UNISYS

ALLY® Software Development Environment

C-ISAM Developer Notes

June 1988

Printed in U S America
Priced Item
UP–12970 Rev. 1

Preface

This manual describes ALLY release 2.0.

The ALLY Software Development Environment can run on many computers, operating systems, and data access methods. Therefore, the ALLY manuals are generic—they describe the system-independent features of ALLY.

These developer notes tell you how to build applications that use the C-ISAM indexed sequential access method. These notes are a supplement to the standard set of manuals provided with ALLY.

These notes include information about:

- using ALLY with C-ISAM
- building a C-ISAM Base Data Source Definition (Base DSD)
- creating and using C-ISAM indexes
- modifying a C-ISAM Base DSD
- concurrency issues

At the end of these notes is a description of a sample C-ISAM application.

The section titled "Building a C-ISAM Base DSD" is designed as a supplement to the "Data Source Definitions" chapter of the *Dialog User's Guide* (UP-12505).

We assume that you are familiar with C-ISAM as well as with ALLY and the Dialog. Before reading the developer notes, you should read the documentation conventions that are provided in the preface of the *Dialog User's Guide*.

End of Preface

Contents

C-ISAM Developer Notes

Appendix A. Dialog Structure for C-ISAM DSDs

Figures

Tables

C-ISAM Developer Notes

C-ISAM, a product of Relational Database Systems, Inc., is an indexed sequential access method. A C-ISAM file consists of two physical files. One holds the data, and the other holds the indexes. By definition, the data file name has the extension ".dat," and the index file name has the extension ".idx." Although there are two files, they are always used together as a single unit. Figure 1 shows a sample data file and index file.

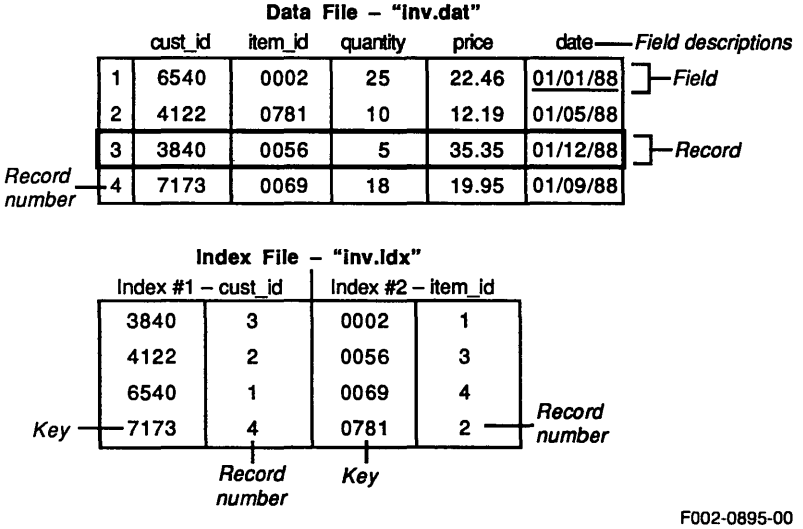


Figure 1. C-ISAM Data File and Index File

A C-ISAM data file is a collection of fixed-length records. Each record contains at least one field.

A C-ISAM index is composed of record numbers and values extracted from one or more fields of the corresponding records. A value extracted from a record for inclusion in an index is called a key.

C-ISAM allows you to define an unlimited number of indexes for each data file. All indexes related to a data file are saved in the same index file. As a data file is updated, C-ISAM automatically updates all of the indexes. Creating or removing indexes does not affect the records in the data file.

Using ALLY with C-ISAM

You can build ALLY applications that use existing C-ISAM files. You can also use ALLY to create new C-ISAM files and to add indexes to existing files.

This section compares the meaning of the terms *key* and *primary key* in C-ISAM documents with the meaning of these terms in ALLY documents. It also describes the following characteristics of C-ISAM files used in ALLY applications:

- file names
- data types
- record length
- record layout

Comparison of Terms

The terms *key* and *primary key* are used in descriptions of C-ISAM and in descriptions of ALLY. However, the C-ISAM terms and the ALLY terms do not have the same meaning.

In C-ISAM descriptions, the term *key* refers to a component of an index. A C-ISAM key is used to define the order in which the records in a file are to be processed. A *primary key* is a key in the first index that is created as the C-ISAM file is created. Within ALLY, the distinction between the first index to be created and indexes that are added later is not significant.

In ALLY descriptions, the term *key* is used to describe an item that references one or more Base DSD fields. In ALLY applications that use C-ISAM files, Base DSD keys enable you to create C-ISAM indexes and to sort records. In ALLY, a *primary key* uniquely identifies a record. When you create a C-ISAM index from a primary key, you prevent the application user from entering duplicate values in the field(s) that the key references.

When a key in a Base DSD is assigned the characteristic "defines a C-ISAM index," the key is referred to in this document as an *indexed key*.

File Names

A C-ISAM file name can be up to ten characters long. However, ALLY does not validate the name length, because this restriction is imposed by UNIX.

If the data file is located in a directory different from the application's directory, the name of the data file can include a path name.

C-ISAM adds the extensions ".dat" and ".idx" to the name you specify when you create a C-ISAM Base DSD. When you enter a C-ISAM file name in a Dialog form, do not type the extension as part of the name.

The characters in the path name and the three-letter extension do not apply to the file name's character count.

Data Types

ALLY supports all data types supported by C-ISAM. The ALLY data types CHAR, DATE, and NUMBER are mapped to C-ISAM data types as described below.

Character Values

Character values are stored in fixed-length segments within a record. However, the data stored in character fields is typically variable in length. By default, ALLY uses space characters to fill any unused positions in a character field when data is written to the file. These "pad characters" are removed when the data is read.

When you create a character field in a C-ISAM Base DSD, use the value CHAR for both the ALLY data type and the storage data type.

Date Values

ALLY supports two formats for the external storage of dates in C-ISAM files: DATE and CHAR DATE. Both of these external data storage types are compatible with the ALLY (internal) data type DATE.

If you use DATE for the storage (external) data type, the date will be stored in ALLY's internal, binary format.

If you create a Base DSD to describe a C-ISAM file in which date values are stored as characters, use the CHAR DATE format for the character/date fields. You cannot specify this data type when you create a Base DSD, but you can change a Base DSD field's external data type to CHAR DATE after the field has been created. See the section titled "Creating a Character/Date Field" for additional information.

Dates stored with the CHAR DATE format may not be sorted in the same order as dates stored with the DATE format. For example, if the format of the date in the C-ISAM file is yyyy/mm/dd, sorting will be chronological. If the format is mm/dd/yyyy, sorting will not be chronological.

Number Values

ALLY supports all the numeric formats defined by C-ISAM (integer, long integer, float, double, and decimal). The Dialog calls integer "SINT16" and calls long integer "SINT32."

All numeric data can be stored in the data file in ALLY's internal format called NUMBER. Although other formats may be more compact, using the NUMBER format for the storage type is usually most efficient in terms of processing time because ALLY does not have to convert the number to a different format when reading or writing data.

When you select the storage type NUMBER or DECIMAL, the Dialog computes a default field length (in bytes) based on the ALLY data type of the field (NUMBER). You can change the field length if necessary, but we do not recommend it. If you increase the length above the default, no data will be lost; however, the number of significant digits will not be increased. If you reduce the length below the default, the number of significant digits stored will be lower than the number specified by ALLY, and the Dialog will report an integrity error.

Record Length

A C-ISAM file contains fixed-length records with a maximum length of 65,535 characters per record.

When you create a C-ISAM Base DSD, ALLY calculates field storage length, offsets, and record length based on the storage type and width you specify. Table 1 shows how ALLY calculates the field storage length (in bytes) of each type of field.

Table 1. Calculation of Field Storage Length

If the Storage Type Is	The Field Storage Length Equals
CHAR	the number of characters specified in the "Width" field
DATE	14
SINT16	2
SINT32	4
FLOAT	4
DOUBLE	8
DECIMAL	(width - precision + 1)/2 + (precision + 1)/2 + 1
NUMBER	(width - precision + 1)/2 + (precision + 1)/2 + 3

When ALLY calculates the width of DECIMAL and NUMBER fields, nonintegers are rounded down.

Record Layout

Fields cannot overlap in a data file, and they cannot extend beyond the end of a record.

A field's offset indicates the number of bytes into a data record that the field begins. ALLY automatically calculates the offset for a field of a C-ISAM Base DSD based on the offset and length of the preceding field in the record.

If the "row identifier" file open option is on, the first usable field offset is 4. If this option is off, the first usable field offset is 0. (Refer to the section titled "File Open Options" for additional information on this option.)

Building a C-ISAM Base DSD

When you build an ALLY application that creates C-ISAM files or uses data from existing C-ISAM files, you must build C-ISAM Base Data Source Definitions (Base DSDs) to describe the application's data.

Before you build a Base DSD to describe a C-ISAM file, you should know the following information about each field in the file:

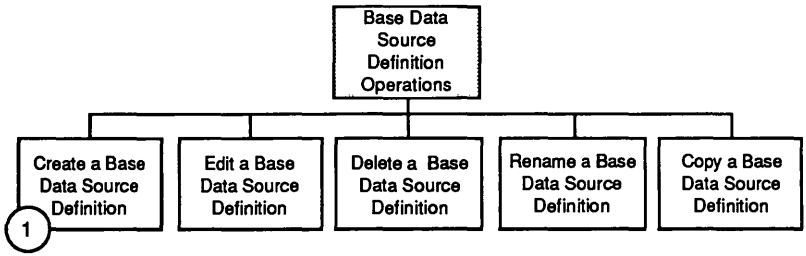
- name
- ALLY data type
- storage data type
- width
- precision of number fields

The following section describes the process of building a C-ISAM Base DSD.

Creating the Base DSD

When you work on an application's Base DSDs, you are in the "Data Definitions" branch of the Dialog—choice 3 from the Dialog's main menu.

Figure 2 shows the location of the Dialog form you will use.



F002-0648-01

Figure 2. C-ISAM Base Data Source Definition Path

① Name the C-ISAM Base Data Source Definition and define the fields.

Menu path: **3 1 1** from the Dialog's main menu
Form name: *Create a Base Data Source Definition*

Figure 3 shows this form.

```
        Create a Base Data Source Definition

DSD name:

DSD type:

Create from:      CREATE BY HAND

Table or file name:

Display optional information for this DSD type?  (Y/N) N

Create fields for this DSD?  (Y/N) Y
```

Figure 3. Create a Base Data Source Definition Form

Name the C-ISAM Base DSD and choose "CI" as the type of Base DSD you are defining. The cursor skips the "Create from" field and moves to the "Table or file name" field.

In the "Table or file name" field, the Dialog places the name you entered in the "DSD name" field. You can change this name; however, do not add the ".dat" extension to the name. C-ISAM adds the file name extensions ".dat" and ".idx" to the appropriate file names automatically. Remember that a C-ISAM file name cannot exceed ten characters.

The "Display optional information for this DSD type?" field accesses the *C-ISAM DSD Characteristics* subform. We discuss the subform later in the "Optional information" section.

Type <Return> in the "Create fields for this DSD?" field to access the *List of DSD Fields* subform (Figure 4).

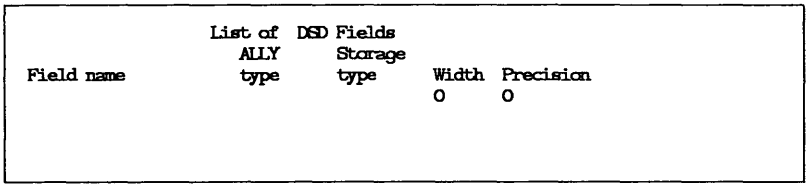


Figure 4. List of DSD Fields Subform

In this subform, name the field and specify the ALLY (internal) data type and the storage (external) data type.

If you define a character data type, you must specify the width of the field.

Date data types require 14 bytes of storage space in the data source record. If you define a date data type, the Dialog automatically places the number 14 in the "Width" field.

If you plan to create a character/date field (a date field in which the values are stored as characters), define the data type as CHAR when you create the Base DSD. See the section titled "Creating a Character/Date Field" for additional information.

If you define a number data type, select the appropriate external storage type from the list of values. Then specify the width of the number. You can also specify the precision of the number.

Optional information

To edit the optional information, you must move to the "Display optional information for this DSD type?" field and type "Y<Return>." The *C-ISAM DSD Characteristics* subform appears (Figure 5). If the cursor is in the *List of DSD Fields* subform, you must use the 'up' command, usually assigned to the up arrow key (<↑>), to move to this subform.

Figure 5. C-ISAM DSD Characteristics Subform

On this subform, you can specify the record's length. Remember that the Dialog will calculate the record's length according to the fields that you define on this form. If the data file does not already exist, you do not have to specify the record's length. However, the Dialog does not leave extra space in a record. If you want to pad a record, you must specify the record's length. If the data file already exists, you can enter the record length.

You can also use this subform to enter a comment about the Base DSD.

Optional Steps for Base DSD Fields

The *Base DSD Field Characteristics* menu (menu path 3 1 2 < > 2 2 < > from the Dialog's main menu) provides access to six Dialog forms that allow you to edit a Base DSD field's:

- offset in the data source record
- initial and null values and data formats
- minimum and maximum values
- options inheritable by a form/report field
- ALLY (internal) data type
- storage (external) data type

Define Field Offset

Menu path: **3 1 2 < > 2 2 < > 1** from the Dialog's main menu
Form name: *C-ISAM DSD Field—Characteristics*

This form allows you to edit a field's offset in the record.

Be aware that if you change a field's offset, you may have to change:

- the record length
- the offsets of other fields in the record

Define Field Initial and Null Values and Data Formats

Menu path: **3 1 2 < > 2 2 < > 2** from the Dialog's main menu
Form name: *Field—Initial and Null Values, Data Formats*

This form allows you to specify the initial and null values of a Base DSD field and the data format of the field (if applicable).

Chapter 4 of the *Dialog User's Guide* discusses ALLY formats. By default, ALLY provides the following initial and null values and input and output formats for character, number, and date fields.

Table 2. Default DSD Field Values and Formats

Data Type	Initial Value	Null Value	Input Format	Output Format
Character	none	none	none	none
Number	0	0	free format*	free format*
Date	none	none	MM/DD/YY*	MM/DD/YY*

* These formats are the Dialog's global defaults and are used when no format is specified for a DSD field.

Define Field Minimum and Maximum Values

Menu path: **3 1 2 < > 2 2 < > 3** from the Dialog's main menu
Form name: *Field—Field Validation*

This form allows you to specify a minimum and maximum value for the field you are defining. By default, ALLY does not assign a minimum or maximum value to DSD fields.

Define Inheritable Form/Report Options

Menu path: **3 1 2 < > 2 2 < > 4** from the Dialog's main menu
Form name: *Options Inheritable by a Form/Report Field*

This form allows you to specify the options of a Base DSD field that will be inherited by all form/report fields that reference the Base DSD field. These options allow the field to require a valid value, be addable only, be enterable only, or accept no blank characters.

Define a Field's Internal Data Type

Menu path: **3 1 2 < > 2 2 < > 5** from the Dialog's main menu
Form name: *Base DSD Field—Internal Data Information*

This form allows you to edit a Base DSD field's ALLY (internal) data type and specifications.

Use this form to indicate whether or not the value of a number field is a floating point number.

Be aware that if you change a field's data type after you add data to the database, your data will be invalid when you run the application.

If you change a field's internal data type, you must change its external data type to correspond.

If you change the width of a field, you may need to change:

- the record's length
- the offsets of other fields in the record

Define a Field's External Data Type

Menu path: **3 1 2 < > 2 2 < > 6** from the Dialog's main menu
Form name: *C-ISAM External Storage*

This form allows you to edit a Base DSD field's external (storage) data type and specifications.

Be aware that if you change a field's data type after you add data to the database, your data will be invalid when you run the application.

See the section titled "Creating a Character/Date Field" for additional information about this form.

File Open Options

Menu path: **3 1 2 < > 1** from the Dialog's main menu
Form name: *C-ISAM Base DSD—Characteristics*

ALLY offers several options for the opening of C-ISAM data files. The default settings of these options:

- allow users to read from and write to files
- cause a C-ISAM data file and index file to be created at runtime if they do not exist
- cause ALLY to add a row identifier to each record of the data file

Figure 6 shows the subform that allows you to change these options.

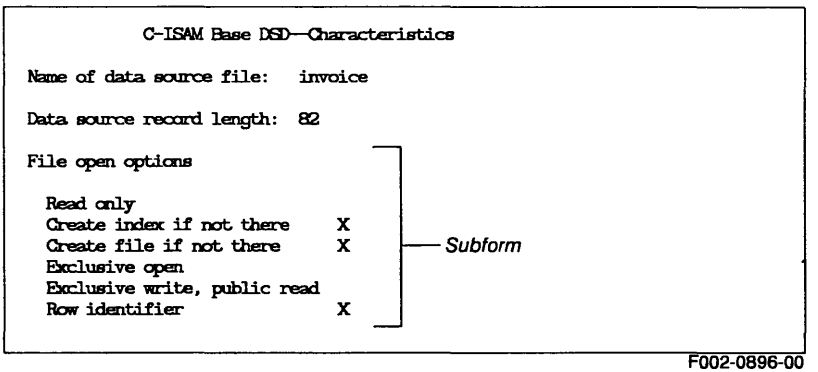


Figure 6. File Open Options Subform

Read only (default off)
This option determines whether changes can be made to the data file. When this option is on, any attempt to insert, update, or delete records generates an error message.

Create index if not there (default on)
This option determines whether ALLY will verify at run-time that the indexes defined by the Base DSD are present. When this option is on, ALLY will build any indexes that are missing. ALLY is unable to build indexes if other users are accessing the file at the same time. After all index changes are completed in an application, you may want to turn off this option to avoid the overhead of verifying indexes.

Create file if not there (default on)
This option determines whether ALLY will create the C-ISAM data file if it does not already exist. If it creates the file, ALLY opens it for exclusive access until the DSD referencing it is closed.

Exclusive open (default off)
This option determines whether ALLY will open the C-ISAM data file for exclusive access. When this option is on, no other user can open the data file while the Base DSD referencing it is active. Individual records are not locked during an update transaction. This makes updates more efficient and avoids problems associated with the

upper limit on the number of records that can be locked at one time. (See the section titled "Concurrency" for additional information on record locking.)

Exclusive write, public read (default off)
This option determines whether ALLY will lock the file whenever it is opened so that no other user can modify the file until it is closed. This option is provided for situations in which only one user at a time should have write access to a file. It is similar to the "exclusive open" option, but it allows other users to have read access to the file when it is being modified. (See the section titled "Concurrency" for additional information on record locking.)

Row identifier (default on)
This option determines whether ALLY will put a unique identifier into the first four bytes of each record of the application's data source. The purpose of this identifier is to allow ALLY to manage correctly certain concurrent update situations. (See the section titled "Concurrency.")

This option should normally be on. The ability to turn it off is provided primarily to allow ALLY applications to access C-ISAM data files created outside of ALLY. It should be off only in an application in which one of the following situations is true:

- Users make no updates or deletions.
- Only one user makes updates or deletions at any given time.
- Records cannot be deleted, but only inserted and/or updated.

Options Inheritable by Forms/Reports

Menu path: **3 1 2 < > 1** from the Dialog's main menu
Form name: *C-ISAM Base DSD—Characteristics*

ALLY provides several options that are inheritable by a form/report that references a C-ISAM Base DSD. Figure 7 shows the subform that allows you to select these options.

```
Options inheritable by a form/report

Delete dependent records          Update not allowed
Ignore null records               Insert not allowed
Record commits not automatic      Delete not allowed
ALLY does not log transactions  X
```

Figure 7. Options Inheritable by a Form/Report Subform

Delete dependent records (default off)
This option refers to subordinate records in a form/report. It determines whether ALLY will delete dependent records from a Base DSD when an application user deletes the records' parent record.

Ignore null records (default off)
This option determines whether ALLY will ignore a record when all of the fields of the record contain null values.

Record commits not automatic (default off)
This option determines whether record changes will be made to a data file automatically or only when the user invokes the 'commit' command. If this option is off, a commit is done automatically when the cursor moves from a changed record. If this option is on, no changes will be made to the data file until a user invokes the 'commit' command. When this option is off, the "ALLY does not log transactions" option should be on. This is the default condition.

ALLY does not log transactions (default on)
This option determines whether ALLY will maintain a file to log transactions for a form/report that references a

C-ISAM Base DSD. ALLY uses this log to return rolled-back records to their previous internal values in a form/report. ALLY maintains this transaction file when this option is off; it does not maintain the file when the option is on. When this option is on, the "record commits not automatic" option should be off.

Update not allowed (default off)
This option determines whether ALLY will prevent records from being updated in a C-ISAM file.

Insert not allowed (default off)
This option determines whether ALLY will prevent records from being inserted into a C-ISAM file.

Delete not allowed (default off)
This option determines whether ALLY will prevent records from being deleted from a C-ISAM file.

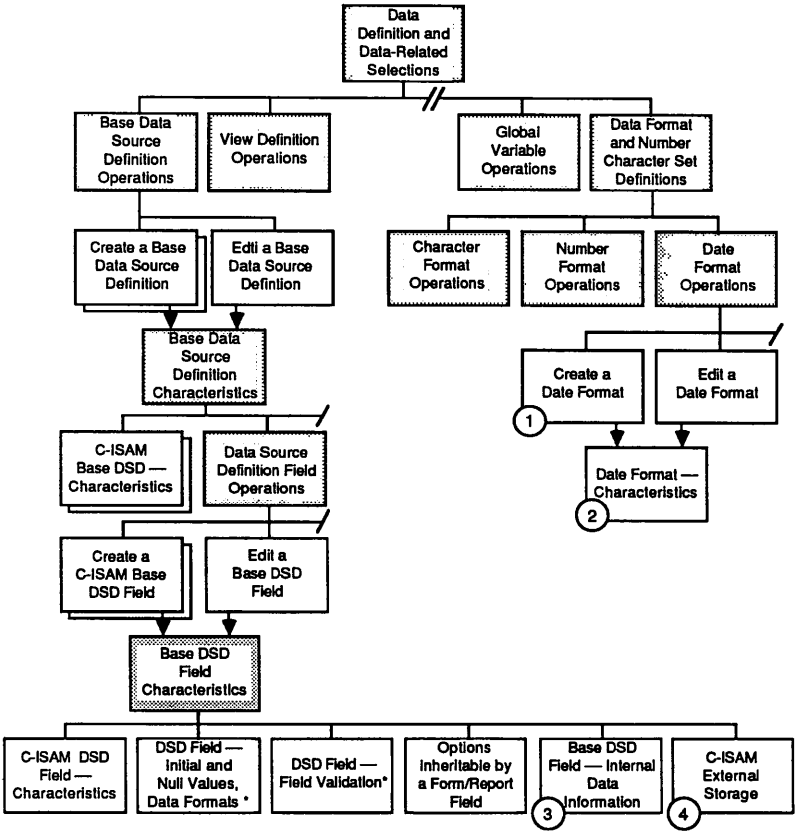
Creating a Character/Date Field

You can create a C-ISAM Base DSD to describe a C-ISAM file in which date values are stored as characters. In the C-ISAM Base DSD, you must create a field with the external storage type CHAR, and then modify the field.

First, follow the instructions in the section titled "Building a C-ISAM Base DSD" to create the Base DSD and all of its fields. In the *List of DSD Fields* subform, name the character/date field. Assign it the ALLY data type CHAR, and specify a width that will accommodate the output of the date format.

The steps required to modify the Base DSD field to change its data type follow. When you modify this Base DSD field, you will be in the "Data Definitions" branch of the Dialog—choice 3 from the Dialog's main menu.

Figure 8 shows the location of the Dialog forms you will use.



F002-0809-00

Figure 8. C-ISAM Base DSD Path: Character/Date Field

① Name the data format item.

Menu path: **3 5 3 1** from the Dialog's main menu
Form name: *Create a Date Format*

Name the data format item and exit from the form to move to the *Date Format—Characteristics* form.

② Define the date format.

Form name: *Date Format—Characteristics*

Type a combination of date picture symbols, literal characters, and literal strings to create the date format you need. For example, if the format of the dates in your C-ISAM file is yyyymmdd, type yyyymmdd on this form. Once you create a date format, you can use it for more than one field. Chapter 4 of the *Dialog User's Guide* describes the process of creating date formats.

③ Change the character/date field's internal data type.

Menu path: **up up 1 2 < > 2 2 < > 5**
Form name: *Base DSD Field—Internal Data Information*

Change the internal data type from CHAR to DATE, and exit from the form to display the *Base DSD Field Characteristics* menu.

④ Change the character/date field's external data type.

Menu path: **6**
Form name: *C-ISAM External Storage*

Change the external C-ISAM data type from CHAR to CHAR DATE.

Type <Return> in the "Length of storage container" field to display a subform containing the "Date format" field. Then select from the list of values the date format you created in steps 1 and 2.

If the width you specified when you created the Base DSD is too short for the date format, the Dialog will report an integrity error when you exit from this form.

Modifying a Base DSD for an Existing C-ISAM File

The Dialog provides no restrictions on making changes to a Base DSD that describes a C-ISAM file. However, once ALLY creates a C-ISAM file from a Base DSD, you cannot change the record length specified in the Base DSD without deleting the file and re-creating it.

Once you have added data to a C-ISAM file, you should not define additional fields even if there is room for them within the record length. If you define additional fields, their contents will be unpredictable in the existing records.

If you modify the data type of a field after data has been added to the data file, the change will make the existing records unreadable.

Indexes

ALLY allows you to define an index by creating a key within a Base DSD. You must use the form called *Base Definition Key* (menu path 3 1 2 < > 3 2 < > 1 from the Dialog's main menu) to assign the characteristic "defines a C-ISAM index" to the key. When this characteristic is set, ALLY will create a C-ISAM index for the key at runtime if the index does not already exist. When a key has this characteristic, we refer to it as an *indexed key*.

An indexed key can be used to:

- read records in sorted order
- ensure that index entries are unique
- optimize the record-selection searches that implement foreign key links or that satisfy user queries

An indexed key can contain a maximum of eight fields. The sum of the lengths of the fields in an indexed key cannot exceed 120 bytes.

Base DSD Keys

Menu path: **3 1 2 < > 3 2 1** from the Dialog's main menu
Form name: *Base Definition Key*

When you define a key in a C-ISAM Base DSD, ALLY allows you to assign various characteristics to the key. Figure 9 shows the subform that allows you to assign these characteristics to a key.

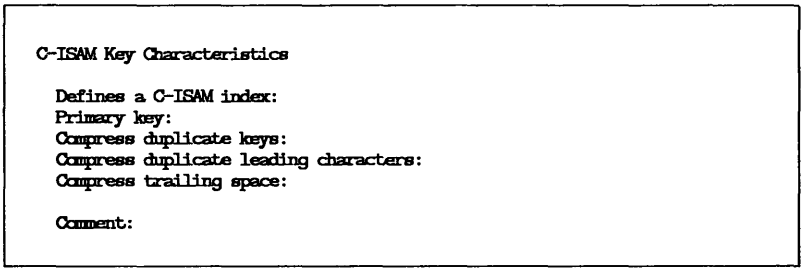


Figure 9. C-ISAM Key Characteristics Subform

Defines a C-ISAM index (default off)
This option determines whether ALLY will create a C-ISAM index for this key, if one does not already exist.

If you want to select any of the remaining options, you must select this one.

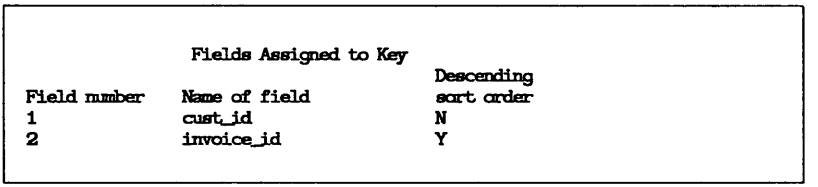
Primary key (default off)
This option determines whether ALLY will give the C-ISAM index the "duplicates not allowed" characteristic. When this option is on, ALLY displays an error message if a user attempts to create a record with a duplicate value in the primary key's field(s).

Compress duplicate keys (default off)
This option determines whether C-ISAM will compress index values that are identical. Selecting this option (and the other compression options shown on this subform) may reduce the size of your index file and may increase the efficiency of record-selection searches.

Compress duplicate leading characters (default off)
This option determines whether C-ISAM will compress index values with the same initial sequence of bytes in the key field(s).

Compress trailing space (default off)
This option determines whether C-ISAM will compress the values of indexed fields when the fields of the data file are padded with trailing space characters. To make use of this feature, ALLY uses a space character rather than a null character as the default pad character for CHAR fields.

When you define a key, you must assign one or more Base DSD fields to the key. Figure 10 shows the subform that allows you to assign fields to a key.



Fields Assigned to Key

Field number	Name of field	Descending sort order
1	cust_id	N
2	invoice_id	Y

Figure 10. Fields Assigned to Key Subform

The section titled "Reading Records in Sorted Order" explains how to use the "Descending sort order" field of this subform. The section titled "Optimizing Record-Selection Searches" provides information to help you decide which fields to assign to a key.

Reading Records in Sorted Order

ALLY can read a C-ISAM data file in sorted order if the records contain at least one field that is referenced by an indexed key. If an application is to read C-ISAM records in sorted order, it must include:

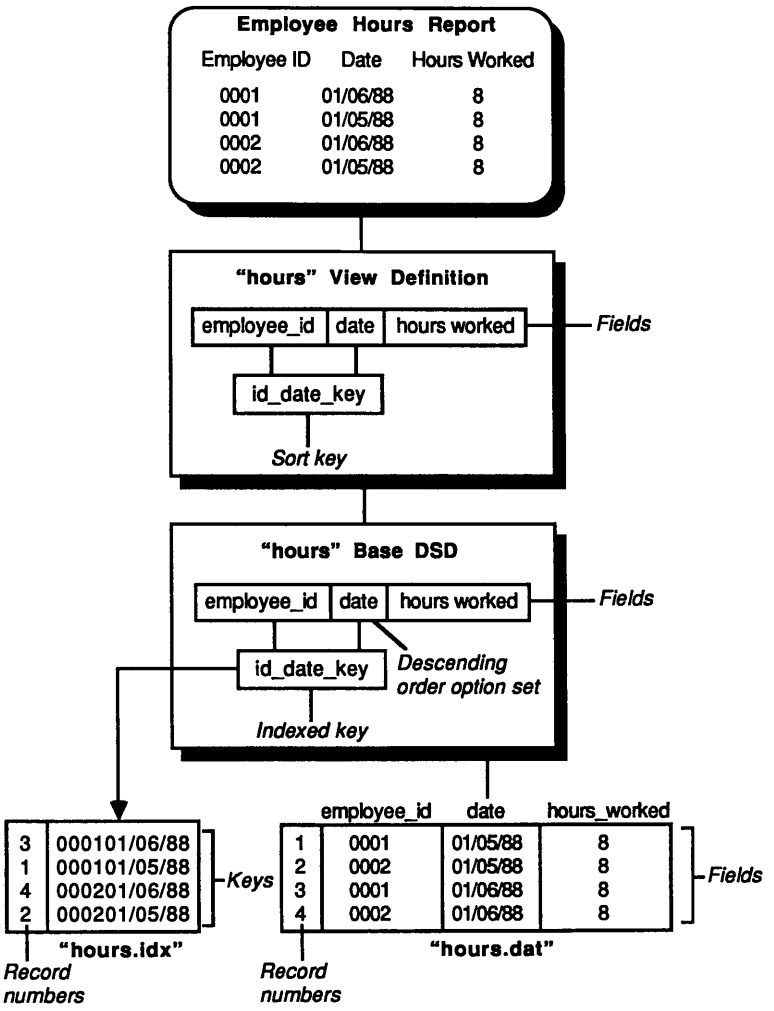
1) a Base DSD that contains an indexed key
2) a View Definition that:
   - references the Base DSD
   - references all the fields used in the indexed key
   - uses the indexed key as its sort key

The View Definition allows ALLY to read records in order, sorted by the field(s) of the sort key.

C-ISAM can sort records in ascending order or in descending order. It sorts in ascending order unless you set the "descending sort order" option. Use the *Base Definition Key* form (menu path 3 1 2 < > 3 2 < > from the Dialog's main menu) to set this option for a field assigned to the sort key.

Several fields can be assigned to the same key, and you can set the "descending sort order" option for each of those fields. Therefore, ALLY can sort simultaneously in ascending order of one field of the key and in descending order of another field. The priority of the sort order within a key is determined by the order in which the fields are assigned to the key. The first field assigned to the key has the highest priority.

Figure 11 shows an example of this capability. In the example, the records are sorted first in ascending order by employee ID and then in descending order by date.



F002-0899-00

Figure 11. Ascending and Descending Sort Order

Optimizing Record-Selection Searches

At runtime, ALLY looks for an index to use whenever it must:

- sort records
- implement a foreign key link
- locate selected records to satisfy a query

Although a C-ISAM application can have many indexes, ALLY can use only one index at a time. To determine which index it will use, ALLY follows a protocol designed to make record-selection searches as efficient as possible. Understanding this protocol will help you design more efficient applications.

ALLY's Protocol for Selecting an Index

If a form/report group or ADL procedure references a View Definition that has a sort key, ALLY always uses the index associated with the sort key. It does not attempt to find any other index.

If there is no sort key, but there is a foreign key link associated with the View, ALLY selects an index that implements the foreign key link. If there is no sort key and no foreign key link, but a query is performed, ALLY selects an index that optimizes the record-selection search required to satisfy the query.

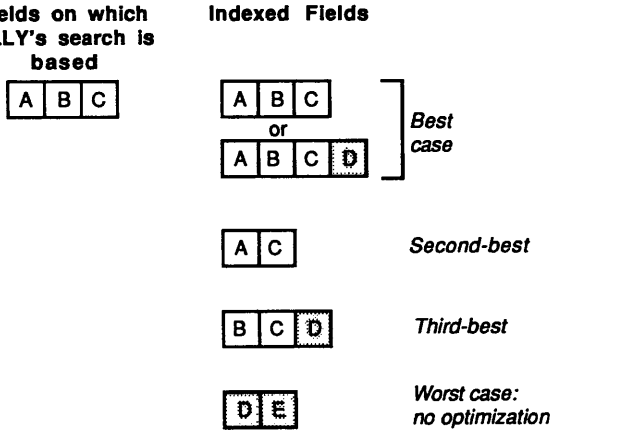
When ALLY is implementing a foreign key link or locating selected records to satisfy a query, it must find only those records that have certain values in certain fields. In other words, it must find the records that meet certain selection criteria. In these cases, ALLY begins by selecting an index to optimize its record-selection search.

ALLY can locate the best available index because it can recognize varying degrees of match between the available indexes and the fields on which the record-selection search is based. The best case exists when the fields in the foreign key link or the fields used in the query match exactly the fields in an index. For example, if ALLY's search is based on fields A, B, and C, it finds an index

that uses fields A, B, and C, and no others. In this case, ALLY reads only the records that meet the selection criteria, and it does not have to examine any other records.

Efficiency is reduced only slightly if the fields on which the search is based form a prefix of the fields used in an index. For example, if ALLY's search is based on fields A, B, and C, it finds an index that uses fields A, B, C, and D.

If ALLY does not find a "best case" among the available indexes, it continues to search for an index to use. Figure 12 illustrates the levels of optimization provided by varying degrees of match.



F002-0907-00

Figure 12. Degrees of Match in C-ISAM Indexes

Second-Best Case — The fields used in the index are a subset of the fields on which the search is based. For example, if ALLY's search is based on fields A, B, and C, it finds the index that contains fields A and C only. ALLY uses the index for finding records that contain fields that match the indexed fields. However, it must then examine these records to determine whether the fields that are not indexed meet the selection criteria.

Third-Best Case	The first field used in the index is one of the fields on which the search is based. For example, if ALLY's search is based on fields A, B, and C, it finds the index in which the first field used is field B. ALLY uses the index to find the first record, in indexed order, that meets the selection criteria. It then reads the rest of the records for matching fields.
Worst Case	The fields on which the search is based do not match any of the fields used in indexes. For example, if ALLY's search is based on fields A, B, and C, it cannot locate an index because the only fields used in indexes are D and E. In this case, the record-selection search is not optimized. ALLY reads and examines all records in the file to determine whether they meet the selection criteria.

To take advantage of ALLY's protocol for selecting an index, you can use the following strategies:

- Set the "defines a C-ISAM index" option for the local key of the foreign key link.
- Create a sort key in which the first field is the field used in the local key of the foreign key link and additional fields are added for the purpose of sorting records at runtime.
- For a field that is often used in queries, define an ALLY key and set the "defines a C-ISAM index" option.

Removing or Modifying a C-ISAM Index

Once ALLY creates a C-ISAM index, do not update the Base DSD key's options or its field assignment. Changing the Base DSD key could cause a runtime error or cause a new index to be created. If changes are necessary, delete the key and re-create it.

If you need to remove or modify a C-ISAM index, follow these steps:

1) As a safeguard, make a backup copy of the C-ISAM data file and index file.
2) Migrate the data to a transportable file.
3) Modify the Base DSD key definition as needed.
4) Delete the old data file and index file.
5) Migrate the transportable file back to the C-ISAM file.

For information on migrating files, refer to the *Utilities User's Guide* (UP-12508).

ALLY Development Language (ADL) and C-ISAM

You can use ADL generic Data Manipulation Language (DML) functions to manipulate the data in a C-ISAM file through ADL procedures. These functions are described in the *ALLY Development Language (ADL) User's Guide* (UP-12507).

Concurrency

When a user tries to modify or delete a record, ALLY locks the record and verifies that the displayed values for the record's fields are still correct. If the record has been modified or deleted by another user, ALLY displays an error message. ALLY then updates the user's display to reflect the record's status in the data file and releases the lock when a successful commit or rollback transaction is completed.

When user B attempts to lock a record that has been locked by user A, ALLY waits up to ninety seconds for the lock to be released. If the record is still locked at the end of this time, user B's attempt to modify the record generates an error message.

The number of records that can be locked at one time is determined by the operating system. If this number is exceeded, the error message "No record locks available" will be displayed. This type of error is most likely to occur when a form/report stores a computed field in a data file or when an ADL routine updates many records without calling DB_COMMIT. Setting either the "exclusive open" file open option or the "exclusive write, public read" file open option will resolve this problem because these options cause the entire file to be locked when it is opened. Resetting the operating system parameter that defines the number of locks the system can handle will also resolve this problem.

The file open option "row identifier" allows you to manage certain concurrent update situations. Here is an example of a problem that the "row identifier" option prevents.

Assume that user A and user B access a C-ISAM data file at the same time.

1) User A reads record 17.
2) User B deletes record 17 and commits this change. User B then inserts (and commits) a new record with different values, which C-ISAM happens to assign as record number 17.
3) User A attempts to modify data in the record 17 that was displayed in step 1.

ALLY's response to user A will differ, depending on the state of the "row identifier" option.

Row identifier option on	An error message tells user A that this record has been deleted.
Row identifier option off	An error message tells user A that this record has been updated, and ALLY replaces the values shown on user A's display with the values stored in the database.

Table 3 shows the result of attempting to open a C-ISAM file in "new mode" when it has already been opened by another user (a different operating system process). In this table, Read/Write mode represents the default case—when no option has been selected.

Table 3. Compatibility of File Open Modes

	Previous Mode			
New Mode	**Read/ Write**	**Read/ Only**	**Exclusive Read**	**Exclusive Open**
Read/Write	OK*	OK	OK**	Fail
Read Only	OK	OK	OK	Fail
Exclusive Read	OK***	OK	Fail	Fail
Exclusive Open	Fail	Fail	Fail	Fail

* Attempts to modify the same record may result in lock time-out errors.

** Attempts to modify, delete, or insert records will result in lock time-out errors until the previous user closes the file.

*** If the previous user has records locked at the time the open is attempted, the open will fail. Otherwise, the previous user will be unable to modify the file until the new user closes the file.

When the same file is opened twice by the same operating-system process (for example, by two tasks within the same ALLY session), all open modes are compatible.

Sample C-ISAM Application

Suppose you are designing a C-ISAM application that will include a report that lists invoices, grouped by customer number. In the report, the customers must be listed in alphabetical order, and the invoices must be listed in numerical order.

There are two C-ISAM data files from which the data for this report can be taken. The "customers.dat" file has the following fields:

- name
- customer identification number

The "invoices.dat" file has the following fields:

- customer identification number
- invoice identification number
- date

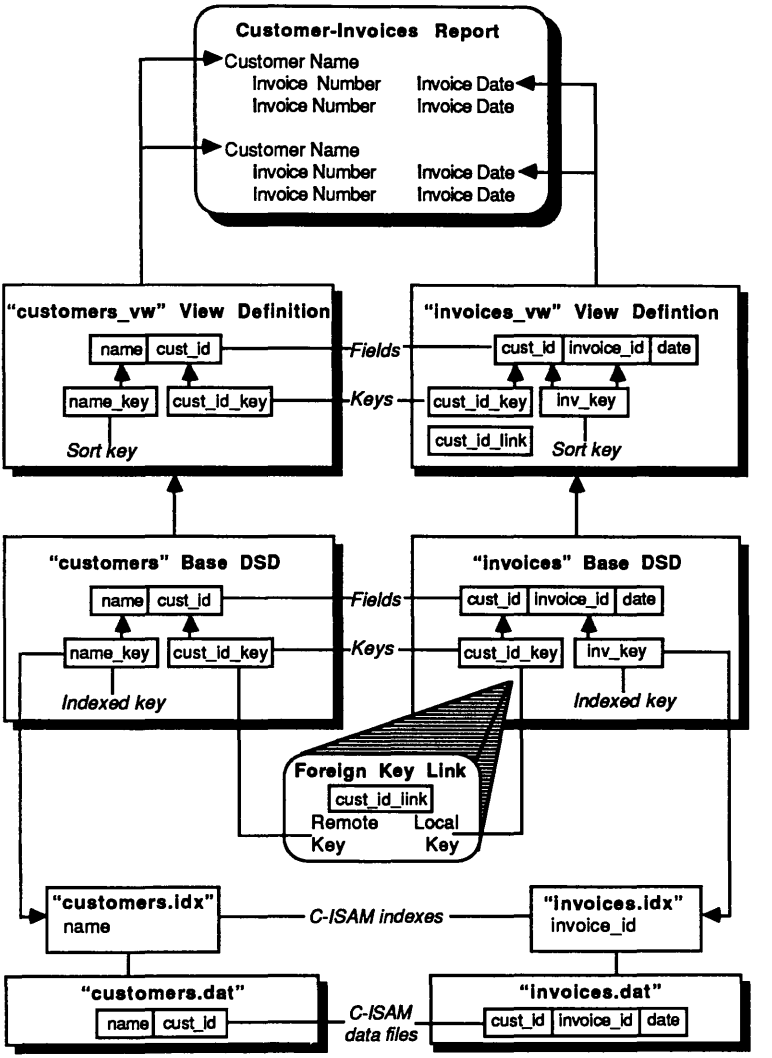
Because the two data files have one field in common, you can define a relationship between the files.

To build the application, you create two Base DSDs linked by a foreign key link, two View Definitions, and a form/report. Figure 13 shows the components of this application and the relationships between them. These are the items that comprise the application:

Customers Base DSD	This Base DSD has two fields that correspond to the two fields of the "customers.dat" file.
Name key	This indexed key is defined in the "customers" Base DSD. (At runtime it will produce the C-ISAM index "customers.idx.") The field assigned to this key is the "name" field.

Customer ID key	This key, defined in the "customers" Base DSD, will be used as the remote key in a foreign key link that establishes a relationship between the two Base DSDs. The field assigned to this key is the "cust_id" field, which is the field the two Base DSDs have in common.
Invoices Base DSD	This Base DSD has three fields that correspond to the three fields of the "invoices.dat" file.
Invoice key	This indexed key is defined in the "invoices" Base DSD. (At runtime, it will produce the C-ISAM index "invoices.idx.") The fields assigned to this key are the "cust_id" field and the "invoice_id" field. They must be assigned in this order.
Customer ID key	This key, defined in the "invoices" Base DSD, will be used as the local key in a foreign key link. The field assigned to this key is the "cust_id" field.
Customer ID link	This foreign key link is defined in the "invoices" Base DSD, which contains the subordinate records.
Customers View Definition	This View Definition references the "customers" Base DSD. It uses the indexed key (name_key) as its sort key.
Invoices View Definition	This View Definition references the "invoices" Base DSD. It uses the indexed key (inv_key) as its sort key. Its foreign key link is "cust_id_link."

Customer-Invoices Report	This form/report has a parent group containing a "customer name" field and a subordinate group containing an "invoice ID" field and a "date" field. The parent group references the "customers" View Definition, and the subordinate group references the "invoices" View Definition.

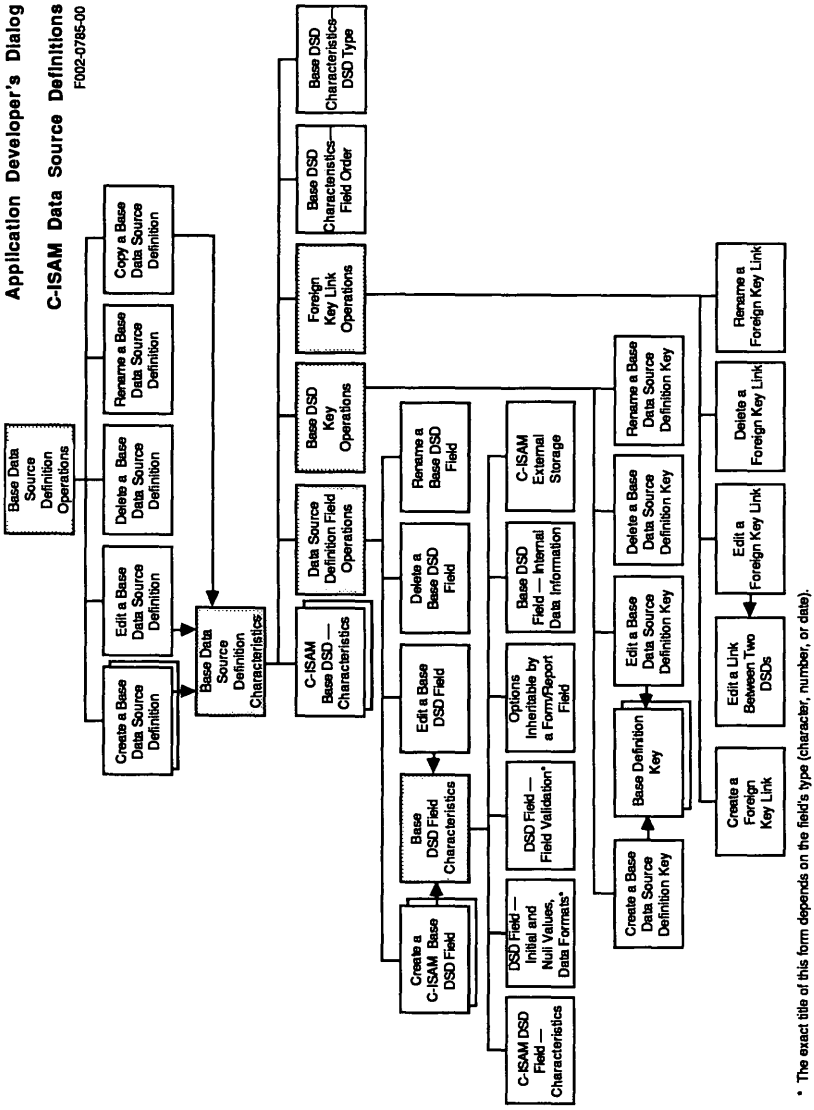


F002-0897-00

Figure 13. Sample Application

End of C-ISAM Developer Notes

Appendix A
Dialog Structure for C-ISAM DSDs



End of Appendix A

Index

End of Index