Type in a subject title for your message. Remember that the topic of the message will appear in the Table of Contents of the recipient's mail window. Therefore, the topic should accurately express the content of your message so that interested people will take the time to read the message, but others can delete it without reading it. For example, if your message contains ideas for improving this tutorial, the topic might be "Suggestions: improving the mail tutorial," NOT "Tutorial" or "Suggestions".

When you are finished entering your subject, press the NEXT key again. As you have seen, this key allows you to move between fields without needing to use your mouse. Whenever a tool has fields, you can move through them using this key.

You should now be in the To: field, with your type-in point set. The word Recipients should disappear just as Topic did; the words enclosed in brackets are simply reminders of the kind of information that should go in the particular field.

start
02471 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The To: field
To: @NewUsers

The To: field should contains the user names of the recipients of your message. A user name has three parts, separated by colons. The second part is a "domain", the third part is an "organization", and the first part is the name for someone in that domain and organization. A domain is just a device for grouping related names. Domains correspond roughly to organizations within Xerox, such as OSBU (for Office Systems Business Unit) North or OBSU South.

The organization is just a grouping one level higher. For example, most employees of Xerox will have Xerox as their organization. A fully-qualified name includes all three parts, as in James R. Herz:OSBU North:Xerox. However, you don't always need to use a fully-qualified name; when there is no ambiguity, you can omit both the domain and organization, and alias the simple name.

For example, you can omit the domain and organization for recipients who are in your domain and organization, just as you may omit an area code when telephoning someone in your neighborhood. Thus, someone in OSBU North could send a message with the following acceptable message header:

Subject: Demonstration of recipient naming
To: Person1, Person2, Herz
cc: FarAwayPerson1:OSBU South

MailTool assumes that names without registries are in the sender's domain and organization, which in this case is OSBU North:Xerox. Since the person receiving a copy of the message has an explicitly named domain (OSBU South), MailTool knows to send the copy of the message there and not within OSBU North. (Note: Commas must be included between the names of multiple recipients.)

Fill in the To: field with the name of a friend, or your own name. A person's user name is often just his last name; if there is more than one person with the same last name, one of the user names will have a first initial appended. Because there is occasional ambiguity with user names, you may sometimes receive mail which was not intended for you. If this happens, you should forward the message message back to the person who sent you the message, and explain that the message did not reach its intended destination.

You may capitalize the recipient names any way that you like: Jones:OSBU North, jones:osbu north, and jOneS:OSbu nOrTh are all equivalent.

start
01781 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Other header fields
To: @NewUsers

The next field in the text subwindow is the cc: field, in which you can enter names of people to whom you wish to send copies of your message. Your own name is automatically included in this field, so that you will receive a copy of each message that you send. You may fill in this field with any acceptable recipient, just as in the To: field, or you may delete the field altogether if you do not wish to send any copies of your message, even to yourself. The cc: field is optional; the Subject: and To: fields are not optional.

When you have a message in your mail file, the name of the sender appears in the Table of Contents. For example, the messages in this file are all marked as being from XDE-Training:OSBU North. When you send a message, therefore, your name will appear in the Table of Contents for the recipient. There are times, however, when you want a name other than your own user name to appear in this field. For example, if you are acting for your group, you might want to have the message "from" your group, rather than from you personally. To do this, you can add a From: field to the message, directly below the Subject: field. In this field, you can specify the name that you would like to have appear in the Table of Contents. (When you use the From: field, MailTool will automatically generate a Sender: field that contains your own user name. Thus, when you use the From: field, the recipients of your message will be able to tell that you actually sent the message; the only effect of using a From: field is to change the name that appears in the Table of Contents.)

start
00932 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Which type of message?
To: @NewUsers

The Send As: enumerated item is used to choose the kind of message that you are going to send. If you chord in this field, you will see that there are three choices: Text, Mail Note, and Mail Note with attachment. A Mail Note is the simplest of the three, but it is limited to 8,000 characters.

If you select Mail Note with attachment, you will see that two new fields appear at the bottom of the command subwindow, Attachment Type: and Attachment File:. You should not worry about the Mail Note with attachment choice for now.

The last choice is Text message. A text message is a more complicated form of a mail note, and requires more overhead, but frees you from the 8,000 character restriction. See the documentation for more details.

start
01227 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Sending your message
To: @NewUsers

When you have finished filling in the form fields of your Send Tool, you are ready to type your message. Pressing NEXT again will advance to the <<Message>> field, delete the word Message, and set a

type-in point in the text subwindow. Now type a message, using the editing tecnniques that you already know. Type a message that consists of your comments about this tutorial, and your suggestions for its improvement. You can return to edit the form fields (the header) at any time before you actually send the message, if you decide that the information you filled in is inappropriate or incomplete.

Deliver your message now by invoking the Deliver! command in the Send Tool. Notice that the message subwindow at the top of the window gives feedback on the delivery of your message, and that Deliver! changes to say "delivered" when the message has been successfully delivered to the mail service. Notice also that "Deliver!" appears only after you have edited a blank form or after you have edited a message that has already been sent.

```
*start*
01334 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Public Distribution lists
To: @NewUsers
```

The mail system also provides a way to address messages to groups of recipients. The Clearinghouse allows the definition of "groups", which are just collections of users. For example, these message were sent by the group XDE-Training:OSBU North:Xerox. There are also some distribution lists that are left over from a previous mail system, and are specified with the format GroupName↑:PA or GroupName↑:ES or the like. These distribution lists are fairly common within Xerox.

There should be a list of existing groups available to you; ask someone in your area where to find it. You should think carefully about your choice of message and list so as not to bother recipients with messages they don't care to read. Check with experienced users to find out which groups should be used for which kinds of messages.

You can also create your own private distribution list by typing the recipients as a file, and then sending the message to the file name. For example, the messages you are reading were sent to the private distribution list NewUsers.

The correct syntax for creating a private distribution list is given in your XDE User's Guide.

```
*start*
01184 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Other Mail Send commands
To: @NewUsers
```

Look again at the command subwindow of your Send Tool. Find the commands Another!, Destroy!, and Reset!. Invoke Another!. You will now have a second Send Tool on your screen. Type in your own user name in the Subject field.

Now invoke Reset!, and click Point to confirm the command. Notice that Reset! leaves the window open, but clears it of the text you inserted.

Now invoke Destroy!. Destroy! does just that: removes the window and everything in it. If you have destroyed all your Send Tool windows, invoke New Form! in your MailTool window to get another. Destroy the two Send Tool windows on your screen. (MailTool will ask for confirmation of the Destroy command if you have edited the text window.)

"Put" allows you to store an unfinished message in a file on the local disk. You can specify the name of the file in the File: field. Later, when you want to finish sending your message, you can use Get! to get the unfinished message out of the file and back into a Send Tool window.

```
*start*
02379 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The Reply-To field
To: @NewUsers
```

When you are sending a message to a large group of people MailTool sometimes requires that a message have a Reply-To: field. This precaution is to insure that someone who answers the message will direct the answer only to the author of the message, and not to the entire list of recipients.

The "If Need Reply-To:" option on the Mail Send Tool determines what MailTool does when it decides that a message should have a Reply-To: field. Call up a new form and find this option; you might see {don't send} after "If Need Reply-to:".

A field like this one (the name of an option followed by a colon, followed by something enclosed in curly brackets), signifies that the option is an enumerated type item. This means that the possible values for the option are listed in a menu, and that you specify a value for the option by selecting a choice from the menu. To view the menu for this option, move the cursor to point at the words If Need Reply-To:, and Chord. The choices listed on the menu are "don't send", "add to form", and "send anyway".

"Don't send" means that if MailTool thinks that a Reply-To: field is necessary, it will refuse to deliver the message without one. Instead, the Send Tool will post a message in the message subwindow of your Send Tool informing you of the problem. When this happens, you can either override the need for a Reply-To: field, or add one to the form. Using the "don't send" value here serves to alert you to the fact that you are addressing a message to a large group of people, and effectively asks you to check that you know what you are doing.

"Add to form" means that MailTool will automatically add the Reply-To: field to any message that it thinks should have one, and fill it in with your name.

"Send anyway" overrides the requirement for a reply, and delivers the message without a Reply-To: field. Unless you are certain that you want everyone listed in your To: and cc: fields to receive copies of all replies to your message, it is best to make certain that there is a Reply-To: field, and that it carries your name alone. You should avoid burdening a group of people and the mail services with copies of other people's answering messages.

```
*start*
01853 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Reading your mail
To: @NewUsers
```

Now that you know how to send messages, it is time to learn how to read the messages which are sent to you. If your mailbox is currently empty, use your Mail Send Tool to send yourself some mail.

Each user of the mail system has a remote "mail server" (on the ethernet) in which mail directed to his or her user name is deposited. When you ask to read your mail, the mail messages with your name on them are removed from the remote server and attached to the end of your current mail file.

The MailTool polls your remote server every 5 minutes or so to see if you have mail waiting for you. It posts the status of this check in the Black Stripe of the MailTool. It will say something like ">>> New Mail for ... <<<" or "Mailbox empty at 12:00". When your MailTool window tells you that you have mail, invoke "New Mail!" in the command subwindow. Watch the uppermost subwindow (the message subwindow). This subwindow will print the name of the remote mail service where your mail is stored, and the number of messages that MailTool has retrieved from that service.

When MailTool has found all your new mail, it will append the new messages to your current mail file (the one that you are reading when you invoke NewMail!), and add a Table of Contents message for each. Notice that your Table of Contents has been scrolled to display the new messages, and that the title of the first new message has been selected. So the next time that you hit Display! you will be reading your new mail instead of the next message of this tutorial. To return to the tutorial, therefore, you will have to scroll your Table of Contents and select the message entitled "FlushRemote".

```
*start*
00922 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: FlushRemote
To: @NewUsers
```

Normally, when you use New Mail! to retrieve your mail, the messages are moved from the file server onto your local disk, without leaving a copy on the remote server. Sometimes, however, such as when you are not using your own machine to read your mail, you may wish to not delete the remote copy so that you can eventually retrieve the messages permanently onto your own machine.

The FlushRemote option in the MailTool options window can be used to read your mail without removing it from the mail server. (To get the Options window on your screen, invoke Options! in the command subwindow.) The FlushRemote option is usually turned on, meaning that no copy is kept on the remote server. To keep a copy on the server, turn this option off.

```
*start*
01145 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Deleting your mail
```

To: @NewUsers

When you have read your mail, you may either transfer it into another mail file, keep it in the current mail file, or delete it. This message deals with deleting mail; the next few messages show you how to move your mail.

The Delete! command in the MailTool command subwindow (not the DELETE key) deletes a message from your mail file. Select the title of this message in the Table of Contents and invoke Delete!. The message title will be crossed out in the Table of Contents, signifying that the message is marked for deletion, but the message itself will not actually be deleted yet.

If you make a mistake in marking a message for deletion and would like to restore it to good health, select the message title and invoke Undelete!. Do this now.

When you invoke Expunge!, deactivate MailTool, or change mail files, all messages marked for deletion will really be deleted and there will be no way to restore them. Be careful about the messages you delete.

start
01666 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Alternative mail files
To: @NewUsers

You can maintain different mail files for different kinds of messages just as you can maintain different categories in a file cabinet or card file. Using several mail files is a useful way to organize and categorize your mail. For example, you have been reading these messages in several different tutorial mail files, separated by subject.

You can have as many mail files on your local disk as you like; a mail file is just a text file in a special format. However, you can only read one mail file at a time using MailTool.

To see how mail files work, suppose that you want to move the next message in this file to another mail file called temp.nsmail. The To: field in the MailTool command subwindow is used to specify the name of the mail file to which you would like to move a message or messages. Enter Temp in this field. (Mail files conventionally have the extension .nsmail; you do not have to type the extension unless you want to name a mail file with an extension other than .nsmail.)

Now select the title of the next message ("Reading another mail file") and invoke Move!. If the file temp.nsmail does not already exist on your local disk, the iconic mouse will appear, asking you for confirmation of the command. Click Point to confirm.

Moving a message to another mail file causes it to be marked for deletion in the current mail file. You should use Undelete! to restore the message. (You will now have a copy of the message in each of two mail files.)

start
00642 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Reading another mail file
To: @NewUsers

Now that you have a message in the file Temp.nsmail, you need to load Temp.nsmail into MailTool so that you can read the message. You should already know how to read another mail file: chord over the word File: and select Temp.nsmail from the menu.

Go read Temp.nsmail now to see that your message is indeed in that mail file. When you are through, return to this mail file by chording over File: and selecting TeachMailSystem.nsmail from the menu.

start
01265 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The long way
To: @NewUsers

Choosing the name of a mail file from the File: field is the accelerated way of switching between mail files. However, if you have a mail file that does not have the extension .nsmail, or a mail file that is for some reason not listed in the enumeration of mail files, you can explicitly specify the name.

Bring up your options window by invoking Options!. The MailFile: field in this window contains the name

of the mail file that is currently being read. The list of mail files is also available here; chord to see that this is so. In addition to selecting mail files from the menu, however, you can manually edit the mail file listed here. To do so, just click Adjust over the colon following the MailFile:. This accelerator will delete the current contents of the field and set a type-in point. Type in the name of the mail file that you wish to read, and invoke Apply!.

To get to one mail file from another you can either edit this field or choose the name from the menu. In general, you will want to use the menu, but in special cases you may need to edit this entry manually.

start
01234 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: MailTool and the Search Path
To: @NewUsers

Reasonably enough, only mail files that are on the current search path will appear on the Mail File menu. If you want to read a mail file that is in a directory that isn't currently on the search path, you can use the "long method" described in the previous message to view the mail file without changing your search path. (That is, you can enter the fully-qualified name of the mail file in the File: field in the Options window.)

If you have two mail files with the same name on different directories, and both directories are on the search path, the name will appear twice on your File: menu. If you select either instance from the menu, MailTool will load the first instance that it finds; that is, the instance in the directory closest to the top of your search path. Thus, the only way to see the other, identically-named, mail file is to open the Options window and enter the fully-qualified name. Generally, you should avoid having two identically named mail files on different directories: it can cause a great deal of confusion.

start
01203 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: AutoDisplay and DisplayOnNewMail
To: @NewUsers

Open the Options Window again (by invoking Options!), and find the word "AutoDisplay".

AutoDisplay controls whether or not the next message will be displayed when you delete a message. If AutoDisplay is video-inverted (on), then invoking Delete causes MailTool to behave as if you had invoked Delete! followed by Display!. In other words, MailTool will automatically display the next message whenever you delete a message. If AutoDisplay is not turned on, then invoking Delete! deletes the current message from the mail file but leaves it displayed in the message subwindow.

When the DisplayOnNewMail option is turned on, retrieving your new mail will automatically display the first new message. Otherwise, when the option is off, the Table of Contents will scroll to the first new message, but it won't be automatically displayed.

The remaining options in the Options window are Hardcopy options. These options are discussed in the printing tutorial, which lets you practice the Hardcopy command.

start
01539 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Answer! and Forward!
To: @NewUsers

Find the Answer! and Forward! commands in your command subwindow.

Invoking "Answer!" allows you to respond to a particular message. Invoking Answer! creates a new Send Tool window, and automatically fills in the "To:" field with the name(s) of the sender(s) of the specified message, or the names in the "Reply-To:" field (if one exists). It also fills in the "Subject:" field and the "In-reply-to:" field. Try "Answer!" to observe its effect. This command operates on the message that is currently displayed in your MailTool window. Thus, if you want to answer a message, it isn't enough to just select the title in the Table of Contents; you have to display the message first.

"Forward" automatically copies the currently selected message(s) into the message body of a new Mail Send window. Try "Forward" now. Note that you must fill in the "Subject:", "To:" and "cc:" fields, and optionally provide a covering message. Unlike Answer!, Forward! applies to the message which is

currently selected; it is possible to select multiple messages in the Table of Contents, invoke Forward!, and have all the messages copied into one message body. One good time to use the Forward command is when you receive a message that was not intended for you; you should forward it back to the sender so that he realizes that his message was not delivered properly.

start
00516 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Append!
To: @NewUsers

The Append! command allows you to insert text from regular files into your mail file. Thus, you can select text anywhere on your screen and invoke Append!; the selection will be inserted at the end of the mail file, complete with a header and Table of Contents entry, just like all the other messages that you retrieved from the mail service.

start
01874 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The Hardcopy command
To: @NewUsers

The Hardcopy! command is used to print mail messages.

The lower half of your Options window contains options to control the appearance of mail messages that you send to the printer. For example, you can choose the font in which your messages will be printed, and the way that the text will be oriented on the page. The tutorial called TeachPrinting.nsmail discusses the use of these options; for now, you should just leave most of them as they are. (If you have already done the printing tutorial, you should be all set.)

However, the Printer: options allows you to specify the printer to which your output will be sent, and you need to make sure that this item is correct before you can try the Hardcopy! command. If you don't know the name of the printer nearest you, you should check with someone nearby who can help you.

Now type in the name of the printer in the Printer: field. When you have done so, invoke Apply! to put your changes into effect and destroy the Options window. (Invoking Apply! records changes; invoking Abort! ignores all changes and destroys the window. Tinying the window will not record any changes.)

Try using the Hardcopy command on some of your mail messages. The command will apply to any currently selected messages, and not necessarily to the message which is displayed. In other words, if you have a message selected in the Table of Contents that is different from the message currently displayed on your screen, the command will apply to the selected message and not to the displayed one. This makes it possible to select multiple messages in the Table of Contents, invoke Hardcopy!, and have all the messages sent to the printer as one file.

start
01385 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Maintain
To: @NewUsers

Maintain is a program that allows you to access the Clearinghouse database. You can use Maintain to inspect and modify information about users and distribution lists. Whe you run Maintain from the executive, you will get a tool window divided into several sections.

The first thing that you have to do is set the Level enumerated item, which governs the available commands. There are three possible levels: normal, owner, or admisistrative. All of the commands available at the normal level are also available at the other levels; owner and administrative simply provide some additional commands not present at the normal level.

In general, the top half of the form subwindow contains items use to manipulate groups, and the bottom half allows you to change items associated with a user. Thus, for example, you can use Maintain to set your password, add yourself to a distribution list, find out th possible aliases for your name, list the members of a particular distribution list, find out the owner of a distribution list (administrative level) and various other information related to the database. For complete documentation on the Maintain commands, see the Mail Tool's chapter of the XDE User's Guide.

start
01316 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff

@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The Mail File Scavenger
To: @NewUsers

Because a mail file is just a text file, you can edit the content of the messages in your file. To do this, all you have to do is deactivate MailTool, and load the file into a Source window. However, each message in a mail file has a header in a special format to allow MailTool to read the mail file. Among other things, this header includes a count of the number of characters in the message. Thus, if you edit a mail file, you will have to ensure that the header information is correct or MailTool will not be able to read the mail file.

If you have a mail file that you have edited, or that has been damaged in some other way, you can run a tool called MailFileScavenger.bcd to restore the structure of the file. The MailFileScavenger is simple to use; you should read the appropriate chapter in the XDE User's Guide when you want to use this tool.

Because it is fairly rare to have a damaged mail file, you will probably not use this tool very often. However, if you try to read a mail file in MailTool and get a message that the file is "not a properly formatted mail file", you will know to use the MailFileScavenger.

start
00468 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: What now?
To: @NewUsers

You should now be able to easily send, receive and print mail in the Xerox Development Environment. If you have any questions, you should consult the documentation or reread the messages in this file.

When you are ready to go on to the next tutorial, choose TeachPrinting.nsmail from the menu.

start
00465 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Printing documents
To: NewUsers: ;

In this section, you will learn how to print local and remote files. You will also learn how to change the font and format of your printed material to suit your preferences.

Before you start this section, you will need to know the name and location of your local printer.

start
01483 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Changing your default printer
To: NewUsers: ;

Before using the Print program, you should set up your user.cm with some default options, such as the name of your favorite printer in your user.cm file. The printer specified in your user.cm file is the printer to which your material will usually be sent; you can override it for any individual print command.

Load your user.cm file into a Source Window (type user.cm and press MARGINS). Invoke Edit. Now scroll through the file until you find a section labelled [Hardcopy]. You should edit your user.cm file so that it has a default printer name. The entry for a default printer may be somewhat unintuitive; it is Interpress, which is a kind of printer. If the name contains spaces, it must be quoted. For example, if you wanted to use a printer called Nevermore, your user.cm might look like this:

[Hardcopy]
Interpress: "Nevermore:OSBU North:Xerox"

If you are in the same domain as your printer, you do not need to include the domain and organization. For example, if you are in the domain OSBU North, then you could put just Nevermore as the name of your printer. However, if you are in a different domain or organization from your printer (which isn't normally the case), you will have to fully specify the name of the printer. Names with spaces in them must be quoted.

start
01346 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Other user.cm [Hardcopy] entries
To: NewUsers: ;

There are several other aspects of your hardcopy that you can control with user.cm entries:

The PrintedBy: entry determines the name that appears on the cover sheet at the printer. If you do not specify anything here, the name that is currently logged in will be printed on the cover sheet.

An Orientation: entry specifies the orientation of the paper. Portrait specifies standard orientation (8 1/2 by 11); landscape specifies a 90 degree rotation (11 by 8 1/2).

A Font: entry specifies the default font. There should be a font sampler near your printer that illustrates the different fonts available for that printer. You can also have separate PortraitFont: and LandscapeFont: entries, if you like.

For example, your [Hardcopy] section might look like this:

[Hardcopy]
PrintedBy: Groucho
Interpress: "Nevermore:OSBU North"
Orientation: Portrait
PortraitFont: Gacha10
LandscapeFont: TimesRoman8

Before you set the font entry, check the fonts that are available to you and make sure that your printer has the font that you want to use. When you have finished editing your user.cm, invoke Save. Now invoke Reset to clear the window.

start
01570 00071 UU

@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Printing a file
To: NewUsers: ;

You are now ready to send something to your default printer.

Type "Print filename" on a command line in the Executive. If you would like to print more than one file, you can do so by separating the names of the files with spaces. You can also use * as an expansion character to denote a group of files, as you can elsewhere in the environment.

When you type a carriage return, your material will be formatted by the Print program into an .interpress file, and then sent to the printer. (Interpress printers only accept files in interpress format; this format is essentially a program that tells the printer what to print and how to print it.) For example, if you wanted to print your user.cm file and a file called junk.txt, your Executive would look something like this:

```
>Print user.cm junk.txt
Pegasus:OSBU North:Xerox: Spooler available; Formatter available;
Printer available;
Interpressing User.cm/p1... 2 pages
Interpressing junk.txt/p1... 1 page
sending to Pegasus:OSBU North:Xerox...  Done
>
```

You will now need to go pick up your hardcopy. Your print-out will probably still be in the printer; it will have a cover sheet with your name, the date and time, and the name of the printed material on it. If there is other printed material in the printer when you pick up your own, you should remove it from the printer and file it in the shelves near the printer.

start
01118 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: $$$ and remote files
To: NewUsers: ;

If you would like to print only a small part of a large file, or text which is not part of a file, you can type $$$ instead of a file name on your command line. This will print the current selection.

For this convention to work, the text that you want to print must be the current selection on your screen. Thus, you should select the desired text, and then set a type-in point in your Executive using Adjust rather than Point. (If you try to use Point to enter the Print command, you will lose the selection of text that you are trying to print.) Experiment with printing some selections of text until you get the hang of it.

You can also print a remote file directly (without retrieving it onto your local disk). To do so, just type the fully qualified name. Again, if the name contains spaces, you will have to surround it with double quote marks. For example,

```
Print "[Hal:]<Hal Training>Mesa Unit>General>User.cm"
```

start
01060 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Printing switches
To: NewUsers: ;

The file that you just sent to the printer will have been printed with the default characteristics specified in your user.cm file. You can use printing switches to change these values for a particular instance of the command without changing their permanent setting.

For example, suppose that your standard printer is down for some reason, and that you want to send a document to an alternative printer. You don't want to change the entry in your user.cm file, since you will want to return to the original printer as soon as it is up. Instead, you can use a printing switch to override the default printer for a particular print command.

A sample command line might be "Print SeaBiscuit/h Myfile". The /h switch, which stands for "host", specifies that the document should be sent to the printer named SeaBiscuit instead of the printer specified in your user.cm.

start
01480 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: More printing switches
To: NewUsers: ;

Here is a list of some other print switches that you might find useful:

<font>/f		Changes the font to <font> for the following files. The default is Gacha8 in portrait; Gacha6 in landscape.

/c<n>		Sets number of copies to <n> (default 1).

/t<n>		Sets the tab width to <n> spaces (default 8).

A printing switch can be used either locally or globally. When used globally, a switch applies to all files being printed; a global switch is placed at the beginning of the command line. A local switch, on the other hand, applies to only one file; it follows the file name that it affects. Thus, when you are sending several files to the printer at the same time, you have the choice as to whether they share the same properties or each have different switches.

Examples:

```
Print Silly
Print Silly/l
Print /l Silly1 Silly2
Print Silly1 Silly2/c3 /p TimesRoman10BI/f Silly3
Print $$$
```

For example, the fourth example could be read "Print the file Silly1 with the default switches; print Silly2 in in triplicate; set the default orientation for the rest of the command line to portrait; set the default font to TimesRoman, 10-point, bold, italic; Print Silly3."

A complete list of the switches available, along with detailed descriptions of their effects, is in the XDE User's Guide.

start
00702 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: What now?
To: NewUsers: ;

You should now be able to obtain a printed copy of anything that you can see on your screen, and anything stored on a remote server, and how to have it look just the way you want it.

If you are not a programmer, then you are done with the tutorials and ready to enjoy XDE on your own.

If you are a programmer, then you should read the next tutorial, TeachCompile-Bind-RunWithSword.nsmail (or TeachCompile-Bind-RunWithCoPilot.nsmail depending in which debugger you have), which discusses program development in XDE.

start
01479 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The debugger: Sword
To: NewUsers: ;

This tutorial introduces the new Xerox Development Environment debugger, called Sword.

Before you start, you should make sure that you are reading the correct tutorial. In the past there have been two different debuggers. Which you use depends on the version of software that you are running. CoPilot is the debugger that is "built-in" to 12.3 version and earlier CoPilot bootfiles. Sword is a new debugger that will only run on 12.3 version and later Tajo bootfiles. After the 14.0 software release, only Sword will be available.

To find out what version of the bootfile you are currently running, look at the left side of the Herald window. If it says "Tajo 12.3 of ..." or "Tajo 14.0 of ..." then this is the tutorial for you. If it says "CoPilot 12.0 of ..." or "CoPilot 12.3 of ..." you should talk to your mentor about upgrading your version of software. If you decide not to upgrade for whatever reason, then you will need to locate an older version of this tutorial called TeachDebugger.

To continue with this tutorial, make sure that you have Sword.bcd running on your workstation (or, if you are running Tajo 12.3 then you may run Interpreter.bcd instead, if that is what you have). If you do not have it, retrieve it from your release directory or ask someone for assistance.

start
00347 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Before we begin
To: NewUsers: ;

This tutorial is quite long and covers many new things. You may wish to get a drink and settle in a comfortable chair before moving ahead!

Good luck and have fun.

start
01359 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: About Sword
To: NewUsers: ;

Sword is a source level debugger; it allows you to debug with the same language constructs and concepts you used in writing the original source program. Sword also allows you to make procedure calls from the debugger, to assign values to variables during program execution, and to evaluate expressions.

This tutorial contains two different kinds of messages. The first 28 messages discuss the debugger user interface and introduce many of the common debugger commands. You should read through these messages to get a general idea of the kinds of commands that are available, but don't worry about remembering all of the details.

The remaining messages provide some debugging examples, and some suggestions on general debugging techniques. While you are working through the debugging examples, you should use the earlier messages as a reference if you want more complete information on any of the commands.

When you are through with this tutorial, you will not know everything that there is to know about Sword, but you will hopefully have some idea of how much it can help you debug your programs. Sword is a good debugger; learning to use it well can save you a lot of time.

start
01420 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The different styles of debugging
To: NewUsers: ;

There are three styles of debugging: local debugging, outload debugging, and remote debugging. In local debugging, the debugger (Sword) shares the same address space as the client and is located on the same volume. In outload debugging, the debugger resides in a different address space than the client and on a different logical volume. In remote debugging, the debugger and the client are on different hosts on

the network.

Which style you use depends on what kind of development you are doing. Most applications that are being developed for the Xerox Development Environment can be debugged locally. Although, this is not always feasible. For instance, it is not possible to locally debug the operating system or the window package because Sword depends on them. In such cases, outload debugging or remote debugging is used. Also, applications that are being developed for the Viewpoint environment cannot be debugged locally (again, you must use outload debugging or remote debugging).

Since the object of this tutorial is to help you become more familiar with the debugger commands, you will walk through some local debugging exercises. In general, the techniques learned can be used in all styles of debugging.

start
01125 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The Sword Window
To: NewUsers: ;

Locate your Debug.log window and make it active. If it is not inactive, tiny, or active, then create a new one by one of the following methods:

1) If you are running the 14.0 environment then type "Sword" in the Executive, else
2) In the 12.3 environment, you should chord in the root area to bring up the stack of menus. Now bring forward the "Interpreter" menu and invoke the "New Interpreter" command in that menu. This command will create a new debugger window.

You will also need the following files:

    LinkedLIstImpl.mesa
    MiscProcs.mesa

  and these two system files:

    Heap.bcd
    String.bcd

  Use your Executive or your File Tool to verify that you have these four files. If you don't have them, have someone else help you locate them. (LinkedLIstImpl.mesa & MiscProcs.mesa should be in the same location as where these tutorials were stored. Heap.bcd & String.bcd can be found on the release directory.)

start
00850 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Sword form subwindow
To: NewUsers: ;

The form subwindow at the top of the debugger window contains some commands that are used frequently. This is a new feature available only in Sword; CoPilot only worked from commands typed into the file subwindow.

Because of this, most of the commands in the Sword form subwindow are also available in the file subwindow. So, there are frequently two ways to do things. When this tutorial discusses available debugger commands, the commands will sometimes be available through the form subwindow and sometimes through the file subwindow (and sometimes both). When there are 2 ways to do something you can choose the method that you are most comfortable with.

start
01981 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The Sword file subwindow user interface
To: NewUsers: ;

The file subwindow works in command completion mode; that is, you type only as much of a command as is needed to uniquely identify it, no more and no less. The debugger fills in the rest of the command.

Set a type-in point in the debugger window and type a question mark. You will see a list of possible commands. The capitalized letters tell you how much of the command you need to type. For example, the list tells you that B stands for Break. Notice, though, that the letter "A" appears by itself. This

means that there is more than one command that starts with the letter A. To see what those options are, type "A?" (Note that the letter is automatically capitalized, regardless of whether you type it in lower or upper case.)

The options that start with the letter A are AScii and ATtach. Thus, AS is ascii, and AT is attach. Try typing "at" after the command prompt. As soon as you enter the "t", the debugger fills in the rest of the command for you. If you try to type more, the debugger will interpret the remaining letters as the argument to Attach. To see that this is so, clear out the existing command line by hitting the DELETE key, and then try typing "att". The debugger will not recognize the second t, and will just kill the command.

To see the possible arguments to Attach, type "at?" on a command line. You will see that you can ATtach Condition, ATtach Keystrokes, or ATtach Symbols. (Don't worry; you aren't expected to know what any of these commands means -- yet.)

If you type something that you don't mean, you can always use the DELETE key to abort anything you have typed and return you to the top level command processor. As usual, when you give a command or provide input, you will have to enter a carriage return at the end of each line.

```
*start*
01043 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting a debugging session
To: NewUsers: ;
```

Look at the enumerated item called client:. This item is used to open and close debugging sessions. Chording over client: will give you the following choices: local, outload, remote, dormant, and setDUD.

Selecting local will create a local debugging session, selecting outload will create an outload debugging session, and selecting remote will create a remote debugging session. When selecting outload and remote you will be prompted for the name of the outload file or remote host, respectively. (You can try that now, and just Abort! when you are prompted.)

To end a debugging session, select dormant in the client: item.

setDUD is used in conjunction with the programming Interface DebugUsefulDefs. You probably will not have a use for it until you have advanced in your programming abilites. Therefore, it is not used in these tutorials.

```
*start*
01051 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: More about outload debugging
To: NewUsers: ;
```

The basic idea of outload (world-swap) debugging is this: the debugger is on the Tajo volume, but your test programs execute in another volume (maybe User). Thus, if a test program does not work, you can debug it in Tajo, make some changes, and then rerun it in the other volume again.

When you are executing a client program, there are basically three ways that you can return to the debugger. You can interrupt (SHIFT-STOP); this is a request to the system to stop what you are doing and return to the debugger. You can then later return to the test volume and keep going, if you like.

You can also reach the debugger via a breakpoint in your code. This is essentially the same as an interrupt.

Finally, you can reach the debugger via a program error. We will discuss some of the more common errors and how to deal with them later in this tutorial.

```
*start*
00621 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: More about local debugging
To: NewUsers: ;
```

If you are testing a program in the local enviroment the debugger works similarly as if you were testing in another environment. The big difference is that you do not do a SHIFT-STOP to get the debugger volume since you are IN the debugger volume. And when you set breakpoints in your code or your program crashes then a debugger window pops up on your screen showing you why the debugger was

called (instead of a world-swap).

start
01231 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: More on Remote Debugging
To: NewUsers: ;

You can't just remote debug any machine, a machine must be in a state that needs to be debugged. A "915" displayed on a Maintenance Panel (on a 8010) or cursor code (on a 6085) is the state that is calling out for a remote debugger.

In order to debug another machine you must either have its processor number or its clearinghouse name (if it is registered).

Since you probably do not know of any machines right now that need remote debugging you cannot try that. But you can try something else. A little trick used to determine your own processor number is to remote debug your own machine.

To do this, select "remote" in the client field and in the form subwindow you will be prompted for the Remote host. (If you weren't prompted, select "remote" again.) Now, type "ME" (MUST be all capital letters) as the Remote host and hit the Apply! command. Your net number and processor number will be displayed in the file subwindow. You should write this number down and keep it nearby in case your machine ever needs to be remote debugged.

start
01273 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Commands for leaving the debugger
To: NewUsers: ;

Once you are in the debugger, there are a number of commands that you can use to exit the debugger. The enumerated item go: lists some typical commands that you would execute on a client.

"proceed" continues execution of the client program. For example, suppose that you had encountered a breakpoint, and entered the debugger to look at the state of things. When you are ready to continue execution of your program, you can proceed out of the debugger. To prevent you from accidentally resuming when you didn't really mean it, this command asks for mouse confirmation before it actually executes. [This command is also available in the file subwindow as P (roceed).]

"abort" tries to abort the process that was executing when the debugger was entered; this usually deletes that process. abort also requires confirmation. [This command is also available in the file subwindow as Q (uit).]

"kill" ends the current debugging session and re-boots the client. kill also requires confirmation. [This command is also available in the file subwindow as K (ill session).]

start
00961 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at the client world
To: NewUsers: ;

If you are debugging in Tajo, and decide that you need to take another look at something in another volume, you don't have to Proceed back just to take a look at the screen. Instead, the "screen" command in the go: field, or U (serscreen) in the file subwindow, swaps to the outload world for a look at the screen. Control is returned to the debugger automatically after 20 seconds.

Note: This command only works with outload debugging. It is a no-op for local debugging.

This command is useful if, while debugging, you decide that you would like to take a look at the exact state of the display at the time of the swap to the debugger. (You can return to the debugger before the 20 seconds are up by pressing STOP, or keep the Userscreen longer by holding STOP.)

start
00439 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Starting a specified module

To: NewUsers: ;

The "start" command (in the go: field) starts execution of a specifed module. This module is specified by its global frame address which you are prompted for. [This command is also available in the file subwindow as ST (art global frame:).]

start
00557 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: --: inserting comments in the debug log
To: NewUsers: ;

One of the possible debugger commands is --, which is used to insert a comment into the debugger log. When it sees --, the debugger will ignore all input until you enter a carriage return. Thus, you can insert comments amongst your debugging to serve as reminders of what you were doing.

Try inserting some comments in the debug log, if you like.

start
01155 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: destroy! and another!
To: NewUsers: ;

destroy! destroys the window when the debugging session within it is dormant. If it is not, the screen will flash. You can try this command by setting the client: field to dormant and bugging destroy!. The window will go away. If this was your one and only Debug.log, you can get another by using one of the methods described in message # 5.

another! creates a new Debug.log window with the client being "local". You can have many instances of debugger window. This allows you to be debugging many clients at the same time. For instance, you could be debugging an outload session as well as several remote machines at the same time!

(The another! command is only available in the 14.0 environment. To get a new debugger window in the 12.3 environment, you should chord in the root area to bring up the stack of menus. Now bring forward the "Interpreter" menu and invoke the "New Interpreter" command in that menu. This command will create a new debugger window.)

start
00769 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: rep?
To: NewUsers: ;

You can use the debugger to find out the equivalent value of a number in several different formats. The command rep?! acts on the current selection, which should be a number, and prints its value in several formats, including octal, decimal and hexidecimal. For instance, select the number 18 and bug rep?. The debugger should display:

22B = 12H = 18 = = 1:2

These values are octal, hexidecimal, decimal, character, and Nibble:Nibble respectively.

For more information about the available formats, refer to the "Output Conventions" section of the Sword Chapter in the XDE User's Guide.

start
01603 00071 UU
@00045 01536 fffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: showType, and type&bits
To: NewUsers: ;

You can also use the debugger to find out the type of various procedures and variables, provided that you have the appropriate file on your local disk. The command showType! acts on the current selection, of which the syntax must be FileName.SymbolName. FileName should be the name of an interface and SymbolName should be the name of a symbol defined within that interface. You will learn more about interfaces, programs, symbols, etc. later; for now, you just need to worry about the process of finding out the type of something, and not about how you can tell when this will work.

To find out the type of the procedure String.StringToNumber, select the name of the procedure anywhere on your screen (this message is probably the easiest place) and select the ShowType! command. This will display the type declaration of the procedure in the debug log, like this:

String.StringToNumber: PROCEDURE [s: LONG STRING, radix: CARDINAL ← 10] RETURNS [UNSPECIFIED];

Thus, this procedure takes two arguments, a long string and a cardinal, and returns a result of unspecified type.

type&bits! can be used to show the type and bit layout of a selected expression. This is useful for finding the positions of fields within records. Again, the syntax of the expression must be FileName.SymbolName.

(You can also execute the Show Type command by typing SH and then the name of the procedure in the file subwindow.)

start
01719 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: options!
To: NewUsers: ;

You can control the format (octal, decimal or hex) for debugger output. Invoke the options! command located near the right side of the form subwindow and the Sword Options window will appear. The values in this window are all enumerated types: that is, you can see all of the options available to you, and the value currently in effect is highlighted. You can change these values by selecting the desired format and invoking Apply!

The values that you specify in this window will apply to all output. However, you can force a particular interpretation of a number by suffixing it: a suffix of D or d forces decimal; b or B forces octal.

In addition to setting your preferred output format, you can also set how Sword should handle certain types of problems with the four booleans: fault, uncaught, break, and calldebug. These booleans map to the following occurences:

  fault = Address Fault
  uncaught = Uncaught Signal
  break = Breakpoint
  calldebug = SHIFT-STOP

If the boolean is on, then Sword will handle the problem locally. If the boolean is off, then the workstation will crash to 915, which is looking for a debugger on the net.

The filter: line is used to list configs that you wish not to debug locally. Typically, what is listed here are some of the configs that make up the Tajo bootfile. If a crash occurs in any config that is listed on this line, Sword will not handle it locally and the workstation will crash to the net. You should ask your mentor what filters are recommended for the work that you will be doing.

start
01973 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The current context
To: NewUsers: ;

The current context is the domain for symbol lookup: it consists of the current frame and its corresponding process, module, and configuration. For example, when you ask for information on a variable, the debugger will look only in the current context; if it doesn't find the specified variable within the current context, it will look no further.

Try typing "cu" for current context in the debugger window. This command will give you the name of a module and configuration, as well as the global and local frame number (G and L), and the Process State Block (PSB). (You aren't expected to know what all these things are right now: if you do know, that's great; if you don't know, you will learn about them later in your Mesa training.)

When a program crashes, the debugger is quite good at figuring out where the crash occurred, and setting the current context accordingly. Sometimes, however, such as when you manually interrupt from the client world to the debugger, you will have to set the context manually.

If the current context isn't the one that you are interested in, there are several commands that let you change it to something that you are interested in. In particular, SEt Configuration, SEt Module

context, SEt Octal context, SEt Process context, and SEt Root configuration all allow you to control the current context. You can use any of these commands to change the context, depending on whether you are interested in a module, a process, or a configuration.

Try typing "lp" in the debug.log window, to produce a list of currently active processes. Pick any one, and use the SEt Process context command ("sep") to set the current context to that process. (Use the process number, as in "SEt Process context: 76B".) Now take another look at the current context.

```
*start*
01575 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: processes and configs
To: NewUsers: ;
```

The two booleans, processes and configs, in the form subwindow create special subwindows in the debugger when either is selected. The Process subwindow contains a list of all processes, call stacks and local variables. This is mostly the same information you saw when you did a "List Processes". The Configs subwindow contains a list of running configurations, modules, and global variables. A list of running configurations can also be obtained by the List Configurations (lc) command.

Select the processes boolean and you will see a list something like this (your list will not be exactly the same):

```
------------
   PSB: 10B, waiting CV, L: 403230B↑, PC: 2040 (in UserInputsA, G: 425004B↑)
   PSB: 52B*, waiting CV, L: 403410B↑, PC: 362 (in ITFormSW, G: 520170B↑)
   PSB: 56B, waiting CV, L: 404704B↑, PC: 18936 (in TTYSWsA, G: 431674B↑)
   PSB: 115B, waiting CV, L: 403240B↑, PC: 3322 (in MailTransfersA, G: 512320B↑)
   PSB: 120B, waiting CV, L: 406010B↑, PC: 1612 (in ActivityImpl, G: 510144B↑)
    .
    .
    .
------------
```

Select the configs boolean and you will see another subwindow with a list something like this (again, your list will not be exactly the same):

```
------------
   Tajo
   BasicHeadsDLion
   StartIncludedBcds, G:405614B No symbols.
   RightsNotice, G:412130B No symbols.
   FileTransfers
   NSFileTransfers
      .
      .
      .
------------
```

```
*start*
01618 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: More on processes and configs
To: NewUsers: ;
```

Notice that each line in these special subwindows contains a cross () at the beginning of the line. By selecting the cross, you can "zoom" a line to see more detail. For example, zoom one of the stack frame lines in the processes subwindow and the debugger will display the local variables of the stack frame. To close the line, just select the cross again.

```
------------
   PSB: 10B, waiting CV, L: 403230B↑, PC: 2040 (in UserInputsA, G: 425004B↑)
   PSB: 52B*, waiting CV, L: 403410B↑, PC: 362 (in ITFormSW, G: 520170B↑)
   PSB: 56B, waiting CV, L: 404704B↑, PC: 18936 (in TTYSWsA, G: 431674B↑)
     No symbols for L: 404704B↑, PC: 18936 (in TTYSWsA, G: 431674B↑)
       No Variables!
     No symbols for L: 406610B↑, PC: 204 (in TTYImpl, G: 440244B↑)
   PSB: 115B, waiting CV, L: 403240B↑, PC: 3322 (in MailTransfersA, G: 512320B↑)
    .
    .
    .
------------
```

You can also zoom lines in the configs subwindow. Try it. When you zoom a configuration line the debugger will display the nested configurations and modules.

```
------------
   Tajo
     HideIntermediateExpRecs
       PilotKernel
       SpaceCheckPack, G:443024B No symbols.
       MVolumeImpl, G:443070B No symbols.
     Loader
     Base
     XNS
     SubTajo
     SupervisorImpl, G:443230B No symbols.
     TajoControl, G:443354B No symbols.
     VersionImpl, G:443364B No symbols.
   BasicHeadsDLion
     .
     .
     .
------------
```

```
*start*
02152 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting breakpoints
To: NewUsers: ;
```

Like all good debuggers, Sword allows you to set breakpoints so that you can check up on your program while it is running. In the Xerox Development Environment, breakpoints apply to modules that are known within the current context. Thus, you cannot set a breakpoint in your module unless you have first run the program and set the current context so that the debugger knows where to look for your code.

You can set breakpoints either from the Sword form subwindow or via commands in the file subwindow. The items that are listed in the break: field in the form subwindow are all commands that are related to breakpoints.

Through the file subwindow, you can Break Xit procedure (bx), Break Entry procedure (be), Break All Xits module (bax), or Break All Entries module (bae). These commands allow you to set a breakpoint at entry or exit (or both) to a specified procedure, or to set breaks at entry or exit to all procedures in a specified module. You will be prompted either for the procedure name or module name.

If you want to set a breakpoint on a location other than the entry or exit of a procedure, you will have to use the "set" command in the break: field. To set breakpoints you should load your source file into an Empty window, select the line where you would like to set a breakpoint and invoke the "set" command. "set" will set a breakpoint at the closest statement enclosing the selection. (Remember that you cannot do this just yet, as your program must first be loaded so that the debugger knows where to look.)

Also, you shouldn't edit your source file while you are debugging, unless you are ready to recompile it and try again. You don't have to edit the source file to set breakpoints.

Breakpoints are numbered sequentially throughout a debugging session. Breakpoint numbers are never reused within a session: that is, if you set five breakpoints and then remove them all, the next breakpoint that you set will be #6, and not #1 again.

```
*start*
00625 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at breakpoints
To: NewUsers: ;
```

You can use the "list" command in the break: field, or the List Breaks (lb) command to take a look at your currently active breakpoints. This lists the type, the procedure or module name in which it is found (for source breakpoints, the source text is also displayed) for each currently active breakpoint.

The Display Break (db) command takes a breakpoint number as argument and lists the above information for a specific breakpoint.

start

00983 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Conditional breakpoints
To: NewUsers: ;

Breakpoints may be conditional; that is, you can attach a condition so that the breakpoint is only taken when the specified condition holds.

When you execute the "attachCond" command in the break: field, or ATtach Condition (atc) in the file subwindow, you will be prompted for more information. Specifically you will be asked for a breakpoint number and a condition for that breakpoint. (If you don't know the number of the breakpoint that you're interested in, you can find it out with the "list" command.)

You can use any of the relational operators (< , >, =, <=, =>) in conjunction with local variables to specify your conditions. You can also specify the number of times that a break will be bypassed before the debugger is called. For example, "ATtach Condition #: 1, condition: 3".

start
00717 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Clearing breakpoints
To: NewUsers: ;

There are several commands that allow you to remove breakpoints. The "clearall" command in the break: field, or the CLear All Breaks (clab) allows you to remove more than one breakpoint at once. It does what you might expect: Clears ALL breakpoints.

You can also use CLear All Entries (clae) or CLear All Xits (clax) appropriately.

You can clear a specific break with the "clear" command, or the Clear Break (clb) [#] debugger command. You can also CLear Condition (clc), CLear Entry Break (cleb), or Clear Xit Break (clxb).

start
02199 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Display Stack
To: NewUsers: ;

Once you have returned to the debugger, either via a program error or a breakpoint, you can start actually probing around and looking at values within the current context. Display Stack is one of the most common debugger commands; it allows you to look at the runtime state.

Display Stack shows the procedure call stack of the current process. Try typing "ds" in the debugger window now. Notice that the debugger returns with some information (most likely that it doesn't have any symbols for the current module), and then gives another '>'. Type another question mark, and you will notice that you have a different set of commands available. The Display Stack command gets you into a different command mode, where you have a different group of commands available:

g	displays the global variables of the module
j(n)	jumps down the stack n levels
l	lists the source line that invoked the debugger
n	moves to the next frame
b	reverse of n; shows the previous frame
p	displays the formal parameters of the current procedure
q	quits out of Display Stack mode and returns you to the top level of the debugger. (DELETE will also do this.)
r	displays the return values
s	displays the source line that invoked the debugger. Also, loads the source file into a window (if it isn't already), and scrolls to the same source line.
v	displays the frame's variables

Note: the full set of Display Stack subcommands is not always available. For example, when the debugger cannot find a necessary symbol table, the commands that allow you to look at variables and parameters are all disabled. After all, if the debugger can't find the symbols, how can it let you look at them?

Therefore, you will have to wait until the debugging session at the end of this module before you get to practice Display Stack. When you want to try it, you can look back at this message to see the possible options. Remember that you will have to Quit out of Display Stack mode before you will have access to any of the top-level debugger commands.

start
01010 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The interpreter
To: NewUsers: ;

Sword contains an interpreter that allows you to perform operations such as assigning new values to variables, dereferencing, making procedure calls, indexing, field access, displaying variables and types, and simple type conversion.

If you type a question mark in the debugger again, you will see that the first command choice listed is a space (" "). Typing a space gets you into interpreter mode. (You can also type a space to get into the interpreter from within Display Stack mode.)

For example, let the interpreter evaluate an expression for you: try typing " 2 + 6 MOD 4" in the debugger (don't forget the initial space); the debugger will do the calculation and provide the answer.

The rest of this tutorial consists of two practice debugging sessions that will allow you exercise the debugger and its interpreter.

start
01194 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Example: LinkedListImpl
To: NewUsers: ;

The first example is a program called LinkedListImpl, which does not execute properly. To get started, compile the program however you prefer and then run the program by typing "LinkedListImpl" in the Executive.

Type "Help LinkedListImpl" in the Executive to see how the program works, and then type the following into the Executive:

LinkedListImpl 1 2 3 4 5 6 7 8 9

At this point, the program will crash, and a debugger window should pop up. (Note: In 12.3 the debugger window may be created tiny, or might be under some other windows. You may have to look for it.)

To find out what happened, find your debugger window. The debug log should look something like this:

*** Address fault, PSB: 95, at NIL, in PrintResult, L: 606100B↑, PC: 426 (in LinkedListImpl, G: 5105314B↑) ***

An address fault is one of the most common errors that you will run into. Basically, an address fault occurs when a program tries to access an invalid region in memory. This is generally the result of a NIL pointer.

start
02385 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The call stack
To: NewUsers: ;

You should almost always start a debugging session with the Display Stack command, which lets you take a look at the call stack. Execute this command now by typing DS into the debugger window. This will produce a line that looks something like this:

PrintResult, L: 606100B↑, PC: 426 (in LinkedListImpl, G: 5105314B↑) >

This tells you that the error occurred in the procedure named PrintResult which is located in the module LinkedListImpl.

If you continue down the call stack with the next command (just type n), you will see the entire call stack, one module at a time. You may see some lines that start with "No symbols for", such as:

No symbols for L: 613540B↑, PC: 689 (in ExecsA, G: 1153450B↑) >n
No symbols for L: 610674B↑, PC: 137 (in ExecImpl, G: 1153220B↑) >

This means that the debugger cannot find the symbols for the modules ExecsA & ExecImpl. Usually this

can be remedied by retrieving ExecsA.bcd & ExecImpl.bcd to your local disk; once you have those modules the debugger can determine the line of code that was executing when the crash occurred.

However, in general, you should just look down the call stack until you see your own code. There will be times when your code is not on the stack, in which case you will have to work a little harder to figure out what went wrong, but in general you will see your code somewhere on the stack.

In this case, the original cause of the error is in the module LinkedListImpl, so you should go back up the stack until you find the module LinkedListImpl again. Your debug log should then look something like this:

```
*** Address fault, PSB: 95, at NIL, in PrintResult, L: 606100B↑, PC: 426 (in LinkedListImpl, G:
5105314B↑) ***
>Display Stack
PrintResult, L: 606100B↑, PC: 426 (in LinkedListImpl, G: 5105314B↑)  >n
Main, L: 624230B↑, PC: 354 (in LinkedListImpl, G: 5105314B↑)  >n
No symbols for L: 613540B↑, PC: 689 (in ExecsA, G: 1153450B↑)  >n
No symbols for L: 610674B↑, PC: 137 (in ExecImpl, G: 1153220B↑)  >n
No previous frame!  >b
No symbols for L: 613540B↑, PC: 689 (in ExecsA, G: 1153450B↑)  >b
Main, L: 624230B↑, PC: 354 (in LinkedListImpl, G: 5105314B↑)  >b
PrintResult, L: 606100B↑, PC: 426 (in LinkedListImpl, G: 5105314B↑)  >
```

```
*start*
01095 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at the line of code
To: NewUsers: ;
```

Once you have found the correct module on the stack, you can look at the line of code where it died with the Source command or the List command. (Remember, these commands are subcommands within Display Stack mode.) Try doing one of these commands. Your debugger window should look something like this:

```
>Display Stack
PrintResult, L: 606100B↑, PC: 426 (in LinkedListImpl, G: 5105314B↑)  >s  <>TTY.PutNumber[tty, node.num,
[]];  [2099]
```

Thus, the program died while making the call to the procedure TTY.PutNumber. Your next question might be to ask what the procedure TTY.PutNumber is supposed to do. To find this out, you can use the showType! command, as described earlier. Do that now. Essentially what this procedure does is display a number in the specified format to the tty window that it is given.

The next step is to take a look at the values that our program is passing to this procedure.

```
*start*
02092 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Looking at variables
To: NewUsers: ;
```

Once you know the line of code on which the error occurred, the next step is typically to start looking at the values of various variables at that point. To look at the local variables for the procedure, use the v command; to look at the global variables, use the g command, and to look at the procedure parameters (if there are any), use the p command.

Try using these three commands. You should get something like this:

```
>v
h = 610720B↑
tty = Handle(2): 46947  16
  node = NIL

>g
headNode = 4606045B↑
z = ?[4606032B]

>p
h = 610720B↑
```

You can also use the interpreter to look at a specific variable. For example, in this case, you we know that we are interested in a specific node of the linked list (the specific node is referenced in a loop

by the temporary pointer, node) that we are trying to display, so you could look at just that variable by typing a space to enter interpreter mode, and then typing headNode. Your debug log would look like this:

```
> node
node = NIL
```

When you type the name of an identifier (variable, module, etc.) to the debugger, you must capitalize it correctly (i.e., the way that it is declared). In the Mesa Language, gamma, GamMa, and GAMMA are three different variables. Remember, you can also control the format of the output using the debugger's options! command. (See the earlier message entitled options! for details.)

Thus, in this case, we are making a call to TTY.PutNumber, and passing in a number that will be derived by indexing the num field of a record in our linked list. But our pointer to this record, node, is a NIL pointer. Remember, address faults are generally caused by NIL pointers, and you have a NIL pointer on the line of code where the program died, so it is a fairly safe bet that this NIL pointer is what caused the problem.

Thus, the next step is figuring out how you should go about fixing the program.

```
*start*
01202 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Fixing the code
To: NewUsers: ;
```

Look at the line of code directly preceding the line of code where the error occurred. This is where we increment to the next node in our linked list:

```
    node ← node.next;
```

Since we are in a loop at this point, and since our temporary pointer, node, is first initialized to be equal to headNode, we should first operate on the first node (output it to the executive), then change our pointer to the next node as the last statement before we test our loop control statement again. To fix this program, all you have to do is move the above line to be the last line in the loop.

To make this change you should first "abort" this debugging session which will abort the operation that we were trying to do in the Executive. Notice that the Executive displays an appropriate message ("Aborted..."). Now unload this version of LinkedListImpl by typing "Unload LinkedListImpl" in the Executive.

Now you can make the change to the source program, re-compile and re-run the program. This time, it should work just fine.

```
*start*
00927 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Another Example: MiscProcs
To: NewUsers: ;
```

The second debugging example is a "program" called MiscProcs. This program consists of three miscellaneous procedures grouped together in one file: there is no mainline code, so there will be no visible results when you run the program.

Load the file MiscProcs.mesa into a file window, and take a look at the code if you like. (You aren't supposed to be familiar with Mesa syntax yet, but you should be able to get a general idea of what's going on.) There are some global declarations at the beginning of the file, and then four procedures, called Factorial, FreeOldNodes, Interchange, and MakeLinkedList.

You will need to compile and run this program. Not much will happen, since this program does not have any mainline code.

```
*start*
01187 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting the module context
To: NewUsers: ;
```

You should start a local debugging window. Type "cu" in the debugger to take a look at the current context.

Since MiscProcs doesn't execute any mainline code, the current context will not be the context that you are interested in (look at the module name). So, the first thing that you need to do is set the current context. In general, you will have to set the current context when you reach the debugger by world-swapping or when you start a new local debugging session, but not when you reach it via a program error.

To do this, use the SEt Module context ("sem") command, with MiscProcs as argument. You must type the name just as it appears in the internal declaration (MiscProcs: PROGRAM). (I.E., capitalization matters, and the extension should be left off, as it is not part of the internal name.)

(Remember, there are a lot of ways to specify the current context, such as by process number, module name, or configuration name.)

Try CUrrent context again to see the new context.

start
01140 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Making a procedure call from the interpreter
To: NewUsers: ;

You are now going to make a call to the recursive Factorial procedure. As you can see, there is no global code to make a call to Factorial; this module by itself contains no method to make calls to Factorial.

In the debugger window, type " Factorial" (don't forget the space, which invokes the interpreter). This doesn't invoke the procedure; it just displays information about it so that you can be sure that you and the debugger are talking about the same procedure Factorial. (Again, the name of the procedure must match the internal declaration of the procedure.)

Now, to actually make a call to Factorial, type " Factorial[#]", where # is any number between 1 and 12. This number will be used as the procedure argument. The procedure will only accept input in this range; it will return 0 for the factorial of any number not between 1 and 12. When the call has finished the result will be printed in the file subwindow.

start
02126 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Setting breakpoints In Factorial
To: NewUsers: ;

Now suppose that you would like to get a closer look at this recursive procedure. By setting a breakpoint as you exit Factorial, you can watch the recursion unwind. Use the debugger "bx" command to set a breakpoint at the exit of Factorial ("Break Xit procedure: Factorial".) This will be Breakpoint #1.

Call Factorial through the interpreter again (" Factorial[4]"). Probably, at this point, a new debugger window will be created and it should say something like "Break #1 at exit from Factorial, L: 603614B↑, PC: 101 (in MiscProcs, G: 4576670B↑)". In this window type " j" in the interpreter to look at the value of the variable j, which is the value returned by Factorial. (Don't forget the carriage return after the j.) Since this is the first time through, j will be 1, the factorial of 0. Now type p to Proceed (or click on the "proceed" command in the form subwindow) and continue execution. Remember, Proceed just continues execution where you left off.

The second time that you hit your breakpoint, the value of j will still be 1 (the factorial of 1). Use the interpreter to check this. Now try the Display Stack command. This command will tell you that you are in the procedure Factorial, which is in the module MiscProcs. When you are in Display Stack mode, type a question mark to see the commands that are available to you within this mode. Try L, P, R, S, V, and G. When you are through experimenting with the Display Stack subcommands, use Q(uit) or DELETE to return to the upper level command processor.

If you are now convinced that the program works, use Clear All Breaks, or Clear Break #1 to remove the breakpoint. Then Proceed again; the procedure will finish executing and tell you that the factorial of 4 is 24. (Note that the second debugger window goes dormant and the answer to the Factorial function will be displayed in the first debugger window where you made the call from the interpreter.)

start
00957 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North

Subject: Procedure Interchange
To: NewUsers: ;

Take a look at the procedure Interchange in MiscProcs. To see that the array (A) is currently empty, type " A" in the debugger; you will see that the array has 13 elements, and that they are all currently 0.

Suppose that you want to change some of the values in this array. For example, you can set the first element to 7 by typing " A[0] ← 7" (the index of the first element is zero). Change the values of the first three elements of this array to be 7, 8 and 9. The array should now look like this: A = (13)[7, 8, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Now call Interchange, and interchange the values of the first and last elements (" Interchange [0, 12]"). When the call is completed, take another look at the array to verify that your switch has in fact been made.

start
01581 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: MakeLinkedList
To: NewUsers: ;

The last procedure in MiscProcs is called MakeLinkedList. This procedure builds a simple linked list. For example, if you specify a list of 5 elements, headNode will point to an element that contains the letter E; headNode.next to the letter D, and so on. To see the first element, type " headNode↑, to see the second, type " headNode↑.next↑", etc.

You are undoubtedly familiar with linked lists, so we will leave you to experiment with the debugger as much as you like. Set some breakpoints in MakeLinkedList; display the stack; change some values if you like; make a breakpoint conditional.

If you run into trouble, you can always look back at earlier messages in this file, or reference the Sword chapter of the XDE User's Guide. And take heart: all of the material in this tutorial will be covered in the first training unit, and you will have a second chance to learn anything that you have had trouble with or don't remember how to do.

We couldn't possibly cover all that there is to know about Sword and what it can do. This tutorial was meant to get you comfortable with Sword at an introductory level. For more information, refer to the Sword chapter in the XDE User's Guide.

When you are through experimenting, there is one additional tutorial that you might want to look at. This final tutorial, called TeachToolBuilding.nsmail, that discusses how to build new tools for the XDE.

start
00568 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Text manipulation
To: NewUsers: ;

This module covers some of the conveniences available for creating and editing text.

Bring up an Empty window on your screen, and load a file into it to use for the following exercises. Most of the commands discussed in this module are best illustrated when there is a reasonably large amount of text in the window being used. (If you don't have a practice file--make one!)

start
01520 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: FontMonster
To: NewUsers: ;

There is a tool called FontMonster that allows you to control the font of the characters in a given subwindow. However, this tool is a Prototype, and not part of the standard product, so it is not necessarily on your machine. To find out if FontMonster is running on your machine, Chord in the grey bit area. If you are running FontMonster, you should see a FontMonster menu. If you do not see such a menu, you will have to ask someone how to get FontMonster running on your machine.
Bring the menu labelled [FontMonster] to the top of the menu stack. This menu contains a list of all the font files that are stored on your local disk and available for you to use. To change the font of the characters in a file window, select the name of the new font from the [FontMonster] menu; the cursor will turn into a "face". Now click Point over the file window that you wish to change, and the font will change.

Warning: do not use FontMonster on the Table of Contents for the MailTool window. This subwindow is not a standard subwindow, and the characters will not display properly.

You can change the font of most subwindows in the environment. You can also control the "system" font and the menu font separately. You will learn how to set system fonts, menu fonts, and how to control the list of available fonts later in this tutorial.

start
01311 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Text Ops commands: Split
To: NewUsers: ;

All file windows have an associated menu called the TextOps menu, which you can obtain by chording in a file window. The next few messages discuss the commands on this menu, which are used for editing text. You can invoke these commands from the EM symbiote if they are listed there, or you can invoke them from the Text Ops menu.

The Split command is used to divide a region into two subwindows, which can be scrolled separately. This is useful if you want to be able to view two sections of a file simultaneously, or if you would like to maintain your place in a file while looking back at another part of the same file.

Invoke Split in your source window, and notice that a mouse icon appears to ask you for confirmation. Position the cursor at the point where you would like to place your dividing line, and click Point to confirm the Split command. (Note: the subwindows must be each be at least three lines high.)

The dotted dividing line can be moved in the same manner as any dividing line; moving it off the top or bottom of a window will remove it. You can split a window as many times as you like.

start
02266 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: More TextOps commands
To: NewUsers: ;

The Position command is used to specify a particular character by its numerical position in the text. For example, suppose that you run a program (such as the compiler) that tells you that there is something wrong with the 100'th character in your file. To scroll this character to the top of the window, first make sure that the number 100 is typed somewhere on the screen (in digits). If the

number does not already appear in the text, type it in one of the fields in the Find symbiote. Select it. Invoke Position, and the 100th character in the text will be scrolled to the top of your subwindow. This command can use any positive number (including 0) as its argument.

Note: Depending on how your machine is set up, the selected character may be scrolled to the top, middle or bottom of your window. We discuss how to specify top, middle, or bottom later in this tutorial.

J.First positions the first line of text in a file at the top of a subwindow; J.Insert positions the type-in point at the top; J.Select positions the first character of any selected text at the top; J.Last positions the last line of text at the top of the window. Experiment with each of these commands; you will find that they are often more convenient to use than your scrollbar. (You might find "Jump to" a convenient mnemonic device for J.)

The Wrap command controls whether text that is typed without any carriage returns will continue onto the next line or disappear off the screen. Unless you turn Wrap off, a line that has not been terminated by a carriage return will automatically be continued onto the next line. Invoke Wrap again (to turn it off), and type a line of text. Notice what happens when you reach the edge of your screen and keep typing. Now turn Wrap back on by invoking it again. You will probably not want to turn Wrap off very often.

The J. series commands are also available on your keyboard. Refer to your keyboard chart or to page I-18 in the General Tools chapter of your XDE User's Guide to find out which combinations of keys will perform these functions.

start
01863 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Find commands
To: NewUsers: ;

Suppose that you are working on a file in which you have been discussing masa, and someone kindly points out to you that the word should be spelled "mesa". You don't want to have to skim the entire file in order to find each time that you misspelled the word, so you decide to ask the system to do it for you.

One way of doing this is to select one instance of the misspelled word and then press the FIND key on the left side of your keyboard. This will search for the next occurrence of the selected text. When another instance is found, it will video-invert. You can then edit it, and press FIND again in order to find the next instance of the mistake. (You can also use the Find command from the Text Ops menu to perform this operation: the Find command and the FIND key are equivalent.)

One difficulty with this method is that the search for the pattern starts at the instance that you have selected. If you happen to select the first occurrence of something in the file, this is fine; otherwise, you will not "find" the instances that occur early in the file. For example, try selecting the word file in this sentence. Now invoke the Find command using the FIND key. It will find the next instance of the word file that occurs in this tutorial, but will not return to find any instances of it that occurred before this paragraph.

You can, however, hold the SHIFT key down while pressing FIND; this will cause the search to proceed backward from the selected text. Try this. Notice that the FIND key always needs a currently selected argument; in other words, you have to have at least one instance of the desired sequence available in order to use this command.

start
01684 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Edit symbiote
To: NewUsers: ;

The problem with using the FIND key to alter misspellings is that you need to make each of the edits individually. For more advanced editing capabilities, you can use another symbiote, called the Edit Symbiote. The next several lessons discuss the many capabilities of the Edit Symbiote. You should experiment with the various features as they are presented, but you should also realize that you don't need to remember all the details of the editor. You should try to get a general idea of what is available, and familiarize yourself with a basic subset of edit symbiote functionality.

The Edit symbiote is found beneath your EM Symbiote in an Empty window; the Edit symbiote contains the commands All!, S!, RS!, SR!, and R!. There are also two fields in this subwindow, indicated by "←:". The arrow points to the command or commands with which the field is associated. The field on the left is the "find" field; the field on the right is the "replace" field.

Note that if you set a type-in point in an Edit symbiote and press the DOIT key, the symbiote will increase to two lines high instead of just one.

The commands function as follows:

All! lets you change every instance of the string in the find field to the string in the replace field.

S! (Search) searches for the string in the find field.

RS! (Replace/Search) does an R! followed by an S! .

SR! (Search/Replace) does an S! followed by an R!

R! (Replace) replaces the current selection with the text in the replace field.

start
01293 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Search expressions
To: NewUsers: ;

There are several special conventions that you can use in the find field; you are not restricted to just using literals. This message defines the expressions that you can use in this field.

matches any character.
% when used as the first element in a pattern, matches the beginning of a line.

[...] allows you to define a character class. A character class consists of zero or more of the following items:
a literal
a range of literals (a-g)
a negated character class
an escaped character (\c). The two escaped characters that you are most likely to use are \n (carriage return) and \t (tab).

[~...] specifies a negated character class

* "short closure"; this will find the shortest possible match.
** "long closure"; this will find the longest possible match.

As an example, to find a word whose first character is a consonant with a curve in it, whose second letter is a vowel, and whose third letter is between b and r, you could use the following character class:

[bcdgjpqrs][aeiuo][b-r]

To find a word that starts with an upper case letter, you could use:

[A-Z][a-z]**

start
00615 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Replace expressions
To: NewUsers: ;

The Replace expressions are simpler than the Search expressions. Replace expressions are a concatenation of the following:

s	literal
@&	complete match, matches the text found
@n@	partial match, where n = 1,2,3... . n corresponds to the nth element of the regular expression

For example, to delete leading zeroes from numbers:

Find: [~0-9][0]**[0-9]
Replace: @1@@3@
Result: 000000B => 0B, 00343B => 343B

start
01671 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Editor property sheet
To: NewUsers: ;

There is a property sheet associated with the edit symbiote that allows you to control the details of the operation of these commands. This sheet can be found in the window labelled Editor. The forms on this sheet are as follows:

Scope: refers to the scope of the All! function on the edit symbiote. The value {all} will use the entire file as the scope for All!. {rest} will cause the command to start either at the top of the page, or from the insertion point if it is visible. {selection} will restrict the replacement to the current selection.

Interpret match as: determines whether the string in the find field will be interpreted as a pattern or a literal; that is, determines whether characters such as * are interpreted as just characters or as special wildcard characters.

Context of match: determines whether the desired string can occur in the middle of a word or not. "anywhere" means that the pattern can match within a larger word; "words" will only match patterns that are surrounded by non alphanumeric characters.

IgnoreCase When this boolean is on (highlighted), case will be ignored when searching for literals.

Confirm Replace When this boolean is selected, you will be asked to confirm each replacement individually.

Level is discussed in the next message.

The property sheet also has a command subwindow with the commands:

GetDefault! Sets the properties back to a default state.

SetDefault! Lets you specify the default state.

start
01281 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Edit Ops menu
To: NewUsers: ;

When an edit symbiote is attached to a window, an Edit Ops menu is attached to that window as well. This menu contains the same commands as the symbiote, and the additional commands Nest, UnNest, Match and Count. These commands, which operate only on text subwindows, function as follows:

Nest will shift the lines containing the current selection "level" characters to the right, where level is specified in the Editor property sheet.

UnNest will shift the lines containing the current selection level characters to the left.

Match will identify matching parentheses, angle, square, and curly brackets. When one of these character types is selected, Match will extend the selection to the matching character. If you select a character that is not one of these types, the selection will extend in both directions until it contains a match. Successive uses of Match will match larger scopes.

Count will count the number of occurrences of a pattern. The scope and target are specified as in the All! command. The result is given in the message subwindow of the Editor property sheet.

start
01615 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The Edit dictionary
To: NewUsers: ;

The Edit dictionary allows you to create your own set of abbreviations for use in any text window, so that you can simplify the typing of words and phrases that you use often. To get the window for this tool on your screen, press the SHIFT key and the DEF'N key (on the right side of your keyboard)

simultaneously.

The Dictionary tool maintains a dictionary of abbreviation-expansion pairs in a format that permits fast look-up. There is no limit to the number of pairs that you can have in your dictionary. However, when you boot your volume, the pairs stored in your Dictionary will be lost. To protect against this, you can store the dictionary in a file with the extension ".dict". This file will not be destroyed when you have to boot, and you can just load it back into the dictionary tool whenever you like.

Find the Dictionary: field at the far right side of the tool. This field is used to specify the name of the "dictionary" into which you are going to enter your abbreviations. You can fill in this field with any name that you like, provided that it ends with the suffix .dict. You can leave "Default.dict" if you like; this dictionary is currently empty.

You can also maintain several separate dictionaries. For example, you might want one dictionary that contains words that you frequently use when programming, and one dictionary that contains words that you frequently use when creating text files.

start
01403 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Defining an abbreviation
To: NewUsers: ;

To define an abbreviation, you first need to fill in the Expansion: field with the word or phrase that you want to abbreviate. For example, suppose that you type "BEGIN" a lot, and get tired of using the SHIFT key all the time. To circumvent this problem, you can define "b" to stand for "BEGIN".

To do this, fill in "BEGIN" in the expansion field. Now fill in the abbreviation that you would like to stand for the expanded term in the abbreviation field; in this case, "b".

Now invoke Record!. The abbreviation is now recorded in the dictionary. To store it permanently in the .dict file , invoke the Store! command. Now bring up an Empty Window, and type "b", followed by the EXPAND key (on the right side of your keyboard). You can use this key to expand abbreviations while working in any window that allows you to type in text. (Note: the abbreviation must be separated from any previous text by a space. Thus, b will expand to BEGIN, but ab will not expand.)

In addition to defining an abbreviation for a particular word, you can also define an entire phrase. For example, you might wish to define "xde" to stand for "Xerox Development Environment". Abbreviations cannot contain spaces, however.

start
01829 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Automatic abbreviations
To: NewUsers: ;

One difficulty with defining abbreviations in the Dictionary tool is that you add abbreviations at different times, and may make the mistake of defining an abbreviation that you have already defined. This enters the new abbreviation-expansion pair in the dictionary, but destroys the old pair. If you would like to avoid this problem, you can let the dictionary tool avoid duplicate definitions for you.

If you enter a word in both the abbreviation and the expansion field (for example, if you enter "begin" as the abbreviation for "BEGIN"), the dictionary tool will automatically pick the shortest possible abbreviation for that word. For example, it will define "b" as "BEGIN" as long as BEGIN is the only entry beginning with that letter. If you then make another entry that defines "base" as "BASE", the dictionary will require you to type "be" to identify "BEGIN" and "ba" to identify "BASE".

Allowing the dictionary to choose the abbreviation algorithm for you has the advantage that the abbreviation system is consistent and that you can always obtain the correct abbreviation for something; however, you can always choose your own abbreviations if you would like to do it differently. In either case, you can always use the List! command to view the pairs currently in the dictionary. This will help you avoid overwriting an existing abbreviation.

To let the Dictionary tool choose the abbreviations, fill in the entire word or phrase in both fields Abbreviation and Expansion); to choose the abbreviation yourself, enter the full phrase in the Expansion field, and your chosen abbreviation in the Abbreviation field.

start
01104 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)

From: XDE-Training:OSBU North:Xerox
Subject: Other Dictionary Tool commands
To: NewUsers: ;

In addition to the Record! command, there are several other commands available for the Dictionary tool.

Lookup! is used to reference the expansion associated with a particular abbreviation. Try using this command now to look up the abbreviation that you just defined.

List! lists the pairs currently stored in the dictionary database.

Load! loads the specified .dict file into the Dictionary tool database. You will need to load your .dict files each time that you re-initialize (reboot) your volume.

Try defining some words and phrases in your dictionary, changing the name of the dictionary for which you are defining terms, and experimenting with the commands available for the Dictionary tool. For more information on the Dictionary Tool, refer to the Dictionary Tool chapter of your XDE User's Guide. When you are comfortable with this tool, you are ready to move on to the next message.

start
01630 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Your User.cm file
To: NewUsers: ;

Your user.cm is a special file that allows you to personally tailor your workstation environment. For example, this file allows you to specify which tools you want loaded when you first boot, where you want the tiny window for a particular tool to appear, and where you would like active windows to appear. You should have a copy of a user.cm file on your local disk. Load this file into a source window.

The user.cm file consists of a group of separate entries, each headed by a title in brackets, such as [System]. A user.cm file can contain a section for every tool in the environment, as well as sections that apply to an entire logical volume or group of logical volumes. Your user.cm file can be as complex or as simple as you like: you do not even have to have one at all. Remember, though, that the trade-off for a complex user.cm file is that it will take you longer to initialize your system when you boot.

A user.cm file can be organized in any way that you choose; there is no required order for the entries. Some users choose to have the individual sections sorted alphabetically; others group by association, and still others have no system at all. The organization of your user.cm file is entirely up to you.

Whatever the organization of the user.cm file existing on your disk, you should be able to find each of the sections discussed in this tutorial. If you can't find one, create it (just edit it into the file).

start
03726 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: The System entry
To: NewUsers: ;

The [System] section specifies parameters that apply globally to your entire system. It is thus the most general section of your user.cm, and is initially processed when a volume is booted. (Any changes you make to the [System] section will not be applied until you reboot.) A sample [System] entry might be:

```
[System]
User: Glassman
Domain: OSBU North
Organization: Xerox
FileWindow: [x:   0, y: 100, w: 512, h: 321] [x: 300, y: 778] CurrentTasks.txt/t
FileWindow: [x: 511, y:  39, w: 512, h: 769] [x:   1, y:  29] /t
Font: <>Fonts>Gacha12.strike
MenuFont: TimesRoman8.strike
Screen: White
SetPositionBalanceBeam: middle
SearchPath: <>Mailfiles <>Tools <>Fonts <Tajo>
InitialCommand: Run.~ Sword MailTool Print Compiler FontMonster AddHintMenus Calendar
```

The order of entries is up to you.

The Domain:, Organization, and User: entries provide information about the user of the system. You should update these entries in your user.cm. (You will have to ask someone what domain and organization you belong to.)

The FileWindow: entry specifies that upon initialization, a file window will be created of the location and size specified. A window is positioned with four parameters: the x and y coordinates of its upper left corner, and its width and height. The numbers are given in pixels (for example, the 8010 17" display measures 1024 pixels wide by 808 pixels high). The position [x: 0, y: 0] is at the upper left of your screen. Any or all of these four numbers can be defaulted; the default values are [x: 0, y: 0, w: 512, h: 400]. To default all values, specify [].

The second set of numbers in the FileWindow entry, in this case [x: 300, y: 778], determines the tiny position for the window. Finally, you can give a name after the tiny position to specify the name of a file that you would like to have loaded into the file window. This is optional; you can have an Empty window if you like. The switch following the file name can be i(inactive), a(active), or t(tiny); a switch is just a way of "fine-tuning" a command. You don't have to have a switch at all.

You may have as many FileWindow: entries in your [System] section as you like.

Font: and Menufont: determine the font in which characters on your screen will appear. To have your screen displayed in a particular font, you must have a corresponding file on your local disk that contains the information on how to display that font. You will have to ask someone where the font files are stored, and which fonts are available to you. Note that the font file corresponding to the font that you request for your system font must be stored on your root directory (not on a subdirectory) or you must be sure to specify the complete pathname.

SetPositionBalanceBeam: applies to the Position & FIND commands that were discussed earlier; it specifies where the selected character will appear. The possible values here are middle, top, and bottom. If you specify middle, the character will be scrolled to the middle of the screen; top will position the character at the top of the screen, and bottom will position the character at the bottom of the screen.

Screen: specifies the background color for your screen. There are two choices: black and white.

The SearchPath: entry sets the initial search path.

Commands listed in the InitialCommand: line will be run during initialization. An InitialCommand line consists of Run.~ followed by a list of the programs that you wish to run. Remember that a long list of tools will take a correspondingly long time to load.

```
*start*
00766 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: [FileWindow]
To: NewUsers: ;
```

In the FileWindow section, you list the commands that you would like to appear in your EM symbiote, and determine the order in which they are to appear. For example,

```
Menu: Create Destroy Break Save Store Position Reset Split J.First
SetUp: Always Menu Edit
```

Menu: is the list of commands for the EM symbiote; SetUp: determines when they are to appear. Always indicates that you would like symbiotes attached to all file windows. Menu and Edit indicate that you would like both symbiotes. If you wanted only an EM symbiote, and not an Edit symbiote, you could use Always Menu.

```
*start*
00999 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Entries that apply to all tools
To: NewUsers: ;
```

The sections discussed so far have been global sections; that is, they apply to more than one tool and are processed when the volume is booted. You can also have a section for every tool in the environment. Tool-specific sections are processed when the tool is loaded.

This message lists entries that can appear in the section for any tool.

A WindowBox: entry specifies the location that the tool window should occupy when active. A WindowBox: entry requires four parameters: x, y, w, and h. These parameters are the same as the ones for the

FileWindow: entry in the System section, discussed in the message titled [System].

A TinyPlace: entry specifies the x and y coordinates of a tiny window, in pixels.

An InitialState: entry can have one of three values: Active, Inactive, or Tiny.

start
01104 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: Entries for specific tools
To: NewUsers: ;

Almost every tool in the Xerox Development Environment provides possible user.cm entries. The user.cm entries for each tool are documented at the end of the associated chapter in the XDE User's Guide. You should eventually read these sections and customize your user.cm accordingly.

For example, search through your user.cm for the [FontMonster] section. This section lists the fonts that are to appear in the FontMonster menu. To change the items on this menu, you need to retrieve the appropriate font files from the remote directory, and change the entries in this list accordingly. You also need to make sure that the directory where the font files are stored is on the search path when FontMonster is run. (The .strike extension is conventional for fonts; all font files have this extension.)

For more about using FontMonster ask your mentor about getting a copy of the FontMonster documentation.

start
00815 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North:Xerox
Subject: What now?
To: NewUsers: ;

You have finished the tutorials on the basic system. The next tutorial in the sequence discusses the various ways of booting your system, so that you can recover from any problems that you might encounter.

There are two different booting tutorials, depending on the kind of hardware that you have. If you have a 6085, you should read Teach6085Booting.nsmail; if you have an 8010, you should read Teach8010Booting.nsmail. You do not need to read both tutorials. If you don't know which type of machine you have, you will have to ask someone.

Select either Teach6085Booting.nsmail or Teach8010Booting.nsmail from the File: menu now.

start
00725 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Tool building
To: NewUsers: ;

This tutorial provides a brief introduction to the art of building new XDE tools. The next two messages discuss some of the "philosophy" behind XDE tools; the rest of the tutorial covers the mechanics of actually creating a new tool. To do this tutorial, you should be familiar with the compile-bind-run cycle of tool development, but you don't have to be familiar with the Mesa programming language. (If you are not familiar with the compile-bind-run cycle, complete the TeachCompile-Bind-RunWithXXX tutorial before working through this one.)

start
01400 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Subwindows
To: NewUsers: ;

Part of the XDE philosophy is that all tools should have a consistent user interface; this makes life easier for the user as well as for the programmer. The user interface is based on subwindows; there are various standard subwindow types, and most tools have a user interface built from those basic subwindow types. You will learn more about subwindows as you start programming in the XDE; for now, you just need to be familiar with the three most common types: the message subwindow, the form subwindow, and the file subwindow.

A message subwindow is used to post feedback from the program to the user, and a form subwindow is used to simplify parameter collection. A file subwindow is just a text subwindow, and can have various uses. For example, the File Tool has four subwindows: a message subwindow, two form subwindows, and a file subwindow. The first form subwindow lists the various parameters of the tool, and the second form subwindow contains the commands. The lowermost subwindow (the file subwindow) is used for keeping a log of filing operations. (There is no functional reason for having two form subwindows; the tool would work exactly the same way if these two form subwindows were combined into one.)

start
01759 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The FormSWLayoutTool
To: NewUsers: ;

One of the advantages of standard subwindows is that they make it very easy to create a new tool. Basically, to build a tool, you compose it of various subwindows, depending on the kind of functionality that you want. With the exception of a form subwindow, all the standard subwindow types are independent of the particular tool. Thus, a message subwindow in one tool is just like a message subwindow in another tool, and the code to create and manage that subwindow is identical.

This makes it possible to create a new tool interface from an existing one just by changing the layout of the form subwindow. XDE provides a tool called the FormSWLayoutTool (form subwindow layout tool) that supports this approach; it allows you to graphically specify the form subwindow that you want your tool to have, and it then generates code to produce a window with your new form subwindow, a message subwindow, and a file subwindow. Thus, you just "draw" the form subwindow that you want your tool to have, and let the layout tool generate code to create that user interface.

(Note that the FormSWLayout tool is a Prototype, and not part of the standard desktop product.)

The layout tool always generates a window with three subwindows: a message subwindow, a form subwindow, and a file subwindow; the only thing you specify is the format of the form subwindow. However, once the code has been generated, you can edit the code to add or remove subwindows, or reorder the existing ones.

The rest of this tutorial discusses how to use this tool to create your own new tools.

start
01527 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Plagiarize
To: NewUsers: ;

Run the FormSWLayoutTool.bcd from your Executive window.

The FormSWLayoutTool window has three subwindows: a message subwindow, a form subwindow, and a file subwindow.

To use this tool, you "draw" the layout of a new form subwindow in the file subwindow of the layout tool. There are two ways to put a form item on your new form subwindow: you can add them individually or you can "plagiarize" from another form subwindow on your screen.

To see how plagiarizing works, bring up your Command Central window. Now invoke the Plagiarize! command, and then click Point over the form subwindow in Command Central. (The cursor will change into an "eyeball" while you are in plagiarize mode.) When you click Point over the form subwindow that you want to plagiarize, a copy of that subwindow will appear in the bottom subwindow of the FormSWLayoutTool.

Once you have plagiarized a subwindow, you can edit the plagiarized copy using the DELETE, MOVE, STOP, and UNDO keys. MOVE lets you move a selected form item around the file subwindow; DELETE deletes a selected item. UNDO brings back the last form item that you deleted; STOP lets you abort in the middle of a MOVE command. Try using these keys to change the items that you just plagiarized.

When you are through using the editing keys, invoke the Clear! command to clear the bottom subwindow.

```
*start*
02251 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Layout Mode
To: NewUsers: ;
```

You can also create a form subwindow "from scratch" by adding items one at a time. To add a form item to your new form subwindow, you first select the type of item that you want from the FormType: enumeration in the layout tool, and then you enter a tag for it in the Tag: field. (If you aren't familiar with the various form items, such as boolean and enumerated type, check your XDE User's Guide for details.)

For example, suppose that you want build a form subwindow just like the one in the MailTool. (Normally, of course, you would just do this with Plagiarize!, but for now do it one item at a time instead.)

To start off, create the Display! command. To do this, select "command" as the FormType in the layout tool's form subwindow, and enter the word Display in the Tag: field (just the word, not the !). Now move your cursor into the bottom subwindow; notice that the cursor changes form as you do so. Move the cursor around until you find a spot that you like, and then press Point; your new command item will now be in the bottom subwindow. As long as you have a value in the Tag: field, you are in layout mode. Thus, you can continue to move the cursor around in the bottom subwindow and place as many copies of the Display! command as you like.

(Note that you don't have to include the punctuation that normally follows a form item, such as ! or :; the tool deduces the necessary punctuation from the type of the item and adds it automatically.)

Now move your cursor back into the layout tool's form subwindow, and change the value of the Tag: field to Delete, and then move back into the lower subwindow, position the cursor, and click Point. Continue until you have copied each item from the MailTool into your own new form subwindow.

When you are through, or if you wish to edit any of the items that you have already put in the bottom subwindow, remove the item from the Tag: field before doing any editing. As long as there is a value in this field, the tool is in layout mode, and you won't be able to edit the items in the bottom subwindow.

```
*start*
02232 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Enumerated items
To: NewUsers: ;
```

One of the items that you just created is an enumerated item, which means that it should contain a list of all possible values for that entry. However, reasonably enough, the tool is not able to deduce the values that you want your enumerated type to have. Therefore, whenever you create an enumerated item, it is followed by the words {fix the enums}. This should serve as a reminder to you that you need to specify a list of the values that this item can take on.

To do this, remove the value in the Tag: field (if there is one), move your cursor into the lower subwindow, and select your new enumerated type (one click is all it takes). Now press the PROPS key, and a property sheet for that item will appear. This property sheet contains various pieces of

information about the item, most of which refer to the names of variables inside the code. For the most part, you will not have to edit any of the items in this sheet. For enumerated items, however, you need to put the values for your enumerated type in the Choices: field. The values in this list can be single words or quoted strings of multiple words; individual entries are separated by spaces. For example, if you wanted to list your favorite kinds of bagels, your Choices: entry might look like this:

Choices: sesame poppy garlic onion "cinnamon raisin" rye

When you have specified the choices, you also get to specify whether all of the choices will be displayed, or just the one that is currently in effect. If you select the Feedback: {one} option in the property sheet, only the current choice will be displayed. If you choose Feedback: {all}, all of the choices will be dipslayed in the form subwindow, and the choice currently in effect will be highlighted.

When you have set up the property sheet to reflect your enumerated type, invoke Close!.

(Note: there is a property sheet associated with every item that you create. Normally, you will only have to edit the property sheet for your enumerated items, but you can call up a property sheet on any of your form items.)

start
01083 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: FormSWLayoutTool booleans
To: NewUsers: ;

The form subwindow for the layout tool also has several important booleans: AlignX, Usebox, and Anyfont.

AlignX controls the spacing between form items. When AlignX is on, each column will start on multiples of a specific distance from the previous column. Using this boolean will help you to keep the items in your form subwindow cleanly aligned.

Usebox specifies that the generated tool will have the same size window box and screen position as the current size and position of the layout tool. Since the user can always control the size of any tool on the screen, this boolean just allows you to control the initial size of a new tool.

Anyfont causes the layout tool to generate code that will have proportioned space on the form subwindow, regardless of the system font being used. Turning on this boolean is a good idea, since you don't know what system font the user will choose.

start
00800 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Saving a form
To: NewUsers: ;

If you are working on a complicated form, you might want to save an intermediate version of it and later retrieve it. (If you crash your machine without saving the form, you will lose the items in the bottom subwindow of the layout tool.) Also, it is easier to add features later if you have the original form to start from.

To save a form, enter the name of your tool in the Root: field and then invoke Save!. For example, if you invoke Save! with the value SimpleTool, the layout tool will create a file named SimpleTool.by. You can then later use the Load! commmand to load this file back in and continue editing.

start
00668 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: SetDefaults!
To: NewUsers: ;

The SetDefaults! command allows you to set various default characteristics for your form items. Most of these characteristics correspond to those in the property sheet for a form item; you don't have to worry about these defaults if you don't want to. Later, when you become more familiar with the code that the tool generates, you may want to modify some of these defaults. Try SetDefaults! now so that you have an idea of the kind of default values that this command allows you to set.

start

01969 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Creating a simple tool from scratch
To: NewUsers: ;

To illustrate how this all works, you are now going to do a very simple tool from scratch. This tool performs operations such as blinking the display and "beeping" the sound generator.

The first step is to use the FormSWLayout tool to create the user interface for this tool. You will need three command items (Beep, SetBackground, and Blink), two numbers (Frequency and Duration) and one enumerated type (Background: {white, black}). Set up the form subwindow any way you like, but be sure that you include these six items. You should also make sure that you turn on the AnyFont boolean.

Frequency and Duration are the arguments to the Beep command. Frequency specifies the frequency of the beep in hertz; this value must be a positive number. Duration specifies the length of time that the beep should sound; this value is specified in milliseconds.

Background is the argument to SetBackground; white and black are the two possible colors for the background. Note that you will have to edit this item to fix the choices. Make sure that you put White and then Black rather than the other way around. (This matters only because you will be inserting some code that has already been written, and assumes that the order of items is white, black.)

When you have the tool laid out, change the name in the Root: field to be the name that you want your tool to have, and then invoke the Doit! command. This command will create a .mesa file, that has as its base name the value that you put in the root field. For example, if you put DumbTool in the root field, the FormSWLayoutTool would generate a file called DumbTool.mesa

Once you have invoked Doit!, bring up an Empty Window and load your new .mesa file into the window. This is the source file for your tool.

start
00989 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Running the skeleton
To: NewUsers: ;

The code in this file will compile as is. Compile and run this program in the XDE environment with whatever testing methods you have learned so far.

When you run the tool it won't do anything, of course; it will just present the user interface. You should try changing the values of the Duration and Frequency fields to see if it affects your subwindow at all. If you have the items too close together, putting a new value in for a number or string may obscure some of the keyword for the next item. If this happens, just redo the subwindow so that the items are farther apart.

When you are ready to continue with the tutorial, Unload the tool from the Executive, by typing "Unload DumbTool". (Remember to press SHIFT-STOP to return to the debugger if you are testing it in a separate environment).

start
01773 00071 UU
@00045 01536 ffffffffffffffffffffffffffffff
@Date: 5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: The code
To: NewUsers: ;

As you saw by executing the program, the code that the FormSWLayoutTool generates is just the skeleton of a tool. It creates the user interface, but does not provide any of the actual functionality of the tool. Once you have created a user interface for your tool, you still have to add the code that performs the actual operations.

Take a look at the code in your source file. To make this program work, you don't need to understand how this code works; you just need to be able to make a couple of minor changes.

Somewhere near the top of the file you should find "templates" for the commands that you put in your form subwindow. These "templates" are just procedure declarations without any code. They will look like this (You can do a FIND on them):

```
  BeepInternal: PROCEDURE = {
    ENABLE ABORTED => {Done[]; CONTINUE};
    Write["Beep called\n"L];
```

```
    Done[] };

  SetBackgroundInternal: PROCEDURE = {
    ENABLE ABORTED => {Done[]; CONTINUE};
    Write["SetBackground called\n"L];
    Done[] };

  BlinkInternal: PROCEDURE = {
    ENABLE ABORTED => {Done[]; CONTINUE};
    Write["Blink called\n"L];
    Done[] };
```

Delete these templates, and insert the following code:

```
  BeepInternal: PROCEDURE = {
    ENABLE ABORTED => {Done[]; CONTINUE};
    UserTerminal.Beep[data.frequency, data.duration];
    Done[] };

  SetBackgroundInternal: PROCEDURE = {
    ENABLE ABORTED => {Done[]; CONTINUE};
    [] ← UserTerminal.SetBackground[data.background];
    Done[] };

  BlinkInternal: PROCEDURE = {
    ENABLE ABORTED => {Done[]; CONTINUE};
    UserTerminal.BlinkDisplay[];
    Done[] };
```

```
*start*
00930 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Editing the Directory list
To: NewUsers: ;
```

The final change that you need to make is to edit the DIRECTORY and IMPORTS lists at the top of your file. Basically, these lists specify that this module is calling procedures that are found in other modules. You will learn lots more about these lists as you start programming in Mesa. Now, however, just add the word UserTerminal to both of these lists. (Capitalization is important, and make sure that each word on the list is separated by a comma as shown.) The beginning of your program should now look like this;

```
DIRECTORY
  Exec,
  Format,
  FormSW,
  Heap,
  Process,
  Put,
  TextSource,
  Tool,
  ToolWindow,
  UserTerminal,
  Window;

DumbTool: MONITOR
  IMPORTS
    Exec, FormSW, Heap, Process, Put, Tool, UserTerminal = {
```

```
*start*
00743 00071 UU
@00045 01536 ffffffffffffffffffffffffffffffff
@Date:  5 Apr 88 13:37:09 PDT (Tuesday)
From: XDE-Training:OSBU North
Subject: Running the tool
To: NewUsers: ;
```

The tool is now ready to go. Compile and run it again. Experiment with different values. Remember, the values for frequency should range from 50 to about 20000 (depending on how good your hearing is), and the duration parameter is in milliseconds. (10 might be a good value to start with.)

This is the end of the tutorials; you are now ready to go on to the next part of your Mesa training, whatever that might be. If there are aspects of the FormSWLayoutTool that you don't feel comfortable

with, you should experiment with it until you are sure of yourself.

COURIER

A Quarterly Newsletter from the Xerox Systems Institute

Volume 2, Issue 1 March 1987

XEROX SYSTEMS INSTITUTE RELEASES NEW INTERPRESS TOOLS

A much broader set of applications can now work with Interpress printers through the use of newly-released tools from the Xerox Systems Institute. These new tools will assist users of programs that emit Calcomp calls, ASCII files, device-independent Troff, and even MacPaint files.

ManPlot, a new tool, is a library of routines which enables DEC Vax users running either VMS or Unix to print graphics from certain applications programs on Xerox Interpress™ printers. This software package converts Calcomp calls to Interpress commands and works with applications that support Calcomp plotting routines. The ManPlot run-time library is simply linked with existing Fortran or NCAR plotting programs. Using a menu-driven interface, the program allows you to interactively route your output to Xerox laser printers or any one of a wide range of terminals and plotters via an included driver package. Both landscape and portrait modes are supported. This software package is provided on 9-track, 1600 bpi tape in either VMS "Backup" or Unix "Tar" format.

An updated version of the Interpress toolkit has also just been released. In addition to providing converters from ASCII and from device-independent Troff to Interpress, the enhanced toolkit includes among its new features:

- A converter from Unix Plot format to Interpress
- A set of utilities to manipulate files in the Xerox Raster Encoding Standard (RES)™, including a converter between Apple MacPaint and RES
- A font metrics converter program to convert from the Xerox Font Interchange Standard Format (FIS) to that of the specific composing system

ManPlot and the Interpress Tool Kit are unsupported software packages available from the Xerox Systems Institute for a nominal fee. Documentation and examples are included.

WHAT'S INSIDE

UPCOMING INTEGRATION WORKSHOP TO FOCUS ON PUBLISHING; UNIX/XNS INTEGRATION WORKSHOP WELL-ATTENDED

The Xerox Systems Institute is planning a new integration workshop this June 23 and 24 in Boston. The workshop theme will be interoperability of equipment from major vendors in publishing applications, from the PC to the mainframe. Topics will include integration of Xerox publishing products with equipment from major vendors, integration of IBM AFP and Xerox Interpress architectures, and the relation of Interpress to other page description languages.

The cost of the two-day workshop will be $195. Agendas and more details will be sent to *Courier* subscribers shortly. For additional details or to register, please contact the Xerox Systems Institute.

As part of an on-going effort to share information regarding new XNS™ capabilities within Berkeley 4.3 Unix, the Xerox Systems Institute recently held a workshop focusing on developments in integrating the two environments. Over 90 people from 32 companies attended the two-day conference.

Attendance at the December 4 and 5 conference in Sunnyvale, CA, represented a range of manufacturers, end-users, institutions, and government agencies. The majority fell into the first two classifications. Highlights included an open discussion from participants regarding their work or interest in Unix/XNS integration, a report from Fuji-Xerox on their implementation of XNS on a Unix-based system, an update on the Interpress toolkit, and a demonstration of current XNS capabilities under Unix 4.3.

Of particular note were the number of people interested in XNS under AT&T SystemV.3. This prompted the establishment of a System V.3 "interest group" and 22 people signed up. Xerox plans to work with AT&T to establish support for XNS under System V.3. Members of the interest group will be updated through regular mailings and *Courier*. To obtain the minutes of this workshop, contact the Xerox Systems Institute.

EDITOR'S NOTES

This first issue of *Courier* for 1987 will help set a new tone for the newsletter and the Xerox Systems Institute. Recently, Interpress and XNS marketing and support have been merged under the XSI umbrella. While still retaining our old functions, including strong commitment to Interpress and XNS implementors, the XSI will begin a facilitator's role in support of multi-vendor integration. A great deal of Xerox success depends on the ability to integrate with a wide range of vendors and part of our charter is to assist with this effort.

Courier will continue to provide practical information to help users, implementors, and systems integrators take advantage of the latest information regarding XNS/Interpress implementation and integration. If there are additional topics we should cover or if you would like to share your views, write or call.

Bruce Schatzman, Editor
(408) 737-4653

Schatzman:Osbu North:Xerox

NEW WORKSHOPS AVAILABLE FOR INTERPRESS IMPLEMENTORS

Xerox recently announced a new schedule of training courses for implementors of the Interpress Page and Document Description Language. This three day technical workshop describes the Interpress language and its implementation for document creators, software vendors, and printer manufacturers. The class is held either at Xerox, 880 Apollo Street, El Segundo, California, or at the Xerox Wilson Technology Center, 800 Phillips Road, Webster, New York. Cost is $500 for first individual from a company, $350 for each additional individual. For specific dates and locations, see the "Calendar of Upcoming Events" in this issue.

XEROX DEMONSTRATES X.400 SOFTWARE AT HANOVER FAIR

A pre-release version of the Xerox X.400 Mail Gateway was demonstrated March 4-11 at the CeBIT (Welt-Centrum Büro Information und Telekommunikation) exhibition of the annual Hanover Fair in Hanover, West Germany. The software was shown with X.400 products and services offered by 13 other companies from all over the world. Participants in the X.400 exhibition at CeBIT '87 include British Telecom, Bull (France), Data General, the Deutsche Bundespost (German PTT), Digital Equipment Corp., Hewlett Packard, ICL (United Kingdom), NTT (Japan), Nixdorf Computer (Germany), Olivetti (Italy), Philips (Holland), Siemens (Germany), Sydney Development Corp. (Canada), and Xerox. All 14 participants shared a single booth at the fair, and demonstrated the ability to exchange electronic mail among diverse computer systems and public communications services.

For demonstration purposes, Xerox was grouped with Philips and NTT. In a mock "parts ordering" scenario, the demonstration consisted of one of the partners sending a mail message requesting information on several items from one of the other partners. The second partner then forwarded the message to the third partner and requested confirmation of stock availablity for the order. Finally, the third partner responded to both of the others with a quote on price and availability of the items requested. Messages were also exchanged among all of the participants on an ad-hoc basis. Xerox used a 6085™ professional workstation and an XNS server running the X.400 mail gateway to transmit and receive the messages.

ETHERNET INSTALLATIONS ON THE RISE

Market research firms keep trying to estimate the worldwide number of Ethernet networks. No one knows for sure but the last count yielded 60,000 networks supporting a total of about 500,000 devices. This number is expected to increase substantially as a result of IBM's announced Ethernet adaptor for their PC RT. The vendor having the largest installed base of Ethernet networks and devices is Digital Equipment Corporation with about 50,000 attached devices.

ORDERING INFORMATION

To obtain any of the Xerox Systems Institute services described in this newsletter, including documentation, standards specifications, software tools, training, or conference registration, contact Pam Cance, Xerox Systems Institute, 475 Oakmead Parkway, Sunnyvale, CA 94086. (408) 737-4652.

XEROX INTERPRESS IMPLEMENTATIONS CONTINUE TO INCREASE

In 1986, Xerox Corporation introduced a number of new products that supported the Interpress page and document description language. This included both publishing and printing products.

On the publishing side, Xerox introduced the Documenter workstation, the XPS 701™ publishing system, the 2285™ engineering workstation, the Xerox Desktop Publishing Series: Ventura Publisher Edition™, and many others. Xerox currently supports Interpress on over 50 products.

Details on Interpress status for Xerox printers is shown in the table below.

Product	Market Notes	Interpress Level	Availability	Connections Supported		
				Ether -net	Mag tape	Channel connect
9790 EPS™	120 ppm, 300 dpi	Commercial Set	1H, 1987	X	X	X
9700 EPS®	120 ppm, 300 dpi	Commercial Set	Now	X	X*	
8790 EPS™	70 ppm, 300 dpi	Commercial Set	1H, 1987	X	X	X
8700 EPS®	70 ppm, 300 dpi	Commercial Set	Now	X	X*	
4050 EPS™	50 ppm, 300 dpi	Commercial Set	Version 1 - Now Version 2 - 1H, 87	X X	X* X	X
3700™ Laser Printer	24 ppm, 300 dpi	Commercial Set	Now	X		
8040™ Series	12 ppm, 300 dpi	Publication Set #	Now	X		
8000 Laser CP™	10 ppm, 300 dpi	Publication Set #	Now	X		
495-1™ Fax	2 ppm, 200 dpi, networked Fax	Publication Set #	Now	X		
Formatting Print Service™	Converts vectors to Commercial Set	Publication Set #	Now	X		
4045 CP™	10 ppm, 300 dpi	Professional Graphics	To be announced			

Notes:

* Magnetic tape access for Interpress on current 4050, 8700, and 9700 requires use of a workaround which impacts performance.

\# These printers are targeted for Publication Set, but currently do not implement certain Publication Set features concerning sampled graphics and curve vectors. They are in the process of being brought into compliance. For a detailed implementation chart, please contact the Xerox Systems Institute.

XNS AND UNIX - NEW OPPORTUNITIES IN SYSTEM INTEGRATION

Bruce D. Schatzman, Xerox Corporation

When Xerox placed its Xerox Network Systems (XNS) protocols in the public domain in 1981, its goal was to increase the number of vendors able to communicate with Xerox products. Recently this goal received a substantial boost when Xerox and the University of California at Berkeley jointly announced that XNS protocols had been incorporated within the latest release of Berkeley Unix (often called Unix 4.3 or BSD 4.3).

Unix 4.3 users and OEMS's can now exchange information with Xerox products and take full advantage of sophisticated document processing, publishing, and printing solutions. Xerox workstations, printers, and servers are simply connected to existing Unix 4.3 networks using standard network hardware and software - there are no special options to purchase. Unix users are able to exchange files with Xerox servers, login to Xerox servers (for system administration), convert Unix documents to Interpress format, and send Unix documents to Xerox printers. Xerox 6085 and 8010™ workstation users are able to exchange documents with Unix-based hosts and access a wealth of development tools and applications on Unix machines over Ethernet.

What Is Included

Unix 4.3 includes a subset of XNS protocols, user-level tools which act as the interface to these protocols, and a set of development tools for both XNS and Interpress.

Network Protocols

The implementation of XNS under Unix 4.3 includes the following protocols:

- *Printing Client* - Supports sending Interpress masters to Xerox print servers.
- *Filing Subset Client* - Allows file exchange between Unix devices and 6085's.
- *Gateway Access Protocol (Virtual Terminal) Client* - Provides for remote login to a Unix device through a terminal emulation window on a 6085.
- *Gateway Access Protocol (Virtual Terminal) Service* - Provides for Unix host login to devices or servers supporting the GAP protocol.
- *Time Protocol* - Maintains consistent time in a distributed network environment
- *Clearinghouse/Authentication Client* - Enables a Unix device to access the Clearinghouse database for network resources and their locations, and verifies user credentials.
- *Bulk Data Protocol* - Provides efficient transfer of large quantities of data
- *Courier Protocol* - Provides the remote procedure calling and data presentation functions for all of the above protocols.
- *Internet Transport Protocols* - Provides for low-level communications functions such as internet datagram creation, error handling, and packet sequencing.
- *Ethernet Protocols* - For packet transmission/reception on the communications medium.

Network Functionality

Unix 4.3 implements a *subset* of XNS. Mail, for example, is not implemented. While Unix-based machines do not have identical communications capabilities as 6085's or other full XNS clients, the capability provided is fairly rich and there are many associated benefits:

6085 Login to Unix Machines

By creating a VT-100 emulation window on a 6085 running Viewpoint™ or XDE™, the user may login to any Unix 4.3 host over the network. This provides access to a wide variety of host application

software. The user then has the dual power of Viewpoint for document creation/editing and a variety of Unix applications such as database management, spreadsheets, and design programs that can be run in 6085 terminal emulation windows. Using the "makescreen" and "makedocument" functions, 6085 users can capture Unix host information and store them in Viewpoint documents for editing. By opening several different emulation windows on a 6085 screen, multiple Unix applications may be run at once. Data is transferred directly over the network at Ethernet speeds.

Unix Login to Xerox Servers

The Unix 4.3 Gateway Access Protocol service allows Unix users to login to Xerox servers over the network for the purposes of remote system administration. Unix users can thus manage file drawer access rights, passwords, distribution list management, etc without having access to a 6085.

File Storage and Retreival

An XNS filing subset client is supported under Unix 4.3 which enables 6085 file transfer to and from the Unix host. 6085 users can thus take advantage of additional file storage capability offered on the Unix host to store and retrieve Viewpoint (or other files). Since no filing server capability exists on the Unix side, a Xerox file server is necessary to act as an intermediary between the workstation and the Unix host. Currently, files are not directly transferred from the 6085 to the Unix system but are accessed on the file server by both devices. In the near future, Xerox plans to make available a new enhancement which enables direct file transfer (see article by Ed Flint on p. 8).

Printing

Unix users can print various files on Xerox (or other) Interpress printers by utilizing the Interpress toolkit, a software package that enables conversion of Unix files in several popular formats to Interpress format. This includes ordinary ASCII files (such as program source code, data files, mailing lists, etc) and device-independent Troff. By invoking the "xnsprint" command, these masters can then be sent to an Interpress printer anywhere on the network. Not included within Unix 4.3, but available through Berkeley, is a DVI to Interpress conversion program. DVI is an output format used by popular Unix text processors such as TeX.

Tools

The two primary tools included with Unix 4.3 are the Interpress toolkit and the Courier™ compiler. The Interpress toolkit assists application developers who wish to send output to Interpress printers. It provides both C-callable and user-level programs for the creation and manipulation of Interpress masters. The Courier compiler is an XNS development tool which takes programs written in the high-level Courier specification language and compiles them to C code.

The Command Interface

The user interface to the XNS software within Unix 4.3 is quite simple and easy to use. The three most frequently used commands are: xnsftp (for file transfer), xnsprint (for printing Unix documents on Interpress printers), and gaptelnet (for remote login and virtual terminal services). Unix users can run these programs either directly from their host machine or through a terminal emulation window on a 6085.

Special Considerations

While Unix 4.3 XNS offers network integration previously unavilable, it is important to understand the limitations. Since there is no gateway from XNS Mail to TCP/IP Mail, and since a mail client is not implemented under Unix 4.3, it is still not possible to exchange mail with Unix systems from Viewpoint. The Xerox Development Environment does, however, support TCP/IP protocols, (including SMTP) and it is possible to exchange mail this way.

There is no editable document exchange other than ASCII. Any Unix file may be incorporated into a Viewpoint document as long as it is first converted to ASCII format. Application-specific

graphics or control characters must be stripped out. There are no tools which convert DiTroff or TeX documents to Viewpoint but these documents may be converted to printer-ready Interpress format.

Although the software has been in use now for almost one year and is considered reliable, it is not currently a supported product of Xerox Corporation and Xerox provides no guarantees of reliability or performance.

An Example

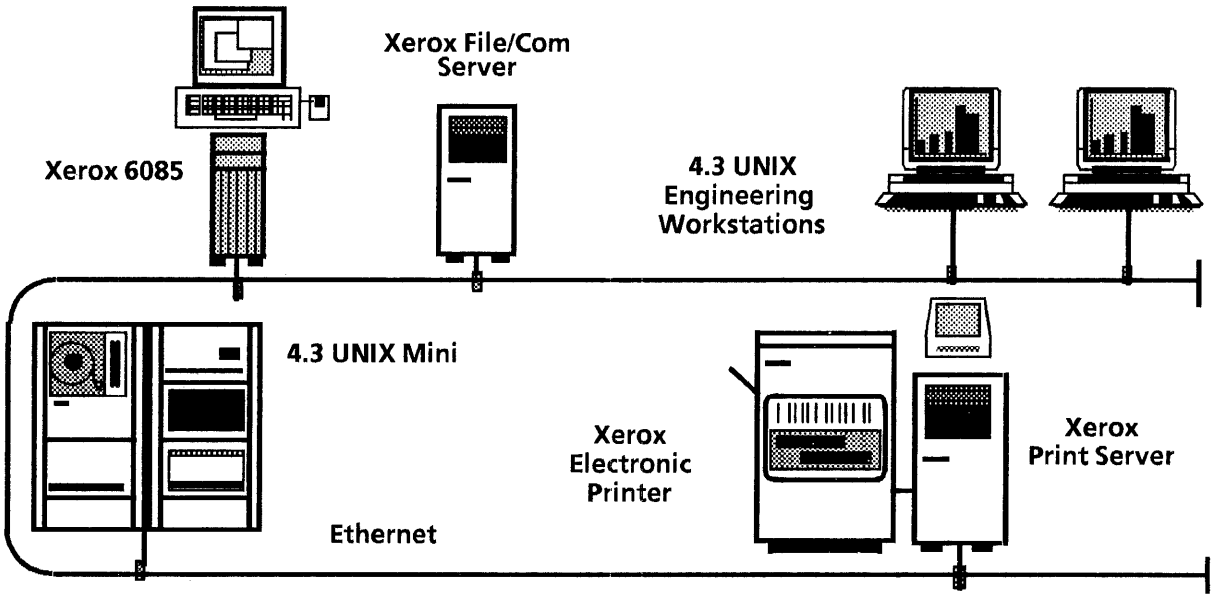


Figure 1. Unix 4.3 might be used in an engineering environment to connect Unix workstations and minicomputers with such Xerox products as the 6085 workstation with Viewpoint software, Xerox file services, and Xerox Interpress printers.

Figure 1 shows an integrated system in an engineering environment. Here, engineers use a combination of Unix 4.3 minicomputers and workstations to develop and run simulation software for the government. They can print simulation results on a line printer but until now have lacked the ability to prepare publication-quality reports to their customer, the Department of Defense. A Xerox 6085, file server, print server, and laser printer can provide the following benefits:

- Direct printing of source code, simulation data file, and other ASCII text output on a high-quality Xerox laser printer from their Unix host.
- UNIX text processor (Ditroff, TeX, etc.) output on Xerox laser printers.
- Host application support on the 6085 using terminal emulation windows from either XDE or Viewpoint.
- "Makedocument" and "makescreen" functions to capture emulation window text (such as simulation data) and place into editable Viewpoint documents.
- Once captured data is placed in a file, Viewpoint Data Capture may be used to convert unstructured data into tables for bar charts, records processing, or spreadsheets.
- File storage and retrieval to/from the Unix host.

XEROX ENHANCES UNIX/XNS INTEGRATION CAPABILITIES

Ed Flint, Xerox Webster Research Center

Xerox has recently extended XNS capabilities within the Berkeley Unix 4.3BSD operating system. The extensions significantly increase the existing functionality between Unix 4.3 hosts and Xerox products. While not distributed by Berkeley, Xerox plans to make this software available through the Xerox Systems Institute in the near future. Among the new Unix/XNS features are:

- FilingSubset Protocol server capability on the Unix host
- An extended Filing and FilingSubset client
- Storage and retrieval of Viewpoint files, including Viewpoint documents, books, Interpress masters, canvasses (RES files), spreadsheets, records files, applications, and fonts without loss of functionality
- Directory archive/restore from a remote file service to a magnetic tape or disk on a Unix 4.3 machine
- Direct viewing of simple text files on a remote Xerox file server from a Unix host
- Printing of simple text files and Interpress masters directly from a remote file server
- Improved support for Authentication lookup and credentials checking

The new file server capability is of particular importance, enabling XNS-based products to store and retrieve Viewpoint desktop files directly with Unix hosts. Previously, it was necessary to use a Xerox file server to accomplish file exchange. This new server capability is consistent with the FilingSubset Implementor's Guide (available from the Xerox Systems Institute) and provides support for Filing defined procedures and extended attribute types except random access and controls manipulation.

The new filing client (xnsftp) includes support for:

- Exchange of Viewpoint related files with Xerox file services
- Serialization and deserialization of file service directories
- Copy, move and rename functions
- Choice of either the FilingSubset or Filing Protocols
- An extended command line interface for non-terminal users (similar to XDE FTP).

The remote file viewer (*xnsbrowse*) allows Unix users to "browse" through remote text files. The file is displayed on the user's terminal or through a Unix page filter such as *more*.

The remote file printer (*xnsrprint*) prints text files or Interpress masters stored on a remote file server without creating the file locally. Text files are processed through *maha* (from the Interpress toolkit) to create an Interpress master which is sent to an appropriate Interpress printer. Remote Interpress masters are delivered directly to the printer via *xnsprint*.

Users may save their XNS credentials in Unix environment variables by invoking a new feature called *xnscreds*. This program verifies the credentials and makes them available to subsequent tools through the user's login environment, thereby reducing the need for querying on individual invocations. Several new facilities for Authentication service lookup and credentials verification are also newly available as part of the Courier run-time library.

CALENDAR OF UPCOMING EVENTS

Interpress Implementors Workshop, March 17 thru 19, Los Angeles, CA. A 3-day workshop for implementors of Interpress on workstations or software programs. Contact Xerox Systems Institute, (408) 737-4652.

European XPLOR, May 11 thru 14, London, England. Electronic printing users group. Contact XPLOR International, (213) 373-3633.

Interpress Implementors Workshop, May 12 thru 14, Los Angeles, CA. A 3-day workshop for implementors of Interpress on workstations or software programs. Contact Xerox Systems Institute, (408) 737-4652.

EDGE, May 31 thru June 4, Orlando, FLA. Ethernet/XNS users group. Contact EDGE Corporate Office (206) 251-6010.

Interpress Implementors Workshop, June 16 thru 18, Webster, NY. A 3-day workshop for implementors of Interpress on workstations or software programs. Contact Xerox Systems Institute, (408) 737-4652.

Xerox Systems Institute Integration Conference, June 23 thru 24, Boston, MA. Presentations and seminars describing ways for vendors and end-users to integrate Xerox publishing and document processing solutions into their systems architecture. Contact Xerox Systems Institute, (408) 737-4652.

XEROX AND INDUSTRY STANDARDS - AN UPDATE

Abhay Bhushan, Xerox Corporation

The Open System Interconnect (OSI) architecture, developed within the international standards organizations ISO and CCITT, represents the best vehicle for integrating products from different vendors. Xerox is committed to satisfying this critical customer requirement.

To help make OSI a reality, Xerox joined with other companies in 1986 to establish the Corporation for Open Systems (COS). Robert Adams, President of the Xerox Custom Systems Division, is a member of the COS Executive Committee, and Jerry Elkind, Vice-President for Systems Integration, chairs the COS Strategy Forum Steering Committee. Xerox is represented on the COS Architecture Committee, on its Message Handling Systems (MHS), File Transfer Access and Management (FTAM), and Test Architecture committees, and chairs the newly formed Document Architecture subcommittee. Xerox is also involved in the NBS OSI workshops, including the Special Interest Groups (SIG) for X.400 MHS, FTAM, Directory, and ODA/ODIF (Office Document Architecture/Document Interchange Format). Xerox furthermore supports the MAP and TOP efforts, and is contributing to their development.

Because of the Xerox pioneering work in the fields of local area networking and internetwork communications, it has been closely involved with standards efforts in this area. Xerox is active in the IEEE 802 local area network standards, the IEC TC-83 fiber optics work, and the ISO and CCITT committees. In the applications area, Xerox is doing extensive work on developing the OSI architecture for distributed services, and the Message Handling, Directory, Printing, and Filing Services standards, being active in ECMA TC 32, ANSI X3V1 and X3T5, CCITT COM VII, and ISO SC 18 and SC 21 subcommittees.

In printing standards, Xerox is contributing to the development of an international page description language standard and to font standards by participating in the respective ECMA, ANSI and ISO work in this area. Xerox is also influencing the development of multilingual character encoding standards through its work in ANSI X3L2 and ISO SC 2. In the area of document architecture, Xerox is working with other vendors to develop the ODA standards. ODA is finding increased acceptance among vendors and users alike, and Xerox is participating in all of the committees and working groups responsible for ODA development in ECMA, ANSI, CCITT, and ISO.

Xerox will continue to work cooperatively with others to develop capable industry standards, and products that comply with those standards. XNS protocols, including Ethernet and the Interpress Printing Architecture standards provide a solid foundation and a ready solution for document processing, publishing, and printing applications. As OSI protocols become stable and implementable, Xerox plans to integrate them into its products while continuing to provide the functionality and performance of XNS and Interpress to its customers.

XDE - THE SYSTEM INTEGRATOR'S TOOLKIT

Dave Nelson, Xerox Corporation

The Xerox Development Environment (XDE) is a complete programming desktop environment for the systems or applications programmer who wishes to develop custom software for the Xerox 6085 Professional Computer System and the 8010 Information System. XDE contains a wealth of development tools for system integration and Viewpoint (the Xerox integrated office information and publishing package) applications development. It can also be used to create interactive applications programs (called *tools*) for XDE itself.

Software houses that would like to take advantage of the wide range of Xerox publishing solutions and network services will find XDE useful for integrating their own products with Xerox hardware and software. As a system integrator's tool, XDE provides both ready-made applications and development software for linking Xerox products to a variety of other systems such as Unix. Communication gateways, network protocol and format conversions, and host processor interfaces may all be developed with XDE. Whether the intention is to send documents to Xerox printers for high-quality output or to integrate ViewPoint software with vertical and horizontal market applications for business, publishing, or engineering, XDE is an indispensible tool.

XDE offers a variety of applications for source code preparation, network project management, and host interfaces. These tools can dramatically increase programmer productivity, even for complex system development applications, such as those comprising millions of lines of source code, thousands of compilations, and programmer teams of 100 or more at geographically distributed sites.

An XDE Application Example

Many useful tools and applications have been developed to run in XDE environments. A number of these are used to enhance system interconnect and compatibility. One such tool was developed by Jack Callahan, at the University of Maryland's Heterogeneous Systems Lab, Computer Science Department. Creatively named "Norman Mailer", this program is a multiple protocol mail reading/composing tool that speaks to both the Unix and XNS electronic mail systems. Incoming messages are received from all designated mail systems and merged for presentation to the user. Outgoing messages are replicated as needed and sent to the respective mail systems at the tool level. Destination addresses are determined from a parse of the form of the recipient address.

Norman uses the remote procedure call (RPC) model for network services, and consists of two programs: (1) a mail reading/composing tool running in XDE, and (2) a server program running on Unix 4.3 BSD. The mail reading/composing tool is a modified version of the XDE MailTool. The server

program is written in C and implements a subset of the Xerox RPC mail protocol and uses XNS as the low level network transport mechanism.

Norman Mailer is part of the Utilities and Prototypes that can be ordered with XDE. If you want to know more, you can contact your Xerox sales representative or: Jack Callahan, Department of Computer Science, University of Maryland, College Park, Maryland 20742

The XDE Newsletter

XDExpress is a quarterly newsletter published by Technical Services to keep you informed about the Xerox Development Environment (XDE). *XDExpress* covers many topics such as programming tips, new products, University Grant Program highlights, and training information. *XDExpress* is sent to all XDE users and interested parties, either electronically or through the US mail.

The format accommodates the electronic mail reader. A TOPIC list appears at the beginning of each edition to let you search for topics of interest. Content focuses on what you, our customers, told us you wanted to hear about.

If you wish to be added to the distribution of *XDExpress*, just fill in the information requested below.

XDExpress

Please add me to the XDExpress mailing list.
Name: _____
Address: _____
_____
_____
_____

Return to:

Patience Nason
Xerox Corporation
475 Oakmead Parkway
Sunnyvale, California 94086

Email Address: _____
_____

XDE Highlights

- A highly integrated set of system programming tools and electronic desktop for the programmer.
- Application development tools for adding custom applications to Viewpoint or to integrate existing applications with ViewPoint.
- Programming tools for developing software in both C and Mesa programming languages
- Applications and development tools for both XNS and TCP/IP communications including network filing, laser printing, terminal emulation, mail, document/file conversions, and internetwork communications.
- A distributed database management system for project management and other applications.
- Pilot Operating System with concurrent multi-tasking and virtual memory support.
- Software interface packages for integration with all levels of the Xerox Network Systems communications architecture and ViewPoint user interfaces.

NEWSBITS

☐ To further integrate with IBM environments, Xerox recently announced the Xerox Printer Access Facility, or "XPAF™". XPAF enables users of IBM mainframes to output data through Interpress or native formats to all Xerox printers, ranging from the tabletop 4045 up to the high-speed 9790. The package runs under both the MVS/370 and MVS/XA operating systems.

☐ Auto-trol, an industry leader in technical illustration graphic systems for electronic publishing, recently announced its planned support for Interpress. Almost four dozen companies in electronic publishing now have plans for Interpress support.

☐ Xerox continues product introductions that support integration of its equipment with other vendors. Recently, Xerox announced TCP/IP protocol support under XDE, the Xerox Development Environment. Both end-user applications and development tools are provided for ARPA protocols. The following are supported: ARP, FTP, ICMP, IP, SMTP, Telnet, TCP, TFTP, and UDP.

☐ The next EDGE conference is scheduled for the first week of June in Orlando, Florida. EDGE is an association of users of Xerox products with conferences held twice yearly. The conferences are an opportunity for users of Xerox products to hear presentations on the latest developments with Xerox products and also to get involved in special information exchange groups. New within EDGE is "EDGEware", a listing of user-contributed applications and tools that can be shared within the user community.

☐ Xerox announced Ventura Release 1.1, with enhanced integration capabilities and 80 new features, including broader support for a greater variety of imported graphics and text; improved typographic controls; enhanced interactive page composition capabilities; faster printing for some laser printers, and greater connectivity to a broader range of printers. The new release includes conversion for Hewlett-Packard Soft Fonts and the ability to use the entire library of Adobe printer and screen fonts with any PostScript printer. Ventura also supports Interpress, and XNS integration is available from Xerox for IBM PC's and compatibles. First shipment is expected in May.

FOR MORE INFORMATION

Please indicate below requests for:

- ☐ Standards and Protocols Documentation
- ☐ Training Courses
- ☐ XNS/Interpress Conferences
- ☐ Implementation Aids
- ☐ Other _____

Send me information at:

Name _____

Company _____

Address _____

_____

_____

Xerox Systems Institute

Mail to: **Xerox Corporation**
Xerox Systems Institute
475 Oakmead Parkway
Sunnyvale, CA 94086
(408) 737-4652

WS Functional Specification
VP File Conversion of IBM DCA Documents

Author: Harold Shinsato

Date: December 10, 1986

Filed: [CalTech:OSBU South]<OS Documentation>Post OS 6.0 Features>

Approvals:

Donna Oku
WSSBU Product Manager, DCA Conversion

Michalene Casey
NSSBU Product Manager, IBM Products

Terry Roberts
ISD User Interface Board

Reviewers:

Mike Kernaghan
Jean-Marie deLaBeaujardiere
Elisabeth Waymire

XEROX
INFORMATION SYSTEMS DIVISION
Workstation Software SBU
Sunnyvale El Segundo

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Purpose, Scope

The purpose of this document is to describe the functionality of a conversion program which operates in ViewPoint to convert documents in IBM's DCA format to and from ViewPoint format. Since document conversions are the process of translating the features of one format into another, most of the information about the specific conversion features are described in the tables in the appendices. The User Interface section only describes the features which could not be converted closely between these two formats.

2.0 REFERENCES

2.1 *Converter,* available on [CalTech:OSBU South]<OS Documentation>Post OS 6.0 Features>Converter>.

2.2 *DCA Project Gantt Charts,* available on [WS Emulations:OSBU North]<IFC>Doc>.

2.3 *DisplayWrite3 User's Guide,* November 1984.

2.4 *Document Content Architecture: Final - Form - Text Reference,* IBM Publication SC23-0757, May 1985.

2.5 *Document Content Architecture: Interchange Document Profile Reference,* IBM Publication SC23-0764.

2.6 *Document Content Architecture: Revisable - Form - Text Reference,* IBM Publication SC23-0758-0, June 1983.

2.7 *Document Interchange Architecture: Concepts and Structures,* IBM Publication SC23-0759-0, June 1983.

2.8 *Document Interchange Architecture: Technical Reference,* IBM Publication SC23-0781-0, June 1983.

2.9 *Document Interchange between IBM DCA and Xerox ViewPoint,* Jean-Marie deLaBeaujardiere, May 1986, available on [WS Emulations:OSBU North]<IFC>Doc>

2.10 *Foreign Conversion (ESCN),* available on [CalTech:OSBU South]<OS Documentation>OS 6.0 Features>Star 4.0 Features>Converter>.

2.11 *IBM 3270 Information Display System Character Set Reference,* IBM Publication GA27-2837-3, December 1979.

2.12 *Xerox Character Code Standard,* XSIS 058404, April 1984.

3.0 OVERVIEW

3.1 Background

In this background section, I can not possibly hope to explain all the features of DCA. Such an explanation would be very difficult and lengthy, and far beyond the scope of this functionality specification. The goal of this document is to describe the functionality of the DCA ↔ ViewPoint conversion, not DCA. Instead I will try to explain the aspects and peculiarities of the DCA/IBM world and of DCA functionality which are important to the DCA conversion.

3.1.1 REVISABLE FORM - FINAL FORM

There are two DCA document types: Revisable Form Text (RFT), and Final Form Text (FFT). Revisable Form functionality is a superset of Final Form. A document stored in Revisable Form may easily be picked up again and edited without loosing the intent of the document's original creator. For example, one may create a paragraph so that a subsequent editor may add a sentence to the paragraph while leaving the paragraph and its associated properties, such as margins and justification, intact.

A document stored in Final Form DCA is not intended to be edited, but rather to be sent to a printer to render the document on paper. A paragraph in revisable form would become an unconnected series of lines in final form.

3.1.2 TYPEWRITER DOCUMENT MODEL

An important aspect of DCA is its typewriter-oriented model of documents. As Jean-Marie said, "Ever present is the image of a printer assembly moving horizontally and vertically over a sheet of paper" [Ref 2.9] . DCA has numerous features that seem almost to have been tacked onto the original printer commands, but the essential typewriter model remains. Advanced possibilities more suited to laser printer technologies, such as the display of bitmap images and structured graphics, are not available with DCA.

3.1.3 NETWORK ENVIRONMENT - DIA

Although DCA documents can exist by themselves, as they do on local PC environments (as with DisplayWrite3 software), DCA documents may also be part of a larger set of communication protocols called Document Interchange Architecture (DIA). DIA allows documents to be filed and retrieved from library services, and allows documents to be sent as messages in a networked environment. The IBM literature describes DIA as a "program-to-program communication architecture" [Ref 2.7]. DIA features operate with machines connected by a network, and must be handled with programs that have access to the network. Since the Converter Icon only deals with files on the desktop, the DCA conversion will not handle DIA.

3.1.4 EBCDIC

A peculiarity of DCA is its character set. It is Ebcdic, instead of the more standard Ascii (which ISO has adopted, and which mostly matches Xerox character set 0). Ebcdic byte-codes have different meanings according to the nationality of the terminal or machine on which the byte-codes reside [Ref 2.11]). DCA attempts to avoid this ambiguity with a four byte field, called the Global Coded Graphic Character Set ID (GCID), to indicate which Ebcdic character set should be used to interpret the code-bytes. The first two bytes indicate the character set, and the last two bytes indicate the code page. The code page represents a full map of characters, and the character

set indicates a subset of the characters from the code page. The four byte field can potentially represent over 4 billion possible character mappings. Of this huge number, over 65 thousand may be set by the customers. Unfortunately, IBM describes only two of the character sets in its public DCA references. The DIA Technical Reference [Ref 2.8] mentions a document named the "Registry of Graphic Character Sets and Code Pages, IBM Corporate Specification C-H 3-3220-050," but this document has been impossible to aquire at this date. Many other sources confirm the unavailability of this document.

3.1.5 FILE INCLUSION

A feature of DCA Revisable Form that has no current parallel in the ViewPoint world is the ability to construct a document across several files. DCA has a complex method for allowing the inclusion of DCA files and parts of DCA files in a DCA document. For example, a DCA document may have a command which says to include the document section named "19123" from the DCA file named "foobar". ViewPoint documents may soon have multi-file documents (called dockets), but such a feature has not yet been implemented.

3.1.6 DCA - IBM's DOCUMENT INTERCHANGE FORMAT

DCA is part of IBM's document interchange strategy, which means that IBM software dealing with document formats should be able to understand DCA documents. The present IBM systems which support DCA include DisplayWriter, PC, System/36, System/38 (Final Form only), 4700, and 5520. The ability to interchange DCA documents makes our machines more useful in any office environment which might include these IBM systems, and since the Network Services Section of ISD is currently implementing a Systems Network Architecture (SNA) gateway, the physical ability to interchange such documents will shortly be in place. IBM's dominance of the computer market also increases the importance of DCA compatibility. DCA is supported by several non-IBM systems including Data General Eclipse Product Line, DEC VAX VMS Systems, HP 3000 Series (Revisable Form only), and there are certainly more systems which have the ability to interchange DCA documents (for example, a DCA conversion utility exists for the Apple Macintosh).

3.2 Goals

The goal of file conversion of DCA documents in ViewPoint is to provide the ViewPoint user with the ability to convert any DCA document into ViewPoint format, and to convert any ViewPoint document into DCA. Because the two formats do not contain the same features, another goal is to provide the best mapping of features possible with the resources allotted to this project.

It is **not** a goal to convert everything precisely. Since DCA and ViewPoint have different features and different views of what constitutes a document, many things will never be able to convert precisely. Font sizes are different, so words may not always appear on the same line. Pages might not always paginate into the exact same number of pages in the different formats. Such problems can not be solved by conversion programs. Only better standardization of document formats will allow conversions to be more precise.

3.3 AvailabilityStaging/Schedule Summary

The DCA conversion will be released with the SNA Gateway. Please see the *DCA Project Gantt Charts* for more scheduling details [Ref 2.2].

Many problems may be encountered when processing a file as a DCA document. The syntax may be invalid (for example, we may encounter a multibyte command that we don't recognize), or a parameter may be out of range. Other problems may relate to certain incompatibilities between DCA and ViewPoint. For example, line numbering and footnotes don't exist in ViewPoint.
These problems will be shown in the ViewPoint document at the point where the problem was encountered.

The actual error messages will depend on the nationality of the workstation, but the problem indicator will have the following form:

BEGIN-PROBLEM PROBLEM-MESSAGE END-PROBLEM

In the US, **BEGIN-PROBLEM** will be "<ERROR: ", and **END-PROBLEM** will be ">". **PROBLEM-MESSAGE** will represent a short message describing the problem. The example offered in figure 1 should make this more concrete.

This paragraph includes a multibyte command that the conversion program won't recognize. The <ERROR: Bad multibyte>command may indicate the beginning of italicized text, but the present conversion sees it as an undefined command.

Figure 1: A portion of converted DCA text from a VP document which includes a problem indicator that can be easily searched and discovered.

Since the problem indicators should be short in length to avoid heavy disruption of the actual converted text, the problem messages will not fully explain all aspects of the problem encountered. Such an explanation may be made in the Converter History where the short message can be expanded. The Converter History will list all problems, the number of times each specific problem is encountered, and the error message that was inserted into the document surrounded with quotation marks. Figure 2 gives an example of the Converter history after a conversion that encountered some problems.

Conversion History started at 7-Nov-86 13:54:17

Converting □ Problematic.dca...
9 unknown and discarded multibyte command(s) "<ERROR: Bad multibyte>".
1 unknown and discarded structure(s) "<ERROR: Bad structure>".
5 discarded instance(s) of external text "<ERROR: External text dropped>".
23 character(s) replaced by black box "▮".done.

Figure 2: The Converter History after an error prone conversion.

Note: Some users may not want error messages inserted into the document. Conversion Specific Parameters [Ref 2.1] would allow users to decide whether to omit error messages. This feature of the converter is not yet available, but such an option may be released with a later version of this conversion.

4.1.3 Other Error Conditions, Messages

USER ABORT
The user may abort the conversion by pressing the stop key. The conversion will

stop, and a partially completed ViewPoint document will be output if possible. The aborting will be noted in the Converter history.

NON DCA DOCUMENTS
If the user attempts to convert a document which the DCA converter recognizes is not a DCA document, the conversion will be aborted, and the user will be notified in the Converter history. The DCA Revisable Form conversion will reject files without the Format Unit Prefix (FUP), which is five bytes long. The FUP is the mandatory first structure in all DCA Revisable Form documents. The Final Form Conversion will reject files which begin without a valid Ebcdic character or unibyte command. The Final Form conversion will accept most files since most files will meet such a loose requirement. Any other protection would be expensive to implement, and would still provide no guarantees.

RESOURCE PROBLEMS
The local disk may run out of pages during the conversion. If this happens, then the user is notified and the conversion is aborted and the document is deleted (because there is no room to make the document well-formed).

4.2 IBM DCA Final Form → ViewPoint Document Conversion

DCA Final Form → ViewPoint document conversion enables conversion of DCA Final Form Text (FFT) files. These DCA files may be created on any system, but they must be placed on the desktop before they can be converted.

The DCA Final Form conversion is registered in the Converter Icon to accept file types 0 (unspecified), 2 (text), and 4454 (DCA FFT). If the source format is chosen by overriding the icon type, then any file type will be accepted for conversion.

All features of the Final Form conversion are described in section 4.1, since DCA Final Form is a subset of Revisable Form.

4.3 ViewPoint → IBM DCA Revisable Form Document Conversion

ViewPoint → DCA document conversion enables conversion of ViewPoint documents into files in DCA document format. The conversion is not responsible for the method in which the user transmits the DCA files to another system which understands DCA.

The ViewPoint to Revisable Form conversion produces files of type 4453 (DCA RFT).

4.3.1 Conversion of ViewPoint Features

This section only describes document features that did not have a straight forward mapping from ViewPoint to DCA. Please see Appendix D for a more complete description.

- Cyrillic letters and Kana syllables are mapped to the DCA SUBSTITUTE character (EBCDIC 3F hex which is usually rendered as low-bar "_ ").
- All ViewPoint characters without a corresponding DCA character are mapped to the DCA SUBSTITUTE character (EBCDIC 3F hex).
- Fields are converted to user prompts.
- Conver sheets are copied to the destination file, **not** converted. Unless the destination system understands coversheets, they will be lost during transmission.

- Equation frames, frame captions, and text frames are not converted.
- Graphics frames, pie/bar charts, bit maps, scanned images, and CUSP buttons are not converted.
- ViewPoint proportional fonts will be mapped to Modern (12 pitch) and Essay Italic (12 pitch) -- proportional fonts are not available outside of 12pitch.
- ViewPoint fixed-pitch fonts will be mapped to Courier 10, Courier 12, Courier 15, and the corresponding Courier italic fonts (according to VP font size).
- Paragraph pre-leading and post-leading will be simulated with empty paragraphs (Hard Carrier Returns).
- Multiple column pages are converted as single column pages. Column breaks are not converted.

C1	C2	C3		C4
	C2.1	C3.1	C3.2	
One	A B C	11	22	33 44
Two	D E F	111		333

Figure 3: A ViewPoint Table before conversion to DCA

4.3.2 Enumeration of ViewPoint Tables

ViewPoint tables can be quite complex. Columns can be split into subcolumns, and row into subrows. The complexity makes it difficult to translate a ViewPoint table into DCA columns, although the features are superficially similar. For reasons of complexity and developer resources, we will only enumerate ViewPoint tables, giving each table entry a line of its own in DCA. Since row entries in ViewPoint may be longer than one line, the correspondence between row entry and DCA line will not necessarily be one-to-one.

When a table is encountered, a separator string is written to the DCA document, followed by "Name of VP Table: " and the actual name of the table. The separator string used in the example is 18 dashes. Since the string will be taken from a message file, it need not be 18 dashes on all workstations.

Before the simple enumeration of the table entries, the columnar structure will be described in textual form. Every column and every subcolumn will get a paragraph with the following form:

Column name: **COLUMN-NAME**
Column header: **COLUMN-HEADER**

```
------------------
ViewPoint Table Name: Table1

Column name: Column1
Column header: C1

Column name: Column2
Column header: C2

Column name: Column2.Column1
Column header: C2.1

Column name: Column3
Column header: C3

Column name: Column3.Column1
Column header: C3.1

Column name: Column3.Column2
Column header: C3.2

Column name: Column4
Column header: C4
------------------
One
A
B
C
11
22
33
44
------------------
Two
D
E
F
111

333
------------------
```

Figure 4: The table from Figure 3 after conversion to DCA

COLUMN-NAME will represent the unique name that is displayed in the property sheet. Subcolumns will have the name of the parent column prepended with a period separator, just as in the VP property sheet. **COLUMN-HEADER** will represent the header text that appears in the box above the actual table column.

The separator string line will divide the column descriptions from the row entries, and the same separator will divide each row. The rows themselves are listed from top to bottom. Row entries are taken from each row, top to bottom (for sub rows), and left to right.

Figures 3 and 4 (above) demonstrate table enumeration at work. Please note that subrows are not divided with the separator string. Only the highest level of rows are divided in such a fashion.

4.3.3 Indicating Problems

A certain number of problems may be encountered when converting a ViewPoint document to DCA. These problems are mostly regarding lost content, such as dropped graphic frames, CUSP buttons, etc. Others problems regard certain format incompatibilities (e.g. a smaller amount of text can be handled in a DCA margin text than a ViewPoint header or footer). We would like to indicate the location of such problems in order to facilitate the correction of such problems. Such indications are made in much the same way as in DCA to ViewPoint conversion. Please see section 4.1.2 for a description of the form of such error messages. Figure 5 gives an example of DCA text showing the location of a dropped ViewPoint graphics frame.

```
This is just text that has been converted.

<ERROR: Graphics dropped>

The above VP diagram is a graphic description of the conversion
process.
```

Figure 5: A portion of a DCA document produced by the converter.

Problem indicators should be short in length to avoid heavy disruption of the actual converted text. But since the DCA documents cannot be perused in ViewPoint, the error messages should not lean heavily on the Converter History for interpretation (as is done in the DCA to ViewPoint conversion). The Converter history will still count the instances of problems, and expand on explanations where needed. Figure 6 gives an example of the Converter history after another problematic conversion.

```
Conversion History started at 7-Nov-86 13:54:17


Converting ▮ Problematic...
9 graphics frame(s) discarded "<ERROR: Graphics dropped>".
1 CUSP (s) buttons discarded "<ERROR: CUSP button dropped>".
5 equation frame(s) discarded "<ERROR: Equation dropped>".
23 character(s) replaced by the Substitute code (3F hexadecimal).
done.
```

Figure 6: The Converter History after converting error prone VP document to IBM DCA.

Note: Some users may not want error messages inserted into the document. Conversion Specific Parameters [Ref 2.1] would allow users to decide whether to omit error messages. This feature of the converter is not yet available, but such an option may be released with a later version of this conversion.

4.3.4 Error Conditions, Messages

USER ABORT
The user may abort the conversion by pressing the stop key. The conversion will stop, and a partially completed DCA document will be output if possible. The aborting will be noted in the Converter history.

NON VIEWPOINT DOCUMENTS
If the user attempts to convert a document which is not a ViewPoint document, the conversion will be aborted, and the user will be notified in the Converter history.

VIEWPOINT DOCUMENTS THAT CAN NOT OPEN
In certain circumstances, ViewPoint will not be able to open the document. This may be due to insufficient disk pages, or because the document is damaged, or for unknown reasons. If such is the case, the conversion is aborted, and the user is informed why through the Converter history.

RESOURCE PROBLEMS
The local disk may run out of pages during the conversion. If this happens, then the user is notified and the conversion is aborted and the resulting document is discarded.

5.0 ASSUMPTIONS & CONSTRAINTS

5.1 ViewPoint Document Compatibility

When an older ViewPoint document is dropped on the converter icon and passed to the DCA conversion, the documents implementation may try to upgrade the document. This upgrading process must be accomplished by the documents implemetation and will not be done by the DCA conversion. If the documents implementation is unable to upgrade, the conversion will be aborted.

5.2 Post ViewPoint 1.0 Document Features

We are not attempting to convert new features of the ViewPoint editor. All features which have been or will be added after version 1.0 of ViewPoint, such as dot leaders and table of contents generation, will be ignored. Since ViewPoint document handling will be substantially enhanced, we wait for the comprehensive set of changes before we will convert them.

6.0 PERFORMANCE REQUIREMENTS

6.1 Size, speed, response, throughput

The time required for a conversion is dependent upon the size and complexity of the object to be converted. Also, the time required for conversions to and from ViewPoint documents are directly dependent on the performance of the ViewPoint documents implementation. There are no specific performance requirements for the DCA conversion.

6.2 Security, data protection, privacy considerations

Not applicable.

7.0 RELIABILITY, MAINTAINABILITY REQUIREMENTS

The original object to be converted is not modified during conversion so it should never be damaged if problems occur.

Conversion of an object may be aborted if desired by pressing the STOP key. If a partially converted object is well-formed, the object will be saved and placed on the desktop. If a partially converted object is not well-formed, it will be deleted and the user will not see it appear on the desktop.

8.0 INSTALLATION, INITIALIZATION, PRODUCT FACTORING

The DCA ↔ ViewPoint conversion will be packaged as one application, and will be separately loadable from floppy disk. After loading, the conversion will become available to the user via the Converter option and property sheets.

As with the current conversions, the DCA ↔ ViewPoint conversion will be product factored.

9.0 MULTINATIONAL, MULTILINGUAL REQUIREMENTS

There have been no formal multinational requirements made to this date, but there are several issues that can be settled.

9.1 Character sets

Although the DCA conversion will not handle the full IBM character set (see section 11.1), the current code pages which we handle should cover all US and European needs (except for those needing Cyrillic). Accented characters, greek letters, and most symbols from Xerox Character Set 357 will be translated.

Kanji and Kana mappings may arrive at any point, but there are some ambiguities about the method of encoding Kanji characters. The 3270 apparently uses two-byte codes to encode Kanji, using shift-in and shift-out characters to delimit the two-byte sections of the text. If this is the means used by IBM DCA, more effort will be needed to accomodate the encoding. The addition of single-byte per character code pages only involves writing the tables for both directions of a conversion (into and out of DCA Ebcdic).

9.2 Page sizes

Other cultures have different pages sizes from that of the U.S. DCA allows direct specification of page size, so this is not a concern. We have a one-to-one mapping.

9.3 Tab alignment character

The period is not used universally as a decimal separator. Since decimal tabs are aligned at the decimal separator, we need flexibility for assigning the character on which tabs will align. The DCA tab alignment character may be period, comma, or colon (". , :"). Unfortunately we have no means of specifying the tab alignment character when creating a ViewPoint document. Such an ability will be added to later version of the VP Document Editor.

10.0 DEPENDENCIES & COMPATIBILITY

10.1 Hardware Configuration/Revision Level

None.

10.2 Software Subsystems

This product depends on Basic Workstation 4.2, Filing 8.0, Pilot 12.2, the ViewPoint 1.1 version of the Converter, and ViewPoint Documents 2.1. The ViewPoint documents and Converter Icon software must be loaded.

10.3 Data Conversions & Upgrade

This conversion is not responsible for upgrading ViewPoint documents from previous versions.

10.4 Tableware.

Currently, Titan and Terminal fonts are used in DCA conversions because they are the fullest fixed-pitch fonts. They still do not support the full set of characters defined in

the XC1 character code standard [Ref 2.12]. They also do not support all the needed characters being translated from DCA (See Appendix A). The quality of this product is greatly affected by the incomplete fonts, and would benefit if the fonts were to be filled in before release. Otherwise, customers will have to put up with black boxes where characters should be displayed.

10.5 Standards.

Character conversions are based on the Xerox Character Code Standard.

11.0 UNRESOLVED ISSUES, RISKS

11.1 Full IBM Character Set Support

We currently support Code Pages 256 (charset 337) and 259 (charset 340). These two code pages are not the full IBM character set. We do not have the documentation to support the full character set. The DCA documentation only describes two character sets of code page 256 (337 and 103, a subset of 337). Code page 259 was deduced from the Displaywrite3 manual [Ref 2.3] and software.

There is rumored to be document describing all we need, but it has not been discovered. For a fuller discussion of Ebcdic character set peculiarities, see section 3.1.4.

11.2 DCA Footnotes

Translating footnotes from DCA into ViewPoint is difficult because footnotes do not exist in ViewPoint. The best way of handling the footnotes has yet to be found. If and when such a method is found, the present method may be changed.

11.3 DCA Columns and ViewPoint Tables

DCA has a primitive columnar data capability which we do not translate to the more advanced ViewPoint tables functionality. Such a translation is possible, but would be costly to implement.

Our current method of translating ViewPoint tables into DCA allows the preservation of data. It would be feasible to translate some of the features available in a ViewPoint table to DCA columns, but again it would be costly to implement. The potential offering of Tabular Information Exchange (TIE) would considerably simplify the translation of ViewPoint tables.

11.4 ViewPoint Frames

All frames in a ViewPoint document are dropped when translated to DCA format. It might be preferable to insert blank space corresponding to the size of the dropped frame. Although the conversion of frame captions is possible, they are also dropped. The presence of captions separated from the frame contents would most likely be confusing.

11.5 DCA Italics

When converting ViewPoint italics to DCA, many users may desire that the italics be presented in a different fashion than DCA italics. Apparently, printing italics in DCA can be very difficult due to the necessity of replacing printwheels in the printer. If

such a desire by users is prevalent, then the present ViewPoint → DCA conversion functionality should be changed to use underscore, or whatever the user prefers.

11.6 Page Format Changes in ViewPoint

DCA allows page formats to change only at hard page breaks, while ViewPoint allows page format changes at any point in the text. When converting ViewPoint documents to DCA, we could insert a page break whenever we encounter a ViewPoint page format character. Presently we save the page properties until we hit a ViewPoint page break.

11.7 DCA Character Aligned Tabs

Multinational requirements may force us to handle more than decimal aligned tabs. At present, tabs aligned on a comma or colon are centered when translated to ViewPoint. This limitation is imposed by ViewPoint Documents.

11.8 Customer Assigned Fonts and Character Sets

The way which DCA signifies the current font and character set allows the customer to specify their own fonts and character sets. IBM has reserved sections of the font and character set identification fields for customer assignment. The customer would have to tell the conversion how to interpret alternative character sets and fonts, which we currently do not allow.

11.9 Conversion Specific Parameters [Ref 2.1]

Many of the problems above may be solved if the user could customize the conversion process. A user would be able to specify how he or she wanted italics to be presented in a DCA document, or whether or not to replace unconverted frames with blank lines. All such solutions, however, will take time and we will not be able to attempt each of them. Also, since the design for conversion specific parameters has not been completed, it seems premature to specify any such solutions in this document.

12.0 TEST PLAN

12.1 Test approach

To be designed.

12.2 Test specification

To be designed.

13.0 REVISION LOG

6/23/86	Shinsato	First version after internal review. Released externally.
11/11/86	Shinsato	Many changes to include VP table enumeration, the addition of code page 256 translation tables, and problem indication. This version was released to Spec Review.
12/10/86	Shinsato	Clarifications made from Spec Review comments. Removed feature of anchored text frame enumeration.

-- File: Cv860Changes.doc
-- Last Edit: Shinsato 13-May-85 16:25:55
-- Owner: Workstation Applications - Foreign Conversion Team

This document describes the changes in operation of the 860 <=> Star conversion, as seen by the user.

MAPPING BETWEEN 860 AND STAR is an alphabetic listing of the changes to the mapping of Nova document features into 860 and visa versa. The categories were chosen from the Procedures Guide, pp 17-28, which is part of the Star Reference Library. Any item not included has not changed in functionality. All bracketed statements describe features or problems that no longer exist, or explain why the item is listed (point out specifically what has changed).

USER INTERFACE describes how the conversion can be used and how it communicates with the user. This section basically describes the messages issued, when they occur, and what they mean.

MAPPING BETWEEN 860 AND NOVA

* Headings/Footings
  => After Conversion to 860
    - Only the first page format character is processed. [This was not stated in the Conversion Guide, but was true for the old conversion as well.]
    - The page number delimiter from the message file in the heading/footing is always translated to "Code 2". [This was only done before if the heading/footing began with a new paragraph character. Now all heading/footings must begin with a new paragraph character].
    - The page number pattern is merged before the heading/footing, as it appears in the document. [It is no longer placed on a separate line above the headings/footings.]
    - The text in the heading/footing may change it's looks (see Character Properties.) [The font properties of the converted heading/footing are no longer constrained to the properties of the first visible character in the Nova heading/footing.]
    - Tab settings in the Nova headings and footings will be ignored; the tabs settings of the first paragraph in the Nova document will be used in the 860 heading and trailer instead. [In the Star, tabs in the heading/footing did not mean much, so this was not listed although it was true before as well.]
  => After Conversion to Nova
    - [There was no category listed in the version of the Procedures Guide I have.]
    - Code 2 is now translated according to the page number delimiter from the message file. [Before it was always translated to "#".]

* Space, Required Space, and Non-Breaking Space
  [What is a Required space?]

USER INTERFACE

* Messages
  - All messages to the user are posted in the Converter history and the Attention window unless specifically stated otherwise.

* Aborting
  - If the user presses STOP during the conversion, then the conversion is aborted, the user is notified, and the partially converted document is returned.

* Bad format of the document to be converted
  => Converting Nova documents to 860.
    - Non-Nova documents cannot be opened. The user is notified that the document to be converted could not be opened
  => Converting 860 documents to Nova
    - If the document to be converted does not have a format character, an empty document will be the result. If the document format is bad, the questionable parts will be skipped. {{** Error messages may be generated as well if I put them in.**}}

* Missing implementations
  - If the Nova <=> 860 conversion code is loaded before the Converter then the registration is not done. A message is posted only in the Attention window asking the user to load the Converter and try again.
  - If documents are not loaded when the conversion is attempted, a message

is posted saying so.

* Out of space on the Workstation

* Skipped text

-- File: CvToDIF.doc - last edit
-- Shinsato 11-Feb-85 22:24:40

This document describes the operation of the Star to DIF conversion as seen by the user.

MAPPING FROM STAR TO DIF is an alphabetic listing of the mapping of Star document features into DIF.

USER INTERFACE describes how the conversion can be used and how it communicates with the user. This section basically describes the messages issued, when they occur, and what they mean.

==========

MAPPING FROM STAR TO DIF

* Bold (now called weight in ParaPropsDefs)
  - Converts to DIF emphasis, which is implementation specific.

* Characters, printing
  - An Ascii translation of the characters is attempted.
  - Only Codes0 are translated (character set Roman)
  - All accents are stripped from characters.
  - Characters which aren't Ascii are printed as a question mark

* Characters, properties
  - See Bold, Italics, Underline, Strikeout.

* Equation frames
  - no conversion

* Fields
  - appears to be no conversion

* Graphics and Graphic frames
  - no conversion

* Heading/Footing
  - After conversion, only the left heading/footing of the first page format character is used.
  - Only 8010 text and paragraph characters are translated to the DIF heading/footing. (Is this already obvious?)
  - Only 150 characters (total number of characters from the 8010 heading and footing plus control codes to affect centering, new lines, bold, italics, and overstrike) will be converted to a DIF heading/footing. The conversion attempts to balance the sizes of the DIF heading and footing so that the footing will not disappear if the heading is 150 characters in length.
  - The heading/footing will appear on the first page whether or not this was true for the 8010 document.
  - Only the paragraph properties of the first paragraph character are used when producing the DIF heading/footing, any others are treated as newLine characters.
  - The heading/footing will be centered if the first paragraph of the 8010 heading/footing was centered. Otherwise it will appear at the left side of of the page.
  - If there is a heading, the preleading space of the 8010 heading translates into the DIF top margin.
  - If there is a footing, the preleading space of the 8010 footing translates into the DIF footer margin.
  - All other paragraph properties of the heading/footing are ignored.

* Hyphen and discretionary Hyphen/Dash
  - non breaking hyphen becomes a DIF Hard Hyphen
  - discretionary hyphen becomes a DIF Soft Hyphen

* Italics
  - No conversion.

* Justify
  - Converts precisely

* Line spacing
  - 8010 line height is translated to the closest approximation of single, double, and triple spacing.

* Margins

- Page margins and paragraph margins are added together to produce the equivalent DIF margins.
- Top margin of the DIF document is the space between the top off the page and all text (including the heading text). The top margin is set to the preLeading of the heading if there is one. If there is no heading, the top margin is the same as the 8010's top margin (to the closest approximation of DIF lines).
- Bottom margin of the DIF document converts to the closest approximation of DIF lines.
- The DIF footer margin is set from the 8010 footing's preleading.

* Page Break
  - Required page break converts to Hard Page End and Hard Page Start.
  - No conversion of automatic page breaks.

* Page numbering
  - Page number characters ("#" in USEnglish) are converted whereever found in the 8010 headings/footings to their DIF equivalent ("###").
  - Only the page numbering characteristics of the first pageFormat character are used (page numbering never changes in the converted DIF document).
  - Page number format is ignored.

* Page Size and Page Format properties
  - Only 8.5 x 11 inch format is produced (that's all DIF can do).
  - Only the first page format character is handled.
  - Text is single column only.

* Paragraph properties
  - Text justification is preserved; line spacing of single, double, and triple chosen to most closely approximate the 8010 format, and margins are converted as described in Margins.
  - New paragraph characters which begin a page are only read for their properties. They are not translated to a hard new line in the DIF document unless the new paragraph character immediately follows a page break AND the new paragraph is centered (a limitation of DIF).
  - Center converts properly. Right align converts to left align.

* Pitch
  - The pitch of the first new paragraph determines the document's pitch
  - Point sizes from 6 through 11 inclusive convert to pitch 12. Everything else converts to pitch 10.

* Record files
  - No conversion.

* Space, Required space, and non breaking Space
  - converts to space, .. , and Hard Space respectively

* StrikeOut
  - Is translated to DIF overstrike, which is implementation specific.

* Subscripting and superscripting
  - Any level of superscripting is converted to baseline up.
  - Any level of subscripting is converted to baseline down.

* Tables
  - No conversion.

* Tab motions
  - A tab or paragraph tab is converted according to an approximation of the tab's position in the 8010 Document. The tab's type and/or setting may have to be changed to conform to the 8010 document's appearance.
  - A paragraph tab is converted to a normal tab if the 8010 tab setting was a decimal tab. Otherwise the paragraph tab temporarily changes the left margin to the paragraph tab's position until the next new paragraph.
  - A normal tab is converted to a DIF normal tab unless the 8010 tab motion falls under a decimal tab setting.

* Tab Properties
  - Left and Decimal tabs are converted to left and decimal tabs.
  - Right and center tabs are converted to left tabs.

* Text Frames
  - No conversion.

* Underline

- Is translated precisely (underscore).

==========

USER INTERFACE

* Messages
  - All messages to the user are posted in the Converter history and the Attention window unless specifically stated otherwise.

* Aborting
  - If the user presses STOP during the conversion, then the conversion is aborted, the user is notified, and the partially converted document is discarded.

* Bad format of the document to be converted
  - If the document dropped on the converter is not a Star Document then the the converter does not convert it and messages the user.

* Missing implementations
  - If the Star to DIF conversion code is loaded before the Converter then the registration is not done. A message is posted only in the Attention window asking the user to load the Converter and try again.
  - If documents are not loaded when the conversion is attempted, a message is posted saying so.

-- File: CvToRBS.doc
-- Last Edit: Shinsato 29-Apr-85 17:18:39
-- Owner: Workstation Applications - Foreign Conversion Team

CvToRBS provides the user with the ability to prepare an object on the desktop for transmission via the Remote Batch Service. The command for the conversion is logged in the Attention menu as "Convert to RBS."

When the command is invoked, CvToRBS tries to convert every element of the selection into a folder for RBS. The orignal object remains unchanged, although it now resides in the folder along with an instruction file which is read by the remote batch service.

Any object which contains other files, or which is remote, or which can not be opened, is not converted. If CvToRBS can aquire the name of the object which could not be converted, then it is displayed, along with the reason why it could not be converted.

-- CvWordstar.doc
-- Last Edited by: Yien 20-Jan-87 12:08:16

This document describes the operation of the Wordstar (3.3 and 3.4 versions) <-> Star document conversion.

The mapping of Wordstar3.3 <-> Star document and Wordstar3.4 <-> Star document is only different in Characters.

==========

WORDSTAR (3.3 AND 3.4 VERSION) <-> STAR DOCUMENT CONVERSION MAPPING

* Bold

<< Wordstar -> 8010 >>

- Converts precisely.

<< 8010 -> Wordstar >>

- Converts precisely.

* Characters

<< Wordstar3.3 -> 8010 >>

- All characters painted YELLOW on page G-2 and G-3 are converted.

<< 8010 -> Wordstar3.3 >>

- The codes painted YELLOW on the copies of Xerox Character Sets (Set0, Set357) are converted.

- Other characters are converted as a question mark.

<< Wordstar3.4 -> 8010 >>

- All characters are converted in Wordstar3.3; in addition to it, the ones painted PINK on page G4 and G5 are also converted.

- Other characters are converted as a question mark.

- A sequence of ESC, letter, and FS (34C) of Wordstar3.4 are translated into an accent and a letter or just a letter.

<< 8010 -> Wordstar3.4 >>

- The codes are converted in Wordstar3.3; in addition to it, the ones painted PINK on the copies of Xerox Character Sets (Set0, Set41, Set42, Set46, Set357) are also converted.

- Some combinations of diacritical mark and letter are converted into an 8-bit code preceded by an escape code ESC and followed by a code FS, as per Appendix G.

* Characters, properties

<< Wordstar -> 8010 >>

- See Bold, Italics, Underline.

<< 8010 -> Wordstar >>

- See Bold, Italics, Underline.

* Equation frames

<< 8010 -> Wordstar >>

- No conversion.

* Fields

<< 8010 -> Wordstar >>

- No conversion.

* Graphics and Graphic frames

<< 8010 -> Wordstar >>
- No conversion.

* Heading/Footing

<< Wordstar -> 8010 >>
- Only first Heading and first footing are used at beginning of each page.

<< 8010 -> Wordstar >>

- Heading and footing of the first page format character is used.

- Only characters from character set 0 are converted to the Wordstar heading/footing.

- There are only 65 characters (total number of characters from the 8010 heading and footing) for the heading/footing.

- The heading/footing will appear on the first page whether or not this was true for the 8010 document.

- The heading/footing will appear at the left side of of the page.

- All other paragraph properties of the heading/footing are ignored.

* Double Striking

<< Wordstar -> 8010 >>
- Converts Wordstar double striking to 8010 italics.

<< 8010 -> Wordstar >>
- Converts 8010 italics to Wordstar double striking.

* Page Break

<< Wordstar -> 8010 >>
- Wordstar dot command ".PA" converts to a page break character.

<< 8010 -> Wordstar >>
- A page break character converts to ".PA".

* Page numbering

<< Wordstar -> 8010 >>
- No conversion.

<< 8010 -> Wordstar >>
- No conversion.

* Page Size and Page Format properties

<< 8010 -> Wordstar >>

- Only 8.5 x 11 inch format is produced by default.

- Only the first page format character is handled.

- Text is single column only.

* Paragraph properties

<< Wordstar -> 8010 >>

- Only single line spacing.

- carriageReturn plus newLine (<CR> & <LF>) converted to a new paragraph.
-

<< 8010 -> Wordstar >>

- Only single line spacing.

- New paragraph characters converted to carriageReturn plus newLine.

- The paragraph text is sliced into lines of not more than 65 characters and terminated by marked carriage return (215C) plus newLine

- Center converts properly. Right align converts to left align.

* Font Size and Font Style

<< Wordstar -> 8010 >>

- Converts to Modern 12.

<< 8010 -> Wordstar >>

- Font size changes are ignored.

* Record files

<< 8010 -> Wordstar >>
- No conversion.

* Space and Required space

<< Wordstar -> 8010 >>

- Converts to space and non-breaking space respectively.

<< 8010 -> Wordstar >>
- Converts to space and Hard Space respectively.

* StrikeOut

<< Wordstar -> 8010 >>
- No conversion.

<< 8010 -> Wordstar >>
- No conversion.

* Subscripting and superscripting

<< Wordstar -> 8010 >>

- superscript converts to normal superscript.

- subscript down converts to normal subscript.

<< 8010 -> Wordstar >>

- Any level of superscripting is converted to superscript.

- Any level of subscripting is converted to subscript.

* Tables

<< 8010 -> Wordstar >>
- No conversion.

* Text Frames

<< 8010 -> Wordstar >>
- No conversion.

* Underline

<< Wordstar -> 8010 >>
  - Converts precisely (underscore).

<< 8010 -> Wordstar >>
  - Converts precisely (underscore).

* Pitch

<< Wordstar -> 8010 >>
  - No conversion.

<< 8010 -> Wordstar >>
  - No conversion.

* Tab, ParaTab, Decimal Tab, Centering, Margin and Justification

<< Wordstar -> 8010 >>

  - All those commands in Wordstar are only used in user interface. An example of Tab:
    when user type a tab, the editor writes out a number of blank spaces (that a tab takes).
Therefore, from Wordstar to 8010 all the Tab, ParaTab, Decimal Tab, Centering, Margins,
    Justification convert to desired spaces.

<< 8010 -> Wordstar >>

  - Tab and ParaTab connvert to Wordstar Tab (a number of spaces that a tab takes).

  - No conversions for Decimal Tab, Centering, Margins and Justification

==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ====

Changes associated with new Converter interface

1. Code is now re-entrant.
2. User abort was fixed.
3. Sessions are used for all NSFiling operations.
4. Busy status checked when calling Converter.Register.
5. Replaced Converter.GetZone with BWSZone.Permanent.

THINGS THAT SHOULD BE FIXED

In module CvFromWordstarImpl, procedure SetAlternateMapping: use array
  constructors and reduce this proc to two assignment statements. I understand
  the motivation for using many single-array-element assignment statements: to
  pair the char in one array with the accent in the other array. However, this
  is extremely inefficient. Keep the pairs as COMMENTS if desired but use the
  array constructor.

Unused messages should be removed from CvWordstarMsgImpl (e.g., kdocument)

==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ====

EDIT HISTORY
13-Feb-85 12:16:03 - Dai -
20-Jan-87 11:34:05 - Yien - Changes associated with new Converter interface,
                            THINGS THAT SHOULD BE FIXED.

-- File: BWSBrownie.doc - last edit:
-- Paul Darling:SBD-E:RX 10-Oct-88 18:13:27

What is it?

BWS Brownie is a VP application based on the Brownie tool used in XDE.

Where is it?

(Alt:OSBU North)BWSHacks/4.0/Tools

What does it do?
---

o Brownie was implemented to help maintain consistent copies of master and archive directories on several file servers.

o A brownie run may consist of one or more copy commands, each of which transfers the files in a file drawer or sub-directory described by a source pat h-name to a target file drawer or sub-directory. The command lines are written into either a Star format document or a simple text document (the command file).

o The source file drawer or sub-directory is enumerated to the bottom level, ie. folders within the file drawer or sub-directory are each enumerated as well as any folders that they contain. All file types are retained, including the types belonging to application data files.

o A file will only be copied to the target directory if no file exists with the same name in the target directory or if it is newer than all files with the same name in the target directory. If there are multiple versions of a file to be copied then the file with highest version number in the source directory will be transferred.

o If the target directory that is specified in a command line does not exist then a new directory will be created (providing access rights are sufficient).

o If the transfer of files is halted due to line failure or a server crash then the brownie run will resume with the transfer of the file or folder in the top level directory that was being copied at the time of disconnection once the connection is re-established. (When connection is broken BWSBrownie will check every 3 minutes for re-connection).

o If the source directory is an application folder then the attributes of the folder are copied to the target directory. If the source directory is a normal folder the contents of the target directory will be updated but the attributes of the directory itself will remain unchanged.

User Interface.

Loading and running the application registers a new command, 'BWS Brownie', in the desktop auxiliary menu. Selecting this command creates a BWSBrownie window that supplies the following functions: -

Close!

Closes the BWSBrownie window.

MakeCmFile!

Bugging this command creates a simple text document on the user desk-top called "Brownie CmFile". This document contains a sample command line that is in the format required by the tool for the Run operation.

This file can be edited by system developers to specify the copy operations desired. (Alternatively a Star format document can be used as the command file which will be editable by non-system developers). The file may contain any number of command lines. Each command line specifies the target directory of the transfer, and either a pathname for a specific file to be copied or a pathname ending with a file name containing wild card characters (* matches with multiple characters, #

matches with a single character), to be used to match against the names of the files in the source directory. Directory pathnames and file names can contain spaces.

Switches for the command line should be inserted at the head of the line before the main text of the command (see the section 'Switches').

Run!

This command takes the currently selected icon and uses it as the command file for the Brownie run. The operation is aborted with an error message if there is no icon selected or if the icon selected is not a simple text document. The file is parsed to extract the run data and any line that is not in the expected format is ignored.

The brownie run creates feedback text which is displayed in the tool window while the transfer operations are in process and this text is then written to a simple text document at the end of the run. This document is called "BWS Brownie.log" and is placed on the desktop.

Stop!

This command will stop the currently running Brownie after the file transfer in progress is completed. If the tool is waiting for the connection to either the source or target directory to be reestablished the run will be terminated when the current 3 minute wait expires.

Switches.

When the command file is parsed prior to a brownie run a switch is searched for in each command line. There are three possible switches. Each switch is effective only for the command line in which it is found.

/f If /f is found before the first '(of the target directory pathname and the command line source pathname specifies a directory to be enumerated, any folders found in the directory will be copied as if it were a single file. If the switch isn't found these folders will be enumerated to the lowest level, therefore ensuring

that all new files are transferred regardless of whether the folders containing them are new.

/s If /s is found before the first '(of the target directory pathname then all folders (including application folders) are ignored. They are neither enumerated or copied as single entities.

/a If /a is found before the first '(of the target directory pathname and the command line source pathname specifies a directory to be enumerated, any application folders found in the directory will be copied as if it were a single file. Normal folders will be enumerated to the lowest level.

/b If /b is found before the first '(of the target directory pathname then all files enumerated in the source directory will be copied to the specified destination directory regardless of whether versions with the same or newer creation dates exist in that directory. Using this switch will therefore cause ALL files identified by the source pathname to be "backed up".

Restrictions.

o The logged-on user running the BWS Brownie needs to have read access to the source directory and add access to the target directory for each proposed file transfer operation. It is not possible to provide separate log-on credentials for different directories, as can be done with the XDE Brownie tool.

o There is no way of specifying the time for the brownie run to start using BWS Brownie. The transfer operations begin immediately the Run! command is bugged.

Example.

This is an example of a command file.

(Ruddles:SBD-E:RX)Mesa/12.0/(Goofy:OSBU North:Xerox)Mesa/12.0/*
/f (Adnams:SBD-E:RX)PreAlphaWorkstation/ViewPoint 1.0q/(Eagle:OSBU North:Xerox)AlphaWorkstation/ViewPoint 1.0q/*
(Adnams:SBD-E:RX)VP Applications/-RX Internal Tools/(IPA:SBD-E:RX)ABWSBrownie/1.2/BWS Brownie

LOG

29/10/85	PRD	Created.
27/11/85	PRD	Modified to include enumeration to all levels.
30/01/86	PRD	Modified to remove scrolling restriction.
07/07/87	PRD	Modified to add description of new switches.
17/03/88	PRD	Converted to simple text.
10/10/88	PRD	Added /b switch description.

86 » Feb 11 To: Tokunaga.FX Re: decrease the microcode cycle when byte cod

Date: 11 Feb 86 09:35:35 PST (Tuesday)
From: Trow.pa
Subject: Re: decrease the microcode cycle when byte code are fetched.
In-reply-to: Tokunaga.FX's message of 26 Sep 85 16:37:36 PDT (Thursday)
To: Tokunaga.FX
cc: Trow

Toru,

I think I finally see a problem in the code below. At {4} there is a 256-way branch pending, so you need CANCELBR [stabilizeNow, 0FF], which Mass probably won't allow. The solution is to put stabilizeNow at an absolute address ending in 0FF. Let me know how it goes. I hope you can get it to work.

I've fixed some bugs and shortened the bitblt code. The latest code is on [Dante]<ST80-1108-Stretch>Rum>.

Jay

New fetching routine: (in refillandtraps.mc)

```
stEmpty:
{ when we arrive here, the buffer is empty, and the ip is pointing at the word to be fetched}
        MAR ← [ipHigh, ipLow + 0], Xbus ← uTimeToStabilize, XDisp,          c1, at [400];
        temp1Low ← 1, BRANCH [noStabilize, yesStabilize, 0E],               c2;


yesStabilize:
        IB ← MD, ipLow ← ipLow + 1, GOTO [stabilizeNow], {adjust ip}        c3;


noStabilize:
        IB ← MD, GOTO [stNotEmpty],


stNotEmpty:
        MAR ← ipLow ← [ipHigh, ipLow + temp1Low], Xbus ← uTimeToStabilize,
              XDisp, L3 ← 0,                                                c1, at [500];
stNotEmptyc2:
        AlwaysIBDisp, DISP2 [stabPaCarr],                                   c2;


{1}{no stabilization is needed, page carry does not occur}
        IB ← MD, ipLow ← ipLow - 1, DISPNI [bytecodes],                     c3, at [0, 4,
        stabPaCarr];


{2}{ this case is special }
   { stabilization is needed, page carry does not occur}
        ipLow ← ipLow - 1, DISPNI [bytecodes],                              c3, at [1, 4,
        stabPaCarr];


{3}{ no stabilizaion is neede, page carry occurs }
        ipLow ← ipLow + 0FF + 1,                                            c3, at [2, 4,
        stabPaCarr];

        MAR ← [ipHigh, ipLow + 0], Xbus ← 0, XDisp, GOTO [stNotEmptyc2],    c1;


{4}{ stabilization is needed, page carry occurs }
        ipLow ← ipLow + 0FF + 1, CANCELBR [stabilizeNow, 0F],               c3, at [3, 4,
        stabPaCarr];


Where : temp1Low = 1 whenever IBDisp occurs.
        uTimeToStabilize = U45 because temp1Low = R4, so, MesaStateG (former U45) = U00.
        The stabilization is needed when uTimeToStabilize = 1 rather than 0FFFF.


{1} : it's normal case, so refill the bytecodes and IBDisp.
{2} : most special case, do not refill the bytecode and IBDisp, since IBDisp can not be canceled. The
```

```
stabiliztion is executed when IBEmpty Refill occurs. In this case, we should think about the
possibility that IBEmpty Trap occurs when the bytecode in IB0 or IB1 needs more 1 or 2 bytes, so I
modified, see the routine below.
{3} : just page carry occurs, adjust the instruction pointer and try to fetch the bytecodes again.
{4} : In this case, IBDisp is canceled. so the stabilization is executed first, and after it, refill
the bytecodes.


FatalError:
        temp1Low ← ErrnIBnStkp, ClrIntErr, CANCELBR [$, 0F],                    c1, at [0];
FatalErrorSpin:
        temp1Low ← temp1Low  LRot12,                                            c2;
        [] ← temp1Low and 0C, ZeroBr,                                           c3;

        Xbus ← uTimeToStabilize, XDisp, BRANCH [ibTrap, otherTrap],             c1;


otherTrap:
        CANCELBR [bailout3, 0F],                                                c2;


ibTrap:
        LODisp, BRANCH [noStabIBErr, yesStabIBErr, 0E],                         c2;


noStabIBErr:
        CANCELBR [bailout1, 0F],                                                c3;


yesStabIBErr:
        ipLow ← ipLow - 1, DISP2 [chkByteLen],                                  c3; { point the word to
        be fetched }


{ it's strange that IB empty Error occured when byte length = 1 }
        GOTO [bailout2],                                                        c1, at [0, 4,
        chkByteLen];


{----- byte length = 2 ------}

ibErrByte2:
        MAR ← [ipHigh, ipLow + 0],                                              c1, at [1, 4,
        chkByteLen];
        ipLow ← ipLow + 1, Cin ← pc16, CANCELBR [$, 0],                         c2; { point the word to
        be fetched next }
        IB ← MD, IBPtr ← 1, GOTO [stabilizeNow],                                c3;


{----- byte leongth = 3 ------}

ibErrByte3:
        MAR ← [ipHigh, ipLow + 0], XC2npcDisp,                                  c1, at [2, 4,
        chkByteLen];
        ipLow ← ipLow + 1, BRANCH [pcOneInByte3, pcZeroInByte3, 0E],            c2; { point the word to
        be fetched next }


pcZeroInByte3:
        IB ← MD, GOTO [stabilizeNow],                                           c3;


pcOneInByte3:
        IB ← MD, IBPtr ← 1, GOTO [stabilizeNow],                                c3;


{------- byte length = 4 ------}
{ So far, we have no such bytecode that it needs another 3 byte to execute, so this case does not
implemeted yet.---- bailout ----}
ibErrByte4:
        GOTO [bailout2],                                                        c1, at [3, 4,
        chkByteLen];
```

```
//< >Doc>comb0c.diff

kikucomb0c.mc, comb0c.mc

*****************************************************************************
File 1: Positions 4393 - 4481

        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;
yesLastComb0C021:

**************************************
File 2: Positions 4388 - 4472

        MAR __ [destAddrHigh, destAddrLow + 0], CALL [getSource1],                      c1;
yesLastComb0C021:

*****************************************************************************
File 1: Positions 5765 - 5929

        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C032],                c1;
{******************************** combinationRule = 4 ********************************}

**************************************
File 2: Positions 5755 - 5938

        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C032],                c1; {***** Toku *****}
{******************************** combinationRule = 4 ********************************}

*****************************************************************************
File 1: Positions 10657 - 10745

        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;
yesLastComb0C081:

**************************************
File 2: Positions 10649 - 10733

        MAR __ [destAddrHigh, destAddrLow + 0], CALL [getSource1],                      c1;
yesLastComb0C081:

*****************************************************************************
File 1: Positions 14162 - 14398

        sourceIndex __ MD or sourceIndex, CALL [store],                                 c3; { sourceWord bitInvert bitOr: destinationWord }
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0B1, yesLastComb0C0B1],                             c3, at [0B, 10, store - return];
noLastComb0C0B1:

**************************************
File 2: Positions 14139 - 14486

        sourceIndex __ MD and sourceIndex,                                              c3;
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
        MDR __ ~sourceIndex,                                                            c2;
        destAddrLow __ destAddrLow + Q,                                                 c3;{ sourceWord bitInvert bitOr: destinationWord }
        Noop,                                                                           c1;
        temp3Low __ temp3Low - 1, ZeroBr,                                               c2;
        BRANCH [noLastComb0C0B1, yesLastComb0C0B1],                                     c3;
noLastComb0C0B1:

*****************************************************************************
File 1: Positions 15485 - 15574

        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0C0C2],                 c1;
yesLastComb0C0C1:

**************************************
File 2: Positions 15570 - 15658

        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;
yesLastComb0C0C1:

*****************************************************************************
File 1: Positions 16390 - 16587

        temp2Low __ ~temp2Low,                                                          c2, at [0D, 10, getSource1 - return]; { sourceWord
        bitInvert }
        sourceIndex __ MD or temp2Low, CALL [store],                                    c3; { ~SWord or DWord }
        destAddrLow __ destAddrLow + Q,

**************************************
File 2: Positions 16474 - 16677

        sourceIndex __ ~sourceIndex,                                                    c2, at [0D, 10, getSource1 - return]; { sourceWord
        bitInvert }
        sourceIndex __ MD or sourceIndex, CALL [store],                                 c3; { ~SWord or DWord }
        destAddrLow __ destAddrLow + Q,

*****************************************************************************
File 1: Positions 17617 - 17935

        sourceIndex __ ~sourceIndex,                                                    c2, at [0E, 10, getSource1 - return];
        sourceIndex __ ~MD and sourceIndex, CALL [store],                               c3; { sourceWord bitInvert bitOr: destinationWord
        bitInvert }
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0E1, yesLastComb0C0E1],                             c3, at [0E, 10, store - return];
noLastComb0C0E1:


```

```
**************************************
File 2: Positions 17708 - 18118

        Noop,                                                          c2, at [0E, 10, getSource1 - return];
        sourceIndex __ MD and sourceIndex,                             c3;
        MAR __ [destAddrHigh, destAddrLow + 0],                        c1;
        MDR __ ~sourceIndex,                                           c2;{ sourceWord bitInvert bitOr: destinationWord
        bitInvert }
        destAddrLow __ destAddrLow + Q,                                c3;
        Noop,                                                          c1;
        temp3Low __ temp3Low - 1, ZeroBr,                              c2;
        BRANCH [noLastComb0C0E1, yesLastComb0C0E1],                    c3;
noLastComb0C0E1:

*****************************************************************************
File 1: Positions 18272

13-Sep-85 13:29:06      Tokunga.fx      modify the routine for combinationRule - 2,4,8,9,0B,0D,0E
13-Sep-85 13:29:06      Tokunga.fx      modify when combination Rule = 6 {comb0C06} }

**************************************
File 2: Positions 18455

13-Sep-85 13:29:06      Tokunga.fx      modify when combination Rule = 6 {comb0C06} }
*****************************************************************************

End of differences seen.
```

```
//<>Doc>comb0e.diff

klkucomb0e.mc, comb0e.mc

*************************************************************************
File 1: Positions 6401 - 6487

{sourceIndex : destination Word
  tmep2Low : halftone Word
  otLow : mask }
comb0E04:

*************************************
File 2: Positions 6391 - 6402

comb0E04:

*************************************************************************
File 1: Positions 6847 - 6940

        temp2Low __ ~uHalftoneWord, BRANCH [noLastComb0E04, yesLastComb0E04],        c3;
noLastComb0E04:

*************************************
File 2: Positions 6761 - 6829

        BRANCH [noLastComb0E04, yesLastComb0E04],                                   c3;
noLastComb0E04:

*************************************************************************
File 1: Positions 12381 - 12471

        temp2Low __ ~uHalftoneWord, BRANCH [noLastComb0E09, yesLastComb0E09],       c3;
noLastComb0E09:

*************************************
File 2: Positions 12288 - 12376

        temp2Low __ ~temp2Low, BRANCH [noLastComb0E09, yesLastComb0E09],            c3;
noLastComb0E09:

*************************************************************************
File 1: Positions 17315 - 17405

        temp2Low __ ~uHalftoneWord, BRANCH [noLastComb0E0D, yesLastComb0E0D],       c3;
noLastComb0E0D:

*************************************
File 2: Positions 17239 - 17307

        BRANCH [noLastComb0E0D, yesLastComb0E0D],                                   c3;
noLastComb0E0D:

*************************************************************************
File 1: Positions 17471 - 17636

        Noop,                                                                       c2;
        sourceIndex __ MD,                                                          c3; { get and merging }
        sourceIndex __ sourceIndex or temp2Low,                                     c1;
        Noop,                                                                       c2;
        CALL [store0E2],                                                            c3;

*************************************
File 2: Positions 17373 - 17565

        temp2Low __ ~temp2Low,                                                      c2;
        sourceIndex __ MD,                                                          c3; { get and merging }
        sourceIndex __ sourceIndex or temp2Low,                                     c1;
        temp2Low __ ~temp2Low,                                                      c2;
        CALL [store0E2],                                                            c3;

*************************************************************************
File 1: Positions 19015 - 19199

        temp2Low __ ~uHalftoneWord,                                                 c2;
        sourceIndex __ ~MD,                                                         c3; { get and merging }
        sourceIndex __ sourceIndex or temp2Low,                                     c1;
        Noop,                                                                       c2;
        CALL [store0E2],                                                            c3;

*************************************
File 2: Positions 18949 - 19142

        temp2Low __ ~temp2Low,                                                      c2;
        sourceIndex __ ~MD,                                                         c3; { get and merging }
        sourceIndex __ sourceIndex or temp2Low,                                     c1;
        temp2Low __ ~temp2Low,                                                      c2;
        CALL [store0E2],                                                            c3;

*************************************************************************
File 1: Positions 19994 - 20082

        temp2Low __ temp2Low xor ~temp2Low,
                BRANCH [noLastComb0E0F1, yesLastComb0E0F1],                         c3;

*************************************
File 2: Positions 19942 - 20040
```

```
        sourceIndex _ sourceIndex xor ~ sourceIndex,
                BRANCH [noLastComb0E0F1, yesLastComb0E0F1],                    c3;

*****************************************************************************
File 1: Positions 20168 - 20269

        MDR _ temp2Low, temp3Low _ temp3Low - 1, ZeroBr,                       c2; { put Zero }
        destAddrLow _ destAddrLow + Q,

**************************************
File 2: Positions 20126 - 20230

        MDR _ sourceIndex, temp3Low _ temp3Low - 1, ZeroBr,                    c2; { put Zero }
        destAddrLow _ destAddrLow + Q,

*****************************************************************************

**************************************
File 2: Positions 20699

{ Edit history:
  }

*****************************************************************************

End of differences seen.
```

Date: 12 Jun 87 15:31:59 PDT (Friday)
From: Malcolm.PASA
Subject: Stabilize
To: Trow
cc: lansford, malcolm, "malcolm:xsis:xerox".ns

Here's a first cut:

```
-- essential state comprised of oop and next

Deallocate: PROC [oop: Oop] = BEGIN
-- free this oop and Deallocate any of its fields that refD to 0.
-- essentially recursive, freeing a tree based at oop.
-- this is an 'iterative' version; recursion state is hidden in the tree.
-- last is offset of next field to deal with; held in delta word.
-- last is descending, so don't have to remember stop index.
-- when going another level, link back is put in this last field.
-- (which formerly held reference to the oop for the new level)

   next: Oop ← nil;

   loc: PhysicalAddress;  -- start of object header
   last: CARDINAL;  -- offset of last reference field

   IF OopIsSmallInteger[oop]
   OR OtEntry[Of[oop]].referenceCount.count # 0
   THEN RETURN;

   loc ← Loc[Of[oop]];
   last ← LastPointerOf[oop];

   WHILE oop # nil DO
   -- at this point oop is ready to peel off another reference
   -- or suspend for interrupt, with state minimized to oop and next

      IF InterruptPending[]
      THEN BEGIN
         RealSmash[loc, last];  -- make sure oop remembers where to resume
         SaveOopAndNext;
         DeferToMesa;  -- let Mesa service the interrupt
         RestoreOopAndNext;
         loc ← Loc[Of[oop]];
         last ← RealFetch[loc];
         END;

      IF last >= objectClassOffset
      THEN BEGIN
         ref: Oop ← RealFetch[loc + last];

         IF OopIsSmallInteger[ref]
         OR OtEntry[Of[ref]].referenceCount.count = maxCount
         THEN last ← last - 1
         ELSE BEGIN
            RefD[ref];

            IF OtEntry[Of[ref]].referenceCount.count = 0
            THEN BEGIN
               -- remember where to resume and how to go back up tree
               RealSmash[loc, last];
               RealSmash[loc + last, next];

               -- change levels
               next ← oop;
               oop ← ref;

               -- regenerate locals
               loc ← Loc[Of[oop]];
               last ← LastPointerOf[oop];
               END
            ELSE last ← last - 1;
            END;
         END;

      -- only loops in case class gets deallocated
      -- multiple loop probably impossible or at least astronomically rare
      WHILE last < objectClassOffset
```

```
    DO
       otPtr: LONG POINTER TO OTEntry = Of[oop];
       size: CARDINAL = RealFetch[loc + objectSizeOffset];

       -- adjust OTEntry for the oop
       otPtr.purpose ← free;
       otPtr.referenceCount.count ← maxCount;

       -- and link its chunk into free list (and update words, oops left)
       AddToProperFreeChunkList[
          oop: oop,
          address: Address[otPtr],
          size: size;

       -- go back up the tree
       oop ← next;
       loc ← Loc[Of[oop]];
       last ← RealFetch[loc];
       next ← RealFetch[loc + last];
       last ← last - 1;
       ENDLOOP;

    ENDLOOP;

END;
```