CONVENTIONS AND GUIDELINES FOR CDCNET COMMANDS

ARH6542

DRAFT

B. S. Sekhon
09/05/84

DISCLAIMER: This document is an internal working paper only. It is unapproved and subject to change, and does not necessarily represent any official intent on the part of Control Data Corporation.

REVISION RECORD			
Revision	Description	Author	Date
A	Original version,	BSS	07/25/84
B	Update to resolve pre-TDRB comments	BSS	08/06/84
C	Update to resolve TDRB comments	BSS	09/05/84

Table of Contents

1.0 INTRODUCTION

This document addresses the subject of conventions and guidelines for the CDCNET commands. It consists of three sections. The first section describes the specific conventions and guidelines for the CDCNET commands. The second section identifies the major groups in which the CDCNET commands may be divided. This section also includes the names of objects, verbs and commands in each group and their purpose. The third section lists the changes to be made in the existing CDCNET commands so that they follow the proposed conventions and guidelines. This section also describes the reasons for these changes as well as provides a summary of the new names for the CDCNET commands, their verbs and objects.

The conventions and guidelines for the CDCNET commands are being established to meet the following objectives.

- The CDCNET commands should be easy to learn and easy to use. This should be accomplished by making commands and their parameter names as well as their use as consistent as possible.

- All CDCNET commands should be consistent with NOS/VE commands as much as possible. To achieve this, the CDCNET command conventions should be compatible with the NOS/VE command conventions.

- The rules to obtain abbreviated names for command and parameter names should be well defined with absolutely no deviations or exceptions. It should be always possible to derive the abbreviated name of a command from its full name by following the same rules every time.

2.0 CDCNET COMMAND CONVENTIONS AND GUIDELINES

2.0 CDCNET_COMMAND_CONVENTIONS_AND_GUIDELINES

A command is identified via its name. It is specified in terms of its name, a list of its parameters, relative position of these parameters within the command and the explicit value or type (e.g. integer) of value which may be specified for each parameter. The following are some general conventions. Subsequent sections provide specific conventions for command and parameter names and rules for their abbreviation.

1) All CDCNET commands should follow the NOS/VE SCL syntax.

2) The value of a command parameter should not be used to indicate the function to be performed by the command.

3) Within a single command every effort should be made to minimize the illegal combinations of values of two or more parameters. This will happen if command parmeters are not orthogonal. In some cases this type of situation may be avoided by defining a new parameter in place of the parameters which are not orthogonal and therefore can have values which conflict with each other.

4) Whenever possible, a single command should be defined to perform the same function on one or more elements of the same type. For example the same command ("DISPLAY_NETWORK_STATUS) should be defined to display the status of all network solutions instead of separate commands (like "DISPLAY_HDLC_NETWORK_STATUS") to display the status of each supported network solution. However, this convention should not be followed if its use violates the convention number 2, described above.

5) If a command can perform a function on one or more elements, and this can be indicated by using a singular or plural word in the command name, then such a command should be allowed to be known by two names. The only difference in the two names will be the use of singular or plural word. For example consider the command used to display the configured attributes of a terminal. If this command is designed in such a way that its user can specify (via a parameter) which configured attributes are to be displayed, then this command should be specified via the following two names.

DISPLAY_TERMINAL_ATTRIBUTE

```
DISPLAY_TERMINAL_ATTRIBUTES
```

Both of these command names will have the same abbreviation (i.e. DISTA) and the user will be free to use either of these commands or their abbreviation to display one or more attributes of the terminal.

2.1 COMMAND_NAME_CONVENTIONS

The following conventions and guidelines should be followed in selecting command names.

1) Each command name should consist of at least one and optionally up to two parts. The first part called the command verb should be a verb describing the operation performed by the command. It should be represented by a single word. This word should always be the first word in the command name. Each command name must include the command verb part.

   The second part called the command object should specify the object which is the target of the operation to be performed by the command verb. The command object will be always specified via an object name. Optionally it may also include an object qualifier. If present, the object qualifier will always precede the object name within the command name. All words in a command name will be concatenated with the under score(_) character. To illustrate the various parts of a command name consider the following examples.

   ```
   DEFINE_HDLC_TRUNK
   DISPLAY_SYSTEM_STATUS
   START_DIRECTORY
   ```

   In the first command, "DEFINE" is the command verb and "HDLC_TRUNK" is the command object. Within this command object, "TRUNK" is the object name and "HDLC" is the object qualifier. In the second example "DISPLAY" is the command verb, "SYSTEM" is the object qualifier and "STATUS" is the object name. In the third command "START" is the command verb and "DIRECTORY" is the command object. In this case there is no object qualifier.

2) In specifying command names, the words used for object and object qualifier may be interchanged freely to make commands more meaningful or attractive to the human user. In other

words it is okay to use the object qualifier in one command as the object name in another command, as long as the commands are more meaningful or attractive to the human user. For example consider the following two commands.

```
DISPLAY_LINE_STATUS
START_LINE
```

In this example the word "LINE" is used as the object qualifier in the first command and as the object name in the second command.

3) The command names should be specified such that the same verb is used for same or similar functions. For example, it is possible to use the verbs DISPLAY, SHOW and LIST to display different attributes (e.g. status) of a CDCNET component. However, the command conventions will require selection and use of only one of these verbs.

4) If two commands provide opposite functions, then the verbs used in the two commands should be semantic opposites of each other. Some examples of semantically opposite verbs are:

   START AND STOP
   DEFINE AND CANCEL
   LOAD AND UNLOAD

5) The following rules will be used to determine the abbreviation for a command name.

   - Abbreviation for a command name will be generated by taking the first three characters of the command verb and concatenating it with first character of each word following the command verb in the command name. For example the abbreviation for the "DISPLAY_LINE_STATUS" command will be "DISLS".

   - If the command verb contains less than three characters, then an abbreviation will not be defined for the command.

   - If the names of two or more commands are such that they result in the same abbreviation, then one or more of these names will be changed so that the use of the above rules to determine abbreviations will result in unique abbreviations for all commands.

6) Names of all CDCNET commands will be equal to or less than 30 characters. This is a deviation from the NOS/VE command

conventions, which allows a command name to contain up to 31 characters. This deviation is due to the following reason.

The internal name of any CDCNET command may be up to 31 characters long. However, each command name is required to contain one special character ($), so that the associated entry point can be distinguisged from non-command (processor) entry points. This requirement has been put in place to prevent an attempt to execute any user typed in character string as a command. NOS/VE does not have this problem because of its command list feature.

The names of all parameters of CDCNET commands should be specified using the following conventions.

1) A parameter name will consist of one or more words concatenated with the underscore (_) character.

2) If same parameter is used in more than one command then the same name will be used for this parameter in all commands.

3) If similar parameters are used in more than one command then the same name will be used for such parameters in all commands, unless the names are used to convey the differences in the similar parameters.

4) The size of a parameter name for any CDCNET command will not exceed 31 characters.

5) A parameter name will be abbreviated by using the first character of each word in the parameter name. For example the abbreviation for the parameter name "LINE_SPEED" will be "LS".

6) For a given command, the names of all parameters will be selected so that no two names will result in the same abbreviation.

7) If value of a parameter in one command is to be used as the name of a parameter in another command, then the same name will be used for the parameter value as well as the parameter name. For example consider the command "DEFINE_COMMAND_ENVIRONMENT" used to define the command environment for an operator. This command can have "ce" or "command_echo" as a name of one of the parameters. The command "DISPLAY_COMMAND_ENVIRONMENT" used to display the command environment has a parameter called "do" or "display_options". One valid value for this parameter can be used to display the value of the "command_echo" parameter as

set in the "DEFINE_COMMAND_ENVIRONMENT" command. The keyword name for this value should be "ce" or "command_echo", i.e. same as the name of the corresponding parameter name.

2.2 PARAMETER_VALUE_CONVENTIONS

The values which may be specified for the command parameters will always be of a well known type. The following is a list of valid types.

- Integer
- Name
- String
- Boolean
- keyword
- List type
- Record type
- Union type (Any)
- Application type

These value types are described next using the NOS/VE SCL metalanguage rules. These rules are described in section 5 of this document. Most of the material in this section has been extracted from the NOS/VE command writer's guide.

The description of various types is preceded by a definition of various types of constants which are used in the definition of above types.

2.2.1 DEFINITION OF CONSTANTS

2.2.1.1 INIEGERS

An integer constant is a sequence of digits, the first of which must be a decimal digit, optionally prefixed by a sign and optionally suffixed by a radix enclosed in parentheses. The constant must be delimited at both ends. No spaces are permitted between the digits and the radix specification.

```
<integer> ::= [<sign>] <unsigned integer>

<unsigned integer> ::= <digit> [<hex digit>]...  [<(> <radix> <)>]

<sign> ::= <+¦-> [<sp>]

<+¦-> ::= + ¦ -

<hex digit> ::= <digit> ¦ A ¦ B ¦ C ¦ D ¦ E ¦ F

                        ¦ a ¦ b ¦ c ¦ d ¦ e ¦ f

<radix> ::= <unsigned decimal>

<unsigned decimal> ::= <digit>...
```

An integer constant can be expressed in any radix between two and sixteen. When the radix specification is omitted, ten (decimal) is assumed. Besides ten (decimal), the most common radix values are sixteen (hexadecimal), eight (octal) and two (binary). No distinction is made between lower and upper case hex digits by the CDCNET command parser.

When a radix greater than ten is specified, a leading zero digit may be required to ensure that the constant begins with a decimal digit.

This is to avoid ambiguity between, for example, the sixteenth element of the array whose name is A1 given by A1(16), and the hexadecimal representation of the decimal value 161 denoted by 0A1(16).

Spaces may not appear between the sign and unsigned integer in certain contexts, most notably in expressions for parameter values. See the syntax definition for numeric expressions for details.

Example: 63, 77(8), -63(10), 3f(16), 111111(2), 0abc(16)

2.2.1.2 NAME_CONSTANT

A name is a sequence of from 1 to 31 alphanumeric characters the first of which must not be a digit and which must be delimited at both ends.
In a name the case of letters is irrelevant and all lower case letters appearing in a name are "folded" to their upper case counterparts.

```
<name> ::= <alphabetic char> [<alphanumeric char>]...

<alphanumeric char> ::= <alphabetic char> ¦ <digit>

<alphabetic char> ::= <letter>
                    ¦ <special alphabetic char>
                    ¦ <international alphabetic char>

<letter> ::= A ¦ B ¦ C ¦ D ¦ E ¦ F ¦ G ¦ H ¦ I ¦ J ¦ K ¦ L ¦ M
           ¦ N ¦ O ¦ P ¦ Q ¦ R ¦ S ¦ T ¦ U ¦ V ¦ W ¦ X ¦ Y ¦ Z
           ¦ a ¦ b ¦ c ¦ d ¦ e ¦ f ¦ g ¦ h ¦ i ¦ j ¦ k ¦ l ¦ m
           ¦ n ¦ o ¦ p ¦ q ¦ r ¦ s ¦ t ¦ u ¦ v ¦ w ¦ x ¦ y ¦ z

<special alphabetic_char> ::= # ¦ $ ¦ @ ¦ _

<international alphabetic char> ::= _[_ ¦ \ ¦ _]_ ¦ ^
                                  ¦ ` ¦ { ¦ _¦_ ¦ } ¦ ~

<digit> ::= 0 ¦ 1 ¦ 2 ¦ 3 ¦ 4 ¦ 5 ¦ 6 ¦ 7 ¦ 8 ¦ 9
```

Example: x, $dAtE, this_is_a_semi_long_name

2.2.1.3 STRING CONSTANT

A string constant is any (possibly empty) sequence of ASCII characters enclosed in apostrophes (single quote marks). The apostrophes are not part of the string but serve as delimiters. To include an apostrophe in the string, two consecutive apostrophes are used. The string must be delimited at both ends.

```
 <string> ::= ' [<string char>]... '

 <string char> ::= <any ascii character except '> ¦ ''
```

```
Example: 'This is a string.'
         'Here is an enclosed apostrophe '' character.'
         ''        "a null (empty) string"
         ''''      "a string containing only an apostrophe"
```

2.2.1.4 BOOLEAN CONSTANT

A boolean constant is represented by one of the names shown below. These names are interpreted as boolean values only in those contexts which require boolean values.

```
<boolean> ::= <true> ¦ <false>

<true> ::= TRUE ¦ YES ¦ ON

<false> ::= FALSE ¦ NO ¦ OFF
```

2.2.2 DEFINITION OF TYPES

2.2.2.1 INTEGER_TYPE

```
<integer type> ::= INTEGER [<sp> <subrange>]

<subrange> ::= <min integer> <..> <max integer>

<min integer> ::= <integer expression>

<max integer> ::= <integer expression>
```

The subrange specification can be used to restrict the range of values applicable for the type being defined. The value of <min integer> must be less than or equal to that of <max integer>.

2.2.2.2 NAME_TYPE

```
<name type> ::= NAME [<sp> <name size>]

<name size> ::= [<min name size> <..>] <max name size>

<min name size> ::= <integer expression>

<max name size> ::= <integer expression>
```

The name size specification can be used to restrict the length of names applicable for the type being defined. The value of <min name size> must be less than or equal to that of <max name size> and both must be in the range 1..31. If <min name size> is omitted, 1 is assumed. If <max name size> is omitted, 31 is assumed.

2.2.2.3 STRING_TYPE

```
<string type> ::= STRING [<sp> <string size>]

<string size> ::= <min string size> <..> <max string size>
                ! <fixed string size>
```

```
<min string size> ::= <integer expression>

<max string size> ::= <integer expression>

<fixed string size> ::= <integer expression>
```

The string size specification can be used to restrict the length of strings applicable for the type being defined. The value of <min string size> must be less than or equal to that of <max string size> and both must be in the range 0..$max_string. If the string must be of one particular length, the <fixed string size> specification can be used to give values for both <min string size> and <max string size>. If the <string size> specification is omitted, 0 is assumed for <min string size> and $max_string for <max string size>.

2.2.2.4 BOOLEAN_TYPE

```
   <boolean type> ::= BOOLEAN
```

2.2.2.5 KEYWORD_TYPE

The keyword type is generally used for designating a set of options.

```
<keyword type> ::= KEY <sp!nl>
                     <keyword groups>
                   KEYEND

<keyword groups> ::= <keyword group> <,!sp!nl>
                     [<keyword group> <,!sp!nl>]...

<keyword group> ::= <(> <keyword> [<,!sp> <keyword>]...  <)>
                  ! <keyword>

<keyword> ::= <name>
```

2.2.2.6 LIST_TYPE

Lists should be used for structuring data whose elements are, in general, to be accessed sequentially, or in cases where the actual number of elements in the list will vary from one usage to the another.

```
<list type> ::= LIST [<list type qualifier>]

<list type qualifier> ::=
      [<sp> <list size qualifier>] <sp> OF <sp> <type expression>

<list size qualifier> ::= <min list size> <..> <max list size>

<min list size> ::= <integer expression>

<max list size> ::= <integer expression>
```

The list must have at least <min list size> elements. If <min list size> is not specified, zero is assumed. The list may not have more than <max list size> elements. If <max list size> is not specified, there is no limit to how many elements the list may have.

The list type qualifier may be omitted when defining a type for a parameter of a command. The omission means that the parameter may be passed a list with any element type.

2.2.2.7 RECORD_TYPE

Records provide a structuring mechanism for grouping data items of different types together. Each item is called a field of the record and is referenced via its field name.

```
<record type> ::= RECORD <sp|nl>
                    <field definition> <,|sp|nl>
                    [<field definition> <,|sp|nl>]...
                  RECEND

<field definition> ::=
      <field name> [<sp|nl>] <:> [<sp|nl>] <type expression>
```

2.2.2.8 UNION_TYPE

The union type provides for the case where any one of a number of types is applicable.

```
<union type> ::= ANY [<union type qualifier>]

<union type qualifier> ::= <sp> OF <sp!nl>
                             <member definition> <,!sp!nl>
                             [<member definition> <,!sp!nl>]...
                           ANYEND

<member definition> ::= <type>
```

If the union type qualifier is omitted the union consists of all possible types.

The order in which the members of the union type are defined is significant. If an expression for a union type can be successfully interpreted for more than one of its member types, it is given the first such interpretation.

```
Example:  var
            x: any of
              name
              file
            anyend
          varend

          x = fred
          display_value $type(x)
          NAME
          x = $work.fred
          display_value $type(x)
          FILE
```

2.2.2.9 APPLICATION_TYPE

Application types may be defined to deal with those situations where no other type known by the CDCNET command parser can be used.

```
<application type> ::=
     APPLICATION [<sp> <application value evaluator name>]
```

The application value evaluator designates a procedure supplied by the application (product, utility, user program, etc.) to parse and evaluate an expression of the particular application type. This procedure assumes total responsibility for the evaluation of the "expression" for the application value.

2.2.3 MAXIMUM SIZE OF PARAMETERS OF TYPE NAME

Even though the definition of type name allows one to have names whose maximum size may vary between 1 and 31 characters, all name type parameters (in CDCNET commands) which are externally visible to an end user or a CDCNET operator must support a maximum size of 31 characters. In other words the size of all externally visible name type parameters may range from 1 to 31 characters.

2.3 PARAMETER_NAME_ORDERING

The following rules are used to position values of parameters when a command is used. These rules are being provided here as things to be kept in mind when specifying a command and its parameters. Other than that this information has no purpose in this document.

Parameters for a command are specified as a seauence of individual parameters separated by commas or spaces. Parameters may be specified positionally or by name. When a parameter is not specified by name its position is taken to be one beyond that of the preceeding parameter. The significance of explicitly omitting a parameter (e.g. by placing two commas together) is only to establish the position of the next parameter. Giving a parameter by name has the effect of "tabbing" to that parameter position.

2.4 COMMAND_DESCRIPTION_CONVENTIONS

This section describes the conventions to be used to describe a command in the internal documents like the functional and network ERSs. These conventions should not be used in the external documents like a reference manual or a user guide. These conventions have been established to facilitate extraction of most of the command description from the source code for the associated command processor. As a matter of fact, the conventions to describe the command parameters are identical to the rules for the specification of the parameter description table or PDT.

The following is a line by line description of these conventions. It is followed by an example showing the use of these conventions.

1) The first line of the description contains the command name, its alias if any, and its abbreviated name, each separated by a comma (,) and a space.

2) The first line is followed by a brief English language description of the purpose of the command.

3) The description of the command purpose is followed by a blank line.

4) This blank line is followed by the parameter description table (PDT). The following is the general format in which the PDT is described. The subsequent steps describe the specific conventions to describe the PDT.

```
 PDT varaiable_name_pdt(
     command parameter definition
     command parameter definition)
```

5) The string "varaiable_name" in the first line of the PDT is identical to the abbreviated name for the command being described.

6) If the command does not contain any parameters, then the left parenthesis in the first line of the PDT specification is omitted.

7) The first line of the PDT specification and specification of the definition of the first command parameter should be separated by one blank line.

2.0 CDCNET COMMAND CONVENTIONS AND GUIDELINES
2.4 COMMAND DESCRIPTION CONVENTIONS

8)The following is a specification of the command parameter definition in the NOS/VE SCL metalanguage.

```
<command parameter definition> ::=
        <parameter names>
        [[<sp!nl>] <:> [<sp!nl>] <type>]
        [[<sp!nl>] = [<sp!nl>] <parameter default specification>]
      : <empty>

<parameter names> ::= <parameter name> [<,!sp> <parameter name>]...

<parameter default specification> ::= $REQUIRED
                                    ! $OPTIONAL
                                    ! <value>
```

The following is somewhat less abstract method to specify the command parameter definition within the PDT definition.

```
parameter names : type_specification = default specification
```

9) The parameter names portion of the parameter definition contains the name of the parameter, its alias if any, and its abbreviated name; each separated by a comma (,) or a space.

10) The type_specification portion of the parameter specifies the type of the parameter to be one of the valid types for the CDCNET command parameters, as described in section 2.3.2. The following are some examples of type_specifications.

- boolean - specifies the parameter to be of type boolean.
- integer - specifies the parameter to be of type integer.
- integer 0..7 - specifies the parameter to be of type integer whose value can range from 0 to 7.
- string - specifies the parameter to be of type string.
- string 3..21 - specifies the parameter to be of type string whose size may range from 3 to 21 characters.
- string 3 - specifies the parameter to be of type string whose size is fixed and is equal to 3 characters.
- name - specifies the parameter to be of type name.

- name 1..5 - specifies the parameter to be of type name whose size may range from 1 to 5 characters.

- list of name - specifies the parameter to be a list of names (with no upper limit on the number of names in the list)

- list 2..12 of name - specifies the parameter to be a list of names which may contain between 2 and 12 names.

- any of integer name - specifies the parameter to be either of type integer or name.

- list of any of name integer - specifies the parameter to be a list of either names or integers.

11) The type_specification portion of each command parameter definition should be intended so that this portion starts in the same column for all parameters.

12) The default_specification portion of the parameter definition determines whether a parameter must be included as one of the command parameters, or can be omitted and if so what its default value is, if any. If no default specification is given, it is assumed to be $OPTIONAL.

$REQUIRED specifies that the parameter must be supplied when the command is used.

$OPTIONAL specifies that the parameter may be omitted when the command is used, and if omitted no default value is assigned to the parameter.

If a value is given for the default_speecification, the parameter may be omitted and if omitted, the specified value is assigned to the parameter.

12)The description of the definition of one command parameter may be spread over several lines to make it more readable. This will be specially desirable if the parameter is of type key and several values for the key need to be included in the definition. If multiple lines are to be used to specify the definition of a single command parameter, then the second and subseauent lines should be intended so that they all start in the same column. This column should be one column position to the right of the column in which the last character of the type field resides.

13)The character (:) which separates the parameter name and the type specification portions should be present in the same column for all parameter definitions.

14) The notation of two dots (..) to indicate line continuation should be used as necessary.

15) The description of the PDT should be followed by a brief English language description of each command parameter.

The following example will illustrate the use of above rules.

DEFINE_LINE, DEFL

This command is used to create the initial definition of a communication line. It describes the hardware address of the line and associates a logical name with it. It also specifies the values of various configurable operational parameters for the line.

```
PDT defl_pdt(
    line_interface_module,lim     : integer 0..7 = $REQUIRED
    port_number,pn                : integer 0..4 = $required
    tip_type,tp                   : key asynctip
                                        hasptip
                                        sdlc3270 = asynctip
    terminal_config_procedure     : name = $OPTIONAL
    line_type                     : key switched,dedicated = dedicated
    framing_type                  : key async,sync,sdlc,hdlc = async
    carrier_type                  : key constant,controlled = controlled
    line_speed                    : key 110,150,300,600,1200,2400
                                        4800,9600,19200 = 300
    auto_recognition              : key none,s,sc,scp = s
    connection_connect_timeout    : integer 20..1000 = 100
    connection_disconnect_timeout: integer 4..1000 = 40
    user_connection_limit         : integer 1..16 = 3
    transmission_block_size       : integer 80(16)..0fff(16) = 80(16)
```

The above definition will be followed by a brief description of individual parameters.

3.0 CDCNET_COMMANDS

The CDCNET commands can be grouped into two major groups. The first group includes commands which are available to the end user accessing the CDCNET through a terminal device. The second group includes commands which are available to a CDCNET operator and the network analyst to manage the CDCNET. This section provides a very brief description of commands in these groups. Wherever appropriate, the command verbs and objects are also described separately.

3.1 TERMINAL_USER_COMMANDS

3.1.1 OBJECTS

The following is a list and brief description of the objects used in the terminal user commands.

TYPE	DESCRIPTION
CONNECTION(S)	Used to specify an association between a user and an application to which the user is connected to or wants to connect to.
WORKING_CONNECTION	Used to indicate the primary or currently in use connection if more than one connections exist at a given time.
CONNECTION_ATTRIBUTES	Used to indicate attributes of a connection.
TERMINAL_ATTRIBUTES	Used to indicate the attributes of a terminal.
STRING	Used to specify a data string which may be obtained from the terminal user.
LINE	Used to specify a data line which may be sent to the terminal user.

3.1.2 VERBS

The following is a list and brief description of the verbs used in the terminal user commands.

TYPE	DESCRIPTION
CREATE	This verb is used to create a connection.
DELETE	This verb is used to delete a connection.
CHANGE	This verb is used to change one or more configurable operational parameters of a connetion or a terminal. It is also used to select a connection as the working or primary connection.
DISPLAY	This verb is used to display the list of connections, attributes of a connection and attributes of a terminal.
GET	This verb is used to obtain a string from the terminal user.
PUT	This verb is used to send a data line to the terminal user.
DO	This verb is used to initiate processing of a terminal user procedure.
HELP	This verb is used to establish a connection with a help service.

3.1.3 COMMANDS

The following is a list and brief description of the terminal user commands.

NAME	DESCRIPTION
CREATE_CONNECTION	Used to create a new connection.
DISPLAY_CONNECTIONS	Used to display the current list of

connections.

CHANGE_WORKING_CONNECTION Used to select one of the connections as the working connection.

DELETE_CONNECTION Used to delete a connection.

DISPLAY_CONNECTION_ATTRIBUTES Used to display the attributes of a connection.

DISPLAY_CONNECTION_ATTRIBUTE Used to display a single attribute of a connection.

CHANGE_CONNECTION_ATTRIBUTES Used to change the attributes of a connection.

DISPLAY_TERMINAL_ATTRIBUTES Used to display the attributes of the user terminal.

CHANGE_TERMINAL_ATTRIBUTES Used to change the attributes of the user terminal.

GET_STRING Used to obtain a string from a user's terminal as input.

PUT_LINE Used to send a line to the terminal user.

HELP Used to establish a connection with a service which provides help to a terminal user.

DO Used to initiate processing of a terminal user procedure.

3.2 CDCNET MANAGEMENT COMMANDS

This section describes the various commands used to manage the CDCNET network. These commands have been further divided into sub-groups. Commands in each sub-group are described in separate sub-sections.

3.2.1 COMMANDS TO MANAGE EXTERNAL RESOURCES

Communication lines, ethernet and C170 channels connected to a DI are viewed as its external resources. The commands in this group are concerned with the definition and control of these resources as well as the software directly responsible to interface with them. The commands in this group and their verbs and objects are described next.

3.2.1.1 OBJECTS

The following is a list and brief description of the object names used in the terminal user commands.

TYPE	DESCRIPTION
TRUNK	A trunk is a logical entity. It consists of a physical medium (e.g. ethernet, communication line) and the link layer software used to interconnect two or more CDCNET systems.
NET	A net or network is a logical entity. It consists of a trunk or an X.25 virtual circuit and the definition (e.g. cost, network identification) which lets it to be used as a network solution.
LINE	A line is used to connect a terminal or a unit record device to a CDCNET system. It consists of a communication line and the Terminal Interface Process software needed to transfer data on it.
TERMINAL	This object is used to identify a user terminal.

A trunk or a net can be qualified by its type. The following are the valid qualifiers for these objects.

- HDLC
- CHANNEL
- ETHER

- X25

3.2.1.2 VERBS

The following is a list and brief description of the verbs used in the terminal user commands.

TYPE — DESCRIPTION

DEFINE — This verb is used to create the initial definition of a line, trunk, a network solution or a Terminal Interface Process software (TIP).

CHANGE — This verb is used to change one or more configurable operational parameters of a line, trunk, a network solution or a TIP.

CANCEL — This verb is used to remove or cancel the definition of a line, trunk or a network solution.

START — This verb is used to initialize the hardware and software needed to make a line, trunk or a network solution operational. The term operational as used here implies the state in which an element (e.g. line) can be used for data transfer.

STOP — This verb is used to change the state of a line, trunk or a network solution so that it is no longer operational.

3.2.1.3 COMMANDS

3.2.1.3.1 DEFINITION_AND_CHANGE_COMMANDS

These commands have the following generic format:

```
DEFINE_XXXX

CHANGE_XXXX
```

Where the string XXXX represents the object to be defined or changed. The following is a list of all definition and change commands.

DEFINE_HDLC_TRUNK	CHANGE_HDLC_TRUNK
DEFINE_ETHER_TRUNK	CHANGE_ETHER_TRUNK
DEFINE_CHANNEL_TRUNK	CHANGE_CHANNEL_TRUNK
DEFINE_X25_TRUNK	CHANGE_X25_TRUNK
DEFINE_HDLC_NET	CHANGE_HDLC_NET
DEFINE_ETHER_NET	CHANGE_ETHER_NET
DEFINE_CHANNEL_NET	CHANGE_CHANNEL_NET
DEFINE_X25_NET	CHANGE_X25_NET
DEFINE_LINE	CHANGE_LINE
DEFINE_TERMINAL	CHANGE_TERMINAL
DEFINE_TIP	CHANGE_TIP

The following is a brief description of the define and change commands.

The define commands are used to create the initial definition of the element being defined. This initial definition includes the identification of the hardware resources needed to support the element being defined, values of all configurable operational parameters which are required as well as the values of optional configurable operational parameters, for whom non default values are to be used. This definition also associates a logical name with the element being defined. All references to this element in the subsequent commands are made via this logical name.

These commands are processed by acquiring ownership of the hardware resource (if any) needed to support the element being defined and creating the data structure(s) needed to store its operational parameters.

The change commands are used to change values of one or more configurable operational parameters of the specified element. These commands require that the element whose operational parameters are to be changed must have been defined previously via the define command. This requirement is enforced by requiring the command parameters to specify the logical name of the element defined in the corresponding define command.

It should be noted that a change command can be used as often as necessary. Also the change command changes the values of only those parameters which are explicitly identified in the change command.

3.2.1.3.2 START, STOP AND CANCEL COMMANDS

This group includes the following commands.

START_TRUNK
START_NET
START_LINE
CANCEL_TERMINAL

STOP_TRUNK
STOP_NET
STOP_LINE

CANCEL_TRUNK
CANCEL_NET
CANCEL_LINE

The following is a brief description of the start, stop and cancel commands.

The start commands are used to make the specified element operational. The term operational is used here to imply the situation or state when the element is available for data transfer. The start commands have only one parameter. This parameter identifies the element to be started by its logical name. The processing of start commands usually involves initialization and start up of the software and hardware responsible to support data transfer on the element being started.

The stop commands are used to change the state of an element so that it is no longer capable of supporting data transfer. The stop commands have only one parameter. This parameter identifies the element to be stopped by its logical name. The processing of stop commands usually involves termination of the software responsible to support data transfer on the element being stopped.

The cancel commands are used to delete the definition of the specified element. These commands, among other things, delete the data structures created for the element by the corresponding define command. The cancel commands have only one parameter. This parameter identifies the element, whose definition is to be deleted by, its logical name.

3.2.2 COMMANDS TO MANAGE SOFTWARE RESOURCES

This section describes the commands used to manage software resources in a CDCNET system. Software resources include individual software elements or groups of software elements which together provide a specific service or a function.

The commands in this group are concerned with loading and unloading, start up and termination of software components as well making changes in their configurable operational parameters.

3.2.2.1 OBJECTS

The object names in the commands used to manage the CDCNET software are either the names of individual software elements or functions (e.g. X.25_INTERFACE) provided by a group of software elements. The following is a list of the object names used in the commands used to manage the software resources.

- C170_INTERFACE
- C170_BATCH_GW
- C170_IVT_GW
- X25_INTERFACE
- X25_GW
- DIRECTORY
- DSA
- DEVICE_MANAGER
- XNSSP_TRANSPORT
- CDNA_TRANSPORT

3.2.2.2 VERBS

The following is a list and brief description of verbs used in the commands used to manage the software resources.

TYPE	DESCRIPTION
LOAD	Used to load an individual software module in a CDCNET system.

START — Used to set configurable options and start up either an individual software component or a group of software components which together provide a service or a function.

STOP — Used to stop execution of either an individual software component or a group of software components which together provide a service or a function.

CHANGE — Used to change configurable options of an individual software component or a group of software components which are already executing.

UNLOAD — Used to release the memory being used to store an individual software component which is not executing.

RETAIN — Used to identify software components which should not be automatically unloaded to free up memory.

3.2.2.3 COMMANDS

3.2.2.3.1 SOFTWARE LOAD, RETAIN AND UNLOAD COMMANDS

There is a single load command called "LOAD_MODULE". This command has a single parameter which is used to identify the software module to be loaded. This command is usd to load a software module in a CDCNET system.

There is a single retain command called "RETAIN_MODULE". This command has a single parameter which is used to identify the software module which is to be retained. This command is used to prevent automatic unloading of an inactive software module to free up memory.

There is a single unload command called "UNLOAD_MODULE". This command has a single parameter which is used to identify the software module to be unloaded. This command checks if the module to be unloaded is in use. If it is not being used, the memory used to store it is freed up.

3.2.2.3.2 SOFTWARE START UP,CHANGE AND STOP COMMANDS

The software start up, change and stop commands have the following generic names.

- START_XXXXX

- CHANGE_XXXXX
- STOP_XXXXX

Where the string XXXX identifies an individual software element or a service or function which is provided by a group of software elements.

START_C170_INTERFACE	CHANGE_C170_INTERFACE	STOP_C170_INTERFACE
START_C170_BATCH_GW	CHANGE_C170_BATCH_GW	STOP_C170_BATCH_GW
START_C170_IVT_GW	CHANGE_C170_IVT_GW	STOP_C170_IVT_GW
START_X25_INTERFACE	CHANGE_X25_INTERFACE	STOP_X25_INTERFACE
START_X25_GW	CHANGE_X25_GW	STOP_X25_GW
START_OSA	CHANGE_OSA	
	CHANGE_DIRECTORY	
	CHANGE_XNS_TRANSPORT	
	CHANGE_CDNA_TRANSPORT	

The following is a brief description of the start, change and stop commands.

The start command is used to start up one or more software elements. The parameters on this command are used to provide the values of required operational parameters as well as the optional operational parameters for which non-default values are to be used. If the software element or elements which are to be started are not present in the system, these are loaded in the CDCNET system and then started. There are certain software elements in a CDCNET system which are required for the system to do any useful work (e.g. accept and process a command). These software elements get started automatically when the system is loaded, and therefore no start commands are needed or defined for these elements.

The change command is used to change the values of one or more configurable operational parameters of a single software element or a group of software elements which are already executing. The parameters on this command are used to provide the values of operational parameters which are to be changed.

The stop command is used to stop execution of an individual software element or a group of software elements which are together providing a service or a function.

3.2.3 COMMANDS TO MANAGE LOGICAL ELEMENTS

Certain elements in a CDCNET system can not be designated as software or hardware resources or external resources. Some examples of these elements are the groups of log messages which may be emitted by a given system, or a definition of communities in the network. This section describes the commands which deal with such elements.

3.2.3.1 OBJECTS

The following is a list and brief description of objects used in the commands used to manage the logical elements in a CDCNET system.

NAME	DESCRIPTION
COMMUNITY_SEGMENT	Used to refer to a sub-set of a community, which includes a set of CDCNET systems which are connected to the same network solution and also are members of the same community.
COMMUNITY_MEMBERSHIP	Used in a command which defines a CDCNET system to be a member of a specified community.
LOG_COMMUNITY	Used to associate a community of CDCNET systems with the Independent LOG M-E which is responsible to process certain log messages from systems in this community.
CIM_INTERFACE	Used to refer to the interface between the DVM and CIM firmware.
ESCI_INTERFACE	Used to refer to the interface between the DVM and ESCI firmware.
SYSTEM	Used to refer to the CDCNET system as a whole.
FILE_TYPE	Used to identify the type of files, access to whom is supported by the Independent File Access M-E.
LOG_MESSAGE	Used to refer to a log message which may be emitted from a CDCNET system.
ALARM_MESSAGE	Used to refer to an operator alarm which may be emitted from a CDCNET system.

3.0 CDCNET COMMANDS
3.2.3.1 OBJECTS

LOG_ENVIRONMENT — Used to refer to the list of log messages (and the associated list of log communities) whose emission was enabled during the system configuration, i.e. the list which does not include the changes made by any online commands after the system configuration has been completed.

ALARM_ENVIRONMENT — Used to refer to the list of alarm messages whose emission was enabled during the system configuration, i.e. the list which does not include the changes made by any online commands after the system configuration has been completed.

LOGGING:Used as the object in the command used to indicate the completion of the initial or configured definition of list of log messages which may be emitted from a given CDCNET system.

ALARMING:Used as the object in the command used to indicate the completion of the initial or configured definition of list of alarm messages which may be emitted from a given CDCNET system.

LOGGING_THRESHOLD — Used to refer to the count of an event which is used as a threshold to decide if a log message should be generated to report that event.

BOOT_DEFAULTS — Used to refer to the defaults (e.g. version of the object library) used to load a CDCNET system.

EXCEPTION_SYSTEMS — used to identify CDCNET systems which should be loaded from a non-default object library.

3.2.3.2 VERBS

The following is a list and brief description of the verbs used in the commands used to manage the logical elements in a CDCNET system.

TYPE DESCRIPTION

DEFINE This verb is used to create initial definition of a logical element.

CHANGE This verb is used to change the definition of a logical element.

CANCEL This verb is used to delete the definition of a logical element.

ADD This verb is used to enable the emission of a log or alarm message. It is also used to add a file type to be supported by the Independent File Access M-E.

DELETE This verb is used to disable the emission of a log or alarm message. It is also used to delete a file type to be supported by the Independent File Access M-E.

RESTORE This verb is used to replace the current list of log and alarm messages whose emission is enabled with the corrsponding list defined as a part of the system configuration.

START This verb is used to indicate completion of the initial or configured definition of log and alarm messages which may be emiited from a given CDCNET system.

RESET This verb is used to force a CDCNET system to reset itself.

3.2.3.3 COMMANDS

The following is a list of commands which deal with the logical elements in a CDCNET system.

- DEFINE_COMMUNITY_SEGMENT
- CANCEL_COMMUNITY_SEGMENT
- DEFINE_LOG_COMMUNITY
- CANCEL_LOG_COMMUNITY
- DEFINE_COMMUNITY_MEMBERSHIP
- CANCEL_COMMUNITY_MEMBERSHIP
- DEFINE_SYSTEM
- CHANGE_SYSTEM
- RESET_SYSTEM
- DEFINE_CIM_INTERFACE

- DEFINE_ESCI_INTERFACE
- CHANGE_LOGGING_THRESHOLD
- CHANGE_EXCEPTION_SYSTEMS
- CHANGE_BOOT_LOAD DEFAULTS
- ADD_FILE_TYPE
- DELETE_FILE_TYPE
- ADD_LOG_MESSAGE
- DELETE_LOG_MESSAGE
- ADD_ALARM_MESSAGE
- DELETE_ALARM_MESSAGE
- START_LOGGING
- START_ALARMING
- RESTORE_LOG_ENVIRONMENT
- RESTORE_ALARM_ENVIRONMENT

3.2.4 DISPLAY COMMANDS

The display commands are used to display various attributes of different elements of a CDCNET system. Some examples of the types of attributes which may be displayed are configurtion options, status, metrics (statistics), list of log messages whose emission is enabled, etc.

All disply commands have a parameter called the "display_options". The value of this parameter can be used to provide a list of things (e.g. a specific configuration option) which are to be displayed.

3.2.4.1 OBJECTS

The following is a list and brief description of objects used in the display commands.

NAME	DESCRIPTION
STATUS	Status of a CDCNET element
METRICS	Statistics collected for an element or data traffic in a CDCNET system.
OPTIONS	Configuration options of an element in a CDCNET system.

LOG_ENVIRONMENT

Used to refer to the list of log messages which may be emitted from a CDCNET system and the list of systems where they get logged.

ALARM_ENVIRONMENT

Used to refer to the list of operator alarms which may be emitted from a CDCNET system.

COMMAND_ENVIRONMENT

Used to refer to the command interface environment for a given CDCNET operator.

LOGGING_THRESHOLD

Used to refer to the count of an event which is used as a threshold to decide if a log message should be generated to report that event.

OPERATOR_LOG

Used to refer to the operator log which is used to leave messages for CDCNET operators.

ALARM_HISTORY

Used to refer to the list of operator alarms which are retained for subsequent viewing by the CDCNET operator.

BOOT_DEFAULTS

Used to refer to the defaults (e.g. version of the object library) used to load a CDCNET system.

EXCEPTION_SYSTEMS

used to identify CDCNET systems which should be loaded from a non-default object library.

PATHS

Used to refer to one or more paths between two CDCNET systems.

The STATUS, METRICS and OPTIONS objects can be further qualified by one of the following.

- Line, Trunk or a Network (network solution).
- Type of a trunk (e.g. HDLC_TRUNK) or a network solution (e.g. ETHER_NET).
- System
- Name of a software element (e.g. Directory)

The list of Display commands should be used to determine which of these qualifiers can be used with which object.

3.2.4.2 VERBS

The display commands have only one verb called "DISPLAY".

3.2.4.3 COMMANDS

The following is a list of the DISPLAY commands.

DISPLAY_LINE_STATUS
DISPLAY_TRUNK_STATUS
DISPLAY_NETWORK_STATUS
DISPLAY_TRANSPORT_STATUS
DISPLAY_DIRECTORY_STATUS
DISPLAY_SYSTEM_STATUS
DISPLAY_HARDWARE_STATUS

DISPLAY_LINE_METRICS
DISPLAY_TRUNK_METRICS
DISPLAY_NETWORK_METRICS
DISPLAY_TRANSPORT_METRICS
DISPLAY_DIRECTORY_METRICS
DISPLAY_SYSTEM_METRICS
DISPLAY_FAILURE_METRICS

DISPLAY_C170_INTERFACE_OPTIONS
DISPLAY_C170_BATCH_GW_OPTIONS
DISPLAY_C170_IVT_GW_OPTIONS
DISPLAY_X25_INTERFACE_OPTIONS
DISPLAY_X25_GW_OPTIONS
DISPLAY_DIRECTORY_OPTIONS
DISPLAY_OSA_OPTIONS
DISPLAY_XNS_TRANSPORT_OPTIONS
DISPLAY_CDNA_TRANSPORT_OPTIONS

DISPLAY_PATHS
DISPLAY_LOG_ENVIRONMENT
DISPLAY_ALARM_ENVIRONMENT
DISPLAY_COMMAND_ENVIRONMENT
DISPLAY_OPERATOR_LOG
DISPLAY_ALARM_HISTORY
DISPLAY_LOGGING_THRESHOLD
DISPLAY_LOGGING_THRESHOLDS

3.2.5 OPERATOR INTERFACE COMMANDS

This section describes the commands available to a CDCNET operator to manage his or her environment as well as to send commands to one or more CDCNET systems.

3.2.5.1 OBJECTS

The following is a list and brief description of objects used in the operator interface commands.

NAME	DESCRIPTION
COMMAND	Used as the object used in the command to send a command to one or more CDCNET systems.
RESPONSE	Used as the object in the command used to route responses for one or more commands to a file or another device.
COMMAND_ENVIRONMENT	Used as the object in the command used to define or change the operator interface environment.
OPERATOR_NOTE	Used as the object in the command used to add a note in the operator log.

3.2.5.2 VERBS

The following is a list and brief description of verbs used in the operator interface commands.

NAME	DESCRIPTION
SEND	Used to send a command to one or more systems.
BROADCAST	used to broadcast a command to all systems in one or

more communities.

DEFINE — Used to define or set up the operator interface environment.

CHANGE — Used to change one or more attributes of the operator interface environment.

DO — Used to invoke execution of commands from a command file.

ADD — Used to add a note in the operator log.

ROUTE — Used to route command rsponses to a file or a device (e.g. printer).

QUIT — Used to leave (i.e. exit from) the operator interface application.

3.2.5.3 COMMANDS

The following is a list of the operator interface commands.

SEND_COMMAND

BROADCAST_COMMAND

DEFINE_COMMAND_ENVIRONMENT

CHANGE_COMMAND_ENVIRONMENT

DO

ADD_OPERATOR_NOTE

ROUTE_RESPONSE

QUIT

3.2.6 FAILURE MANAGEMENT COMMANDS

This section describes the commands used to manage failures in a CDCNET system as well as to test various hardware and logical elements.

3.2.6.1 OBJECTS

The following is a list and brief description of objects used in the failure management commands.

NAME	DESCRIPTION
ELEMENT_STATE	Used in the commands used to change the state of a hardware element (e,g, a lim or a port).
TRAFFIC	Used in the command used to change the traffic rate on a trunk.
TEST	Used in the commands to start or stop a test.
PATH	Used in the command to test a path between two CDCNET systems.

The object "TEST" can be qualified with the element to be tested. Some examples of these qualifiers are MPB, SMM, PORT, etc. The command list in the following section shows all valid qualifiers.

3.2.6.2 VERBS

The following is a list and brief description of verbs used in the operator interface commands.

NAME	DESCRIPTION
START	Used to start a test.
STOP	Used to stop a test.

CHANGE — Used to change the state of a hardware element or rate of traffic on a given trunk.

3.2.6.3 COMMANDS

The following is a list of the commands used to manage failures in a CDCNET system.

NAME	DESCRIPTION
CHANGE_ELEMENT_STATE	
CHANGE_TRAFFIC	
START_LOCAL_LOOPBACK_TEST	STOP_LOCAL_LOOPBACK_TEST
START_REMOTE_LOOPBACK_TEST	STOP_REMOTE_LOOPBACK_TEST
START_PATH_TEST	STOP_PATH_TEST
START_MPB_TEST	STOP_MPB_TEST
START_CIM_TEST	STOP_CIM_TEST
START_ESCI_TEST	STOP_ESCI_TEST
START_LIM_TEST	STOP_LIM_TEST
START_PORT_TEST	STOP_PORT_TEST
START_SMM_TEST	STOP_SMM_TEST
START_CHANNEL_TEST	STOP_CHANNEL_TEST

3.2.7 MISCELLANEOUS COMMANDS

The following are some miscellaneous commands which do not fall in any of the command groups described earlier.

NAME	DESCRIPTION

3.0 CDCNET COMMANDS
3.2.7 MISCELLANEOUS COMMANDS

START_XXX_METRICS_COLLECTION

Used to start collection of statistics for a specified element. The string XXX ientifies the entity for which metrics collection is to be started.

STOP_XXX_METRICS_COLLECTION

Used to stop colection of statistics for a specified element. The string XXX ientifies the entity for which metrics collection is to be stopped.

START_XXX_METRICS_REPORTING

Used to start periodic reporting of statistics for a specified element. The string XXX ientifies the entity for which metrics reporting is to be started.

STOP_METRICS_REPORTING

Used to stop periodic reporting of statistics for a specified element. The string XXX ientifies the entity for which metrics reporting is to be stopped.

START_AUTO_RECOGNITION

Used to start automatic recognition of certain (e.g. line speed) attributes of a communication line.

4.0 CHANGES_IN_EXISTING_COMMANDS_DUE_TO_PROPOSED_CONVENTIONS

This section describes the changes needed in the current definition of the CDCNET commands so that they follow the proposed conventions and guidelines. These changes are described separately for the command verbs, objects and the command names.

This section is included in this document to provide some indication of the changes required in the current definition of the CDCNET commands because of the proposed conventions and guidelines. This section may be ignored as far as the review of the proposed conventions and guidelines is concerned.

4.1 CHANGES_IN_COMMAND_VERBS

This section describes the reasons for changes in the current command verbs. It also includes a list of current and proposed verbs along with a description of the proposed verbs. The following is a list of reasons for the proposed changes.

1) The "DISABLE" and 'DISPLAY' verbs have the same first three characters. Because of this the abbreviations for the commands which use these verbs, will be very similar. For example the abbreviations for commands "DISPLAY_LINE_STATUS" and "DISABLE"_LINE" will be "DISLS" and "DISL" respectively. In view of this the following changes are being proposed.

   The "DISABLE" verb should be replaced with the "STOP" verb. Then the command "DISABLE_LINE" will become "STOP_LINE". Most of the functions provided by the "ENABLE/DISABLE" verbs will be provided by the "START/STOP" verbs.

2) The "ACTIVATE/DEACTIVATE" verbs should be replaced with the "START/STOP" verbs, because these verbs are used to provide similar functions.

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS
4.1 CHANGES IN COMMAND VERBS

3) The verbs "INHIBIT" AND "TERMINATE" should be replaced with the "STOP" verb, because all of these provide similar functions.

4) The verbs "CHECK", "ECHO","LOOPBACK" and "EXECUTE" should be replaced with the verb "START". An object name (e.g. "PATH_TEST") should be used to specify the type of testing to be done.

5) The "CONFIGURE" verb should be replaced with the "DEFINE" verb so that the same verb is used to set or define all configuration parameters.

6) The "ENTER" verb should be replaced with the "ADD" because the verb "ADD" can be used equally well to describe the function being performed by the "ENTER" verb.

The following list uses the term "OLD NAME" for the current verb and the term "NEW NAME" for the proposed verb. Also included in this list is a brief description of the proposed verb.

OLD NAME ADD, ENABLE, ENTER
NEW NAME ADD
DESCRIPTION Used to add a file type to the list of files supported by the Independent File access M-E. This verb is also used to add a log/alarm message to the list of log/alarm messages to be transmitted by a given system, as well as to add a message in the operator log.

OLD NAME BROADCAST
NEW NAME Same
DESCRIPTION Used to broadcast a command to one or more destination communities.

OLD NAME CHANGE
NEW NAME Same
DESCRIPTION Used to change the hardware and software configuration parameters, to change state of a hardware element and to change traffic on a trunk.

OLD NAME CREATE
NEW NAME Same
DESCRIPTION Used to create a connection.

OLD NAME DELETE
NEW NAME CANCEL
DESCRIPTION Used to delete the definition of various elements

like a terminal device, a communication line, a trunk, a network solution, etc. It is also used to delete a log/alarm message from the list of log/alarm messages to be transmitted by a given system.

OLD NAME DELOAD
NEW NAME UNLOAD
DESCRIPTION Used to delete the information about the presence of a software module in the system and to free up the memory being used by that module.

OLD NAME DISPLAY
NEW NAME Same
DESCRIPTION Used to display configuration parameters, status and statistics for various hardware and software elements; to display the list of connections and their attributes; to display alarm history and the operator log for a given operator, to disply the operator's command environment as well as to display the list of log messages and alarms whose transmission is enabled.

OLD NAME DO
NEW NAME Same
DESCRIPTION Used to cause a terminal user or operator procedure to be processed.

OLD NAME FLUSH
NEW NAME STOP
DESCRIPTION Used to flush current statistics buffers.

OLD NAME GET
NEW NAME Same
DESCRIPTION Used to input a string from a terminal.

OLD NAME HELP
NEW NAME Same
DESCRIPTION Used to invoke Help utility.

OLD NAME LOAD
NEW NAME Same
DESCRIPTION Used to load a software module in a CDCNET system.

OLD NAME PUT
NEW NAME Same
DESCRIPTION Used to to output a string to the terminal user or to a service connected to the terminal user.

OLD NAME QUIT

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS

4.1 CHANGES IN COMMAND VERBS

NEW NAME Same
DESCRIPTION Used to quit or exit from the operator interface utility.

OLD NAME RETAIN
NEW NAME Same
DESCRIPTION Used to mark a software module to be unloadable via the DELOAD command.

OLD NAME ROUTE
NEW NAME Same
DESCRIPTION Used to route a command response to a file.

OLD NAME SEND
NEW NAME Same
DESCRIPTION Used to send a command to a list of specified systems.

OLD NAME SET, CONFIGURE
NEW NAME DEFINE
DESCRIPTION Used to set configuration parameters of software and hardware elements including communication lines; CDCNET system and terminal devices; to set various aspects of operator interface; to set connection attributes and to select the working connection.

OLD NAME ACTIVATE, ENABLE
NEW NAME START
DESCRIPTION Used to start auto recognition on a communication line as well as to start execution of a software component. Also used to start service on a communication line, trunk or a network solution; and to start statistics collection or reporting.

OLD NAME DEACTIVATE, DISABLE, INHIBIT, TERMINATE
NEW NAME STOP
DESCRIPTION Used to stop execution of a software component. Also used to stop service on a communication line, trunk or a network solution; and to stop statistics collection or reporting.

OLD NAME CHECK, ECHO, LOOPBACK, EXECUTE
NEW NAME START
DESCRIPTION Used to start execution of a diagnostics test. Also used to test a path or paths between two CDCNET systems.

4.2 CHANGES IN THE COMMAND OBJECT NAMES

This section provides a list of current object names and proposed changes to some of them. The following are some of the reasons for the proposed changes.

- Some of the current object names are too long and result in the command name size to exceed the 30 character limit.

- In some cases different names are used for the same object. One example of this is the use of words "DI" and "SYSTEM". There is no reason to have both of these as valid object names.

- The names "LINK" and "LINE" result in identical abbreviations for several commands(e.g DISPLAY_LINE_STATUS and DISPLAY_LINK_STATUS). In view of this the name "LINK" has been changed to "TRUNK". Also in order to maintain consistency names like "HDLC_LINK" and "MCI_LINK" have been changed to "HDLC_TRUNK" and "MCI_TRUNK".

The following is a list of current and proposed object names. Related object names have been grouped together.

4.2.1 OBJECT NAMES FOR NETWORK SOLUTIONS, TRUNKS, LINES AND TERMINALS

OLD NAME X25_NETWORK_SOLUTION_ATTRIBUTES
NEW NAME X25_NET
DESCRIPTION X25_NET is used as the object in the command used to define and change the configuration parameters(e.g cost, remote DTE address) for an X.25 virtual circuit to be used as a network solution.

OLD NAME NETWORK_SOLUTION_ATTRIBUTES
NEW NAME HDLC_NET,MCI_NET,ESCI_NET
DESCRIPTION These objects are used in the commands used to define and change the configuration parameters for network solutions using an HDLC line, channel interface and ethernet, respectively, as the communication media.

OLD NAME NETWORK_SOLUTION
NEW NAME NETWORK
DESCRIPTION NETWORK is used as the object in the commands used to display status, statistics and configuration

attributes of a network solution. A command parameter is used to specify one or more network solutions whose status, statistics or configuration attributes are to be displayed.

OLD NAME HDLC_LINK_ATTRIBUTES
NEW NAME HDLC_TRUNK
DESCRIPTION HDLC_TRUNK is used as the object in the command used to define and change the configuration attributes for layers 1 and 2 software and the needed hardware to support the HDLC line.

OLD NAME MCI_LINK_ATTRIBUTES
NEW NAME MCI_TRUNK
DESCRIPTION MCI_TRUNK is used as the object in the command used to define and change the configuration attributes for layers 1 and 2 software and the needed hardware to support the mainframe channel interface as a network solution or an interface to the C170 network products.

OLD NAME ETHERNET_LINK_ATTRIBUTES
NEW NAME ESCI_TRUNK
DESCRIPTION ESCI_TRUNK is used as the object in the command used to define and change the configuration attributes for layers 1 and 2 software and the needed hardware to support the use of ethernet as a network solution

OLD NAME LINE
NEW NAME Same
DESCRIPTION LINE is used as the object in the commands used to define and display the configuration attributes of a communication line. This object is also used in the commands used to display the status and statistics for a given communication line.

OLD NAME TERMINAL_DEVICE
NEW NAME TERMINAL
DESCRIPTION TERMINAL is used as the object in the command used to define and display the physical configuration of a terminal device, connected to a communication line.

4.2.2 OBJECT NAMES FOR SOFTWARE ELEMENTS

OLD NAME NETWORK_PRODUCTS_INTERFACE
NEW NAME C170_INTERFACE
DESCRIPTION C170_INTERFACE is used as the object in the command used to define the configuration attributes of the software used to provide an interface between a CDCNET system and the C170 network products. At a future time, it may also be used as the object in the commands used to display the status and statistics for the C170 network products interface software.

OLD NAME C170_TRANSPARENT_GATEWAY
NEW NAME C170_BATCH_GW
DESCRIPTION C170_BATCH_GW is used as the object in the command used to define the configuration attributes of the A to A gateway between CDCNET and C170 network products. At a future time, it may also be used as the object in the commands used to display the status and statistics for this gateway.

OLD NAME C170_INTERACTIVE_GATEWAY
NEW NAME C170_IVT_GW
DESCRIPTION C170_IVT_GW is used as the object in the command used to define the configuration attributes of the A to T gateway between CDCNET and C170 network products. At a future time, it may also be used as the object in the commands used to display the status and statistics for this gateway.

OLD NAME PACKET_LEVEL_ATTRIBUTES
NEW NAME X25_INTERFACE
DESCRIPTION X25_INTERFACE is used as the object in the command used to define the configuration attributes of the X.25 packet level software. At a future time, it may also be used as the object in the commands used to display the status and statistics for this software component.

OLD NAME X25_TRANSPARENT_GATEWAY
NEW NAME X25_GW
DESCRIPTION X25_GW is used as the object in the command used to define the configuration attributes of the gateway between CDCNET and an X.25 PDN. At a future time, it may also be used as the object in the commands used to display the status and statistics for this gateway.

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS
4.2.2 OBJECT NAMES FOR SOFTWARE ELEMENTS

OLD NAME TIP
NEW NAME Same
DESCRIPTION TIP is used as the object in the command used to define the configuration attributes of the TIP software. At a future time, it may also be used as the object in the commands used to display the status and statistics for this software component.

OLD NAME DIRECTORY_ATTRIBUTES
NEW NAME DIRECTORY
DESCRIPTION DIRECTORY is used as the object in the command used to define the configuration attributes of the DIRECTORY software. At a future time, it may also be used as the object in the commands used to display the status and statistics for this software component.

OLD NAME LOG_RETAIN_QUEUE_LIMIT, LOG_PRESERVE_QUEUE_LIMIT
NEW NAME None
DESCRIPTION The values of these objects will be hard coded in the Dependent LOG M-E code.

OLD NAME OSA_ATTRIBUTES
NEW NAME OSA
DESCRIPTION OSA is used as the object in the command used to define the configuration attributes of the Operator support application software. At a future time, it may also be used as the object in the commands used to display the status and statistics for this software component.

OLD NAME SOFTWARE_MODULE
NEW NAME None
DESCRIPTION

4.2.3 NETWORK AND SYSTEM LEVEL OBJECT NAMES

OLD NAME SYSTEM_ATTRIBUTES
NEW NAME SYSTEM
DESCRIPTION SYSTEM is used as the object in the commands used to define and display the configuration attributes of a CDCNET system.

OLD NAME INITIALIZATION_DEFAULTS

4.2.3 NETWORK AND SYSTEM LEVEL OBJECT NAMES

NEW NAME BOOT_DEFAULTS
DESCRIPTION This object is used in the command used to add information in the "exception" file about the default load file to be used to load the CDCNET systems not identified in the "exception" file.

OLD NAME SYSTEM_EXCEPTIONS
NEW NAME EXCEPTION_SYSTEMS
DESCRIPTION This object is used in the command used to add information in the "exception" file about systems which should not be loaded from the default load file.

OLD NAME PERIPHERAL_ATTRIBUTES
NEW NAME CIM_INTERFACE, ESCI_INTERFACE
DESCRIPTION These object are used in the commands used to define the interface parameters for CIM and ESCI firmware and the Device manager.

OLD NAME COMMUNITY
NEW NAME COMMUNITY_SEGMENT, COMMUNITY_MEMBERSHIP
DESCRIPTION This objects are used in the commands used to define a CDNA community.

4.2.4 OPERATOR INTERFACE RELATED OBJECT NAMES

OLD NAME ALARM_OUTPUT
NEW NAME None
DESCRIPTION

OLD NAME ALARM_HISTORY
NEW NAME Same
DESCRIPTION This object is used in the command used to display a certain number of previously received alarms at the operator's terminal

OLD NAME CONNECTED_MDI
NEW NAME None
DESCRIPTION

OLD NAME COMMAND_MDI
NEW NAME None
DESCRIPTION

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS
4.2.4 OPERATOR INTERFACE RELATED OBJECT NAMES

OLD NAME	SYNCHRONOUS_MODE
NEW NAME	None
DESCRIPTION	
OLD NAME	ASYNCHRONOUS_MODE
NEW NAME	None
DESCRIPTION	
OLD NAME	COMMAND
NEW NAME	Same
DESCRIPTION	This object is used in the command used by a CDCNET operator to send a command to one or more CDCNET systems.
OLD NAME	COMMAND_LOGGING
NEW NAME	None
DESCRIPTION	
OLD NAME	COMMAND_RESPONSE
NEW NAME	RESPONSE
DESCRIPTION	
DESCRIPTION	This object is used in the command used by a CDCNET operator to route command responses to a file.
OLD NAME	COMMAND_ECHO
NEW NAME	None
DESCRIPTION	
OLD NAME	OPERATOR_NOTE
NEW NAME	Same
DESCRIPTION	This object is used in the command used by a CDCNET operator to add a message in the operator's log.
OLD NAME	OPERATOR_LOG
NEW NAME	Same
DESCRIPTION	This object is used in the command used by a CDCNET operator to display the operator log at the operator's terminal.
OLD NAME	DEFAULT_DOMAIN
NEW NAME	None
DESCRIPTION	

4.2.5 DIAGNOSTICS RELATED OBJECT NAMES

OLD NAME ERROR_REPORTING_THRESHOLD

NEW NAME LOGGING_THRESHOLD

DESCRIPTION This object is used in the commands used to define and display the thresholds used for certain events to decide if these events should be logged.

OLD NAME HARDWARE_ELEMENT_STATE

NEW NAME ELEMENT_STATE

DESCRIPTION This object is used in the command used to change the state (e.g. maintenance) of a hardware element in a CDCNET system.

OLD NAME LINK_TRAFFIC and LINK_TRAFFIC_RATE

NEW NAME TRAFFIC

DESCRIPTION This object is used in the command used to change the amount of traffic on a given network solution

OLD NAME PATH AND PATHS

NEW NAME Same

DESCRIPTION These objects are used in the command used to test and display one or more paths between two CDCNET systems.

OLD NAME DIAGNOSTICS

NEW NAME XXX_TEST,where XXX represents the element to be tested.

DESCRIPTION These objects are used in the commands used to execute diagnostic tests on a hardware element in a CDCNET system.

4.2.6 MISCELLANEOUS OBJECT NAMES

OLD NAME AUTO_RECOGNITION

NEW NAME Same

DESCRIPTION AUTO_RECOGNITION is used as the object in commands used to start and stop auto recognition of line attributes(e.g.speed) by the terminal support software.

OLD NAME FILE_TYPE AND FILE_TYPES

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS
4.2.6 MISCELLANEOUS OBJECT NAMES

NEW NAME Same
DESCRIPTION These objects are used in the commands used to define and display file types supported by a given independent file access ME.

OLD NAME STATISTICS
NEW NAME METRICS
DESCRIPTION METRICS is used as the object in commands used to stop reporting of statistics for one or more CDCNET elements.

OLD NAME STATISTICS_COLLECTION
NEW NAME METRICS_COLLECTION
DESCRIPTION METRICS_COLLECTION is used as the object in commands used to control the collection of statistics for one or more CDCNET elements.

OLD NAME STATISTICS_PERIODIC_REPORTING
NEW NAME METRICS_REPORTING
DESCRIPTION METRICS_REPORTING is used as the object in commands used to control the periodic reporting of statistics for one or more CDCNET elements.

OLD NAME SYSTEM_STATISTICS
NEW NAME SYSTEM_METRICS
DESCRIPTION SYSTEM_METRICS is used as the object in the command used to display the performance statistics for a CDCNET system.

OLD NAME FAILURE_STATISTICS
NEW NAME FAILURE_METRICS
DESCRIPTION FAILURE_METRICS is used as the object in the command used to display the failure statistics for a CDCNET system.

OLD NAME HARDWARE_STATUS
NEW NAME Same
DESCRIPTION HARDWARE_STATUS is used as the object in the command used to display the status of one or more hardware elements in a given CDCNET system.

OLD NAME SOFTWARE_STATUS
NEW NAME Same
DESCRIPTION SOFTWARE_STATUS is used as the object in the command used to display the status of all software components in a given CDCNET system.

OLD NAME DI_STATUS
NEW NAME SYSTEM_STATUS

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS
4.2.6 MISCELLANEOUS OBJECT NAMES

DESCRIPTION SYSTEM_STATUS is used as the object in a command used to display the status of a CDCNET system.

OLD NAME TERMINAL_ATTRIBUTES
NEW NAME Same
DESCRIPTION This is used as the object name in the commands used to define and display the terminal attributes.

OLD NAME CONNECTION AND CONNECTIONS
NEW NAME Same
DESCRIPTION CONNECTION is used as the object in commands used to create or delete a connection or an association for a terminal user. CONNECTIONS is used as the object in a command used to display all active connections for a given terminal user.

OLD NAME WORKING_CONNECTION
NEW NAME Same
DESCRIPTION WORKING_CONNECTION is used as the object in a command used to select one of the active connections for a terminal user as the working connection.

OLD NAME CONNECTION_ATTRIBUTES
NEW NAME Same
DESCRIPTION CONNECTION_ATTRIBUTES is used as the object in commands used to define and display the attributes of a terminal user connection or association.

OLD NAME STRING
NEW NAME Same
DESCRIPTION STRING is used as the object in commands used to obtain from or send to a terminal a character string.

OLD NAME RESOURCE
NEW NAME None
DESCRIPTION

OLD NAME MODULE
NEW NAME Same
DESCRIPTION MODULE is used as the object in commands used to load, deload and retain a software component in a CDCNET system.

4.3 CHANGES_IN_THE_COMMAND_NAMES

This section provides a list of current command names and proposed new names (if any) based on proposed changes in command verbs and objects. The commands are grouped in functional groups.

4.3.1 NETWORK DEFINITION COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
SET_SYSTEM_ATTRIBUTES	DEFINE_SYSTEM
SET_PERIPHERAL_ATTRIBUTES	None
None	DEFINE_CIM_INTERFACE
None	DEFINE_ESCI_INTERFACE
ADD_COMMUNITY	DEFINE_COMMUNITY_SEGMENT
None	DEFINE_COMMUNITY_MEMBERSHIP
ADD_FILE_TYPE	Same
SET_INITIALIZATION_DEFAULTS	SET_BOOT_DEFAULTS
SET_SYSTEM_EXCEPTIONS	SET_EXCEPTION_SYSTEMS
SET_HDLC_LINK_ATTRIBUTES	DEFINE_HDLC_TRUNK
SET_MCI_LINK_ATTRIBUTES	DEFINE_CHANNEL_TRUNK
SET_ETHERNET_LINK_ATTRIBUTES	DEFINE_ETHER_TRUNK
None	DEFINE_X25_TRUNK
SET_NETWORK_SOLUTION_ATTRIBUTES	None
None	DEFINE_HDLC_NET
None	DEFINE_ETHER_NET
None	DEFINE_CHANNEL_NET
SET_X25_NETWORK_SOLUTION_ATTRIBUTES	DEFINE_X25_NET
SET_NETWORK_PRODUCTS_INTERFACE	START_C170_INTERFACE
SET_C170_TRANSPARENT_GATEWAY	START_C170_BATCH_GW
SET_C170_INTERACTIVE_GATEWAY	START_C170_IVT_GW
SET_PACKET_LEVEL_ATTRIBUTES	START_X25_INTERFACE
SET_X25_TRANSPARENT_GATEWAY	START_X25_GW
CONFIGURE_TIP	DEFINE_TIP
CONFIGURE_LINE	DEFINE_LINE
CONFIGURE_TERMINAL_DEVICE	DEFINE_TERMINAL
SET_DIRECTORY_ATTRIBUTES	CHANGE_DIRECTORY
SET_LOG_RETAIN_QUEUE_LIMIT	None
SET_LOG_PRESERVE_QUEUE_LIMIT	None
SET_OSA_ATTRIBUTES	START_OSA
SET_DVM_ATTRIBUTES	None
SET_ERROR_REPORTING_THRESHOLD	CHANGE_LOGGING_THRESHOLD

4.3.2 NETWORK MONITORING COMMANDS

4.3.2.1 CONFIGURATION_DISPLAY_COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
DISPLAY_SYSTEM_ATTRIBUTES	DISPLAY_SYSTEM_OPTIONS
DISPLAY_HDLC_LINK_ATTRIBUTES	DISPLAY_HDLC_TRUNK_OPTIONS
DISPLAY_MCI_LINK_ATTRIBUTES	DISPLAY_CHANNEL_TRUNK_OPTIONS
DISPLAY_ETHERNET_LINK_ATTRIBUTES	DISPLAY_ETHER_TRUNK_OPTIONS
None	DISPLAY_X25_TRUNK_OPTIONS
DISPLAY_NETWORK_SOLUTION_ATTRIBUTE	None
None	DISPLAY_HDLC_NET_OPTIONS
None	DISPLAY_ETHER_NET_OPTIONS
None	DISPLAY_CHANNEL_NET_OPTIONS
None	DISPLAY_X25_NET_OPTIONS
DISPLAY_X25_PACKET_LEVEL_ATTRIBUTE	DISPLAY_X25_INTERFACE_OPTIONS
DISPLAY_DIRECTORY_ATTRIBUTES	DISPLAY_DIRECTORY_OPTIONS
DISPLAY_OSA_ATTRIBUTES	DISPLAY_OSA_OPTIONS
DISPLAY_DVM_ATTRIBUTES	None
DISPLAY_ERROR_REPORTING_THRESHOLD	DISPLAY_LOGGING_THRESHOLD

4.3.2.2 STATUS_AND_STATISTICS_DISPLAY_COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
DISPLAY_SOFTWARE_STATUS	Same
None	DISPLAY_TRANSPORT_STATUS
None	DISPLAY_DIRECTORY_STATUS
DISPLAY_FILE_TYPES	Same
DISPLAY_DI_STATUS	DISPLAY_SYSTEM_STATUS
DISPLAY_HARDWARE_STATUS	
DISPLAY_LINE_STATUS	
None	DISPLAY_TRUNK_STATUS
DISPLAY_NETWORK_STATUS	
DISPLAY_PATHS	
DISPLAY_SYSTEM_STATISTICS	DISPLAY_SYSTEM_METRICS
DISPLAY_FAILURE_STATISTICS	DISPLAY_FAILURE_METRICS
DISPLAY_STATISTICS	None
None	DISPLAY_LINE_METRICS
None	DISPLAY_TRUNK_METRICS
None	DISPLAY_NETWORK_METRICS

None	DISPLAY_TRANSPORT_METRICS
None	DISPLAY_DIRECTORY_METRICS

4.3.3 NETWORK CONTROL COMMANDS

4.3.3.1 CONFIGURATION_CHANGE_COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
CHANGE_HDLC_LINK_ATTRIBUTES	CHANGE_HDLC_TRUNK
CHANGE_MCI_LINK_ATTRIBUTES	CHANGE_CHANNEL_TRUNK
CHANGE_ETHERNET_LINK_ATTRIBUTES	CHANGE_ETHER_TRUNK
None	CHANGE_X25_TRUNK
CHANGE_NETWORK_SOLUTION_ATTRIBUTES	None
None	CHANGE_HDLC_NET
None	CHANGE_CHANNEL_NET
None	CHANGE_ETHER_NETWORK
	CHANGE_X25_NET
CHANGE_X25_PACKET_LEVEL_ATTRIBUTES	CHANGE_X25_INTERFACE
CHANGE_DIRECTORY_ATTRIBUTES	CHANGE_DIRECTORY
CHANGE_OSA_ATTRIBUTES	CHANGE_OSA
CHANGE_SYSTEM_ATTRIBUTES	CHANGE_SYSTEM

4.3.3.2 HARDWARE_AND_SOFTWARE_CONTROL_COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
ENABLE_LINE	START_LINE
DELETE_TERMINAL_DEVICE	DELETE_TERMINAL
DISABLE_LINE	STOP_LINE
DELETE_LINE	CANCEL_LINE
ACTIVATE_AUTO_RECOGNITION	START_AUTO_RECOGNITION
ENABLE_LINK	START_TRUNK
ENABLE_NETWORK_SOLUTION	START_NETWORK
DISABLE_LINK	STOP_TRUNK
DISABLE_NETWORK_SOLUTION	STOP_NETWORK
ACTIVATE_SOFTWARE_MODULE	None
DEACTIVATE_SOFTWARE_MODULE	None
DELETE_COMMUNITY	CANCEL_COMMUNITY_SEGMENT
None	CANCEL_COMMUNITY_MEMBERSHIP

4.3.3.3 STATISTICS_CONTROL_COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
ENABLE_STATISTICS_COLLECTION	START_METRICS_COLLECTION
DISABLE_STATISTICS_COLLECTION	STOP_METRICS_COLLECTION
ENABLE_STATISTICS_PERIODIC_REPORTING	START_METRICS_REPORTING
DISABLE_STATISTICS_PERIODIC_REPORTING	STOP_METRICS_REPORTING

4.3.3.4 RESOURCE_CONTROL_COMMANDS_

OLD_NAME	NEW_NAME(IF_DIFFERENT)
REQUEST_RESOURCE	None
RELEASE_RESOURCE	None
LOAD_MODULE	
RETAIN_MODULE	
UNLOAD_MODULE	

4.3.4 TERMINAL USER INTERFACE COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
DO	
SET_TERMINAL_ATTRIBUTES	DEFINE_TERMINAL_ATTRIBUTES
DISPLAY_TERMINAL_ATTRIBUTES	
None	DISPLAY_TERMINAL_ATTRIBUTE
CREATE_CONNECTION	
DISPLAY_CONNECTIONS	
SET_WORKING_CONNECTION	CHANGE_WORKING_CONNECTION
DELETE_CONNECTION	
SET_CONNECTION_ATTRIBUTES	CHANGE_CONNECTION_ATTRIBUTES
DISPLAY_CONNECTION_ATTRIBUTES	
None	DISPLAY_CONNECTION_ATTRIBUTE
GET_STRING	
PUT_LINE	PUT_STRING
HELP	

4.3.5 OPERATOR INTERFACE COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
SEND_COMMAND	
BROADCAST_COMMAND	
INHIBIT_ALARM_OUTPUT	CHANGE_COMMAND_ENVIRONMENT
DISPLAY_ALARM_HISTORY	
DISPLAY_CONNECTED_MDI	DISPLAY_COMMAND_ENVIRONMENT
SET_COMMAND_MDI	DEFINE_COMMAND_ENVIRONMENT
SET_SYNCHRONOUS_MODE	DEFINE_COMMAND_ENVIRONMENT
SET_ASYNCHRONOUS_MODE	DEFINE_COMMAND_ENVIRONMENT
EXECUTE_COMMAND_FILE	DO
ROUTE_COMMAND_RESPONSE	ROUTE_RESPONSE
ENABLE_COMMAND_ECHO	CHANGE_COMMAND_ENVIRONMENT
DISABLE_COMMAND_ECHO	CHANGE_COMMAND_ENVIRONMENT
ENTER_OPERATOR_NOTE	ADD_OPERATOR_NOTE
DISPLAY_OPERATOR_LOG	
QUIT	
ENABLE_COMMAND_LOGGING	CHANGE_COMMAND_ENVIRONMENT
DISABLE_COMMAND_LOGGING	CHANGE_COMMAND_ENVIRONMENT
SET_COMMAND_COMMUNITY	CHANGE_COMMAND_ENVIRONMENT
DELETE_COMMAND_COMMUNITY	CHANGE_COMMAND_ENVIRONMENT
SET_DEFAULT_DOMAIN	DEFINE_COMMAND_ENVIRONMENT

4.3.6 FAILURE MANAGEMENT COMMANDS

OLD_NAME	NEW_NAME(IF_DIFFERENT)
CHANGE_HARDWARE_ELEMENT_STATE	CHANGE_ELEMENT_STATE
CHANGE_LINK_TRAFFIC	CHANGE_TRAFFIC
CHANGE_LINK_TRAFFIC_RATE	CHANGE_TRAFFIC
CHECK_PATH	START_PATH_TEST
ECHO	START_PATH_TEST
None	STOP_PATH_TEST
LOOPBACK_LOCAL	START_LOCAL_LOOPBACK_TEST
LOOPBACK_REMOTE	START_REMOTE_LOOPBACK_TEST
None	STOP_LOCAL_LOOPBACK_TEST
None	STOP_REMOTE_LOOPBACK_TEST
EXECUTE_DIAGNOSTICS	None
None	START_MPB_TEST
None	START_CHANNEL_TEST
None	START_ESCI_TEST
None	START_CIM_TEST
None	START_SMM_TEST

4.0 CHANGES IN EXISTING COMMANDS DUE TO PROPOSED CONVENTIONS
4.3.6 FAILURE MANAGEMENT COMMANDS

None	START_LIM_TEST
None	START_PORT_TEST
None	STOP_MPB_TEST
None	STOP_CHANNEL_TEST
None	STOP_ESCI_TEST
None	STOP_CIM_TEST
None	STOP_SMM_TEST
None	STOP_LIM_TEST
None	START_PORT_TEST
TERMINATE_DIAGNOSTICS	None
ADD_DIAGNOSTIC_TRAFFIC	CHANGE_TRAFFIC

5.0 NOS/VE SCL METALANGUAGE RULES

5.0 NOS/VE_SCL_METALANGUAGE_RULES

The information in this section has been extracted from the NOS/VE Command Writer's Guide.

5.1 METALANGUAGE RULES

1. The symbol ::= is read as "is defined to be".

2. The symbol ¦ is read as "or" and elements separated by it are mutually exclusive.

3. Elements enclosed by < > constitute a single element in relation to surrounding metalanguage symbols.

4. Elements enclosed by [ ] are optional and constitute a single element in relation to surrounding metalanguage symbols.

5. Elements followed by ... may be repeated. When the ... follows a > or], the ... applies to the metalanguage text between and including the matching < or [.

6. Underlined characters are to be interpreted literally in a metalanguage definition (e.g. "≤" means the character "<", not the metasymbol "<"). An underline character by itself (e.g. _) means itself (as opposed to meaning the space character).

7. Names which appear in a metalanguage definition in upper case letters have, in the context in which they appear, special meaning to SCL. When using such names in actual SCL text, lower case letters may be used interchangably with the corresponding upper case letters.

8. The notation <the ascii xxxxx> is used to denote a particular (non-graphic) character in the ASCII (American Standard Code for Information Interchange) character set (xxxxx describes the character in question).

9. The notation <any ascii character except xxxxx> is used to denote any ASCII character except the character(s) specified by xxxxx.

10. The notation <xxxxx expression> is used to denote a expression of type xxxxx.

5.0 NOS/VE SCL METALANGUAGE RULES

11. The notation <xxxxx function> is used to denote a function that returns a value of type xxxxx.

12. The notation <xxxxx command> is used to denote the command named xxxxx.

13. The notation <xxxxx name> is used to denote a name that designates a xxxxx.

14. The notation <xxxxx variable> is used to denote a variable of type xxxxx.

15. The notation <ascii> is used to designate any ASCII character.

16. The notation <beginning of line> is used to designate the beginning of a command line.

17. The notation <end of line> is used to designate the end of a command line.

18. The notation <eoi> is used to designate the end of information on a file.

19. The notation <eop> is used to designate the end of a partition on a file.

20. The notation <empty> is used to designate an empty (null) syntactic construct.

5.1 USE OF SPACES

Spaces may in general be used to separate other SCL elements from each other in order to improve readability. Also, SCL permits spaces to be used as separators between parameters and between the elements of a list. Wherever spaces may appear, comments may also be used.

```
<sp> ::= <spaces ¦ comment>...

<spaces ¦ comment> ::= <spaces> ¦ <comment>

<spaces> ::= <space>...

<space> ::= <the ascii space character SP>
          ¦ <the ascii horizontal tab character HT>
```

Because of their use as separators, it is necessary to impose restrictions on where spaces can occur. For instance, spaces are not permitted between the component parts of a file or variable reference.

5.2 DELIMITERS

The following definitions are for delimiters used throughout SCL for various purposes. The definitions illustrate whether spaces (see previous section) preceding or following the actual delimiter are considered to be a part of the delimiter.

Delimiters other than those below are used in SCL for various purposes but only for those shown below is the use of surrounding spaces consistent throughout.

```
<,¦sp¦nl> ::= <,> ¦ <sp> ¦ <nl>

<sp¦nl> ::= <sp> ¦ <nl>

<;¦nl> ::= <;> ¦ <nl>

<nl> ::= <eol> <bol> [<eol> <bol>]...

<bol> ::= <beginning of line> [<sp>]
```

5.0 NOS/VE SCL METALANGUAGE RULES
5.2 DELIMITERS

```
<eol> ::= [<sp>] <end of line>

<,¦sp> ::= <,> ¦ <sp>

<,> ::= [<sp>] , [<sp>]

<;> ::= [<sp>] ; [<sp>]

<:> ::= :

<(> ::= ( [<sp>]

<)> ::= [<sp>] )

<.> ::= .

<..> ::= [<sp>] <.><.> [<.>]...  [<sp>]
```