IMS DataPropagator for z/OS

Customization Guide

Version 3 Release 1

SC27-1214-00

IMS DataPropagator for z/OS

Customization Guide

Version 3 Release 1

SC27-1214-00

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 422.

First Edition (October 2001) (Softcopy Only)

This edition applies to Version 3 Release 1 of IMS DataPropagator, 5655-E52, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. This edition is available in softcopy format only. The technical changes for this edition are indicated by a vertical bar to the left of a change.

Contents

Figures

A

Preface

This book explains how to customize IMS DataPropagator (IMS DPROP) and is intended for use by system programmers.

This softcopy book is available only in PDF and BookManager formats. This book is available on the z/OS Software Products Collection Kit, SK3T-4270. You can also get the most current versions of the PDF and BookManager formats by going to the IBM Data Management Tools Web site at www.ibm.com/software/data/db2imstools and linking to the Library page.

What is New in Version 3, Release 1

IMS DataPropagator (IMS DPROP) Version 3, Release 1 presents improvements to both the product and the product library.

This edition, which is available in softcopy format only, includes technical and editorial changes.

Product Changes

IMS DataPropagator V3.1 provides a new, MQSeries-based asynchronous (MQ-ASYNC) propagation of IMS database changes to DB2 tables. With MQ-ASYNC enterprises can implement both:

- Near Real Time Propagation - With Near Real Time propagation, the delay between the update of the IMS database and the update of the DB2 tables can often be as short as a couple of seconds.
- Point-In-Time Propagation - With Point-In-Time propagation, the data content of the DB2 tables matches the IMS database content at a previous clearly identified logical point in time. For example, an enterprise may decide that the content of the DB2 tables will match the following point in times: the logical end of a business day, the logical end of a business month, or the end of a specific IMS jobstream that updated the IMS databases.

Product Library Changes

The Version 3.1 library has been updated with information about MQSeries asynchronous propagation. There are now three *Administrators Guide*s, one for each primary mode of propagation:

- *IMS DPROP Administrators Guide for MQSeries Asynchronous Propagation*
- *IMS DPROP Administrators Guide for Log Asynchronous Propagation*
- *IMS DPROP Administrators Guide for Synchronous Propagation*

There is also a new book, *IMS DataPropagator for z/OS: Concepts*, which provides a conceptual description of data propagation.

Special change indicators are used to identify information that is specific to LOG-ASYNC, MQ-ASYNC, and synchronous propagation:

- **Q** identifies information specific to MQ-ASYNC propagation.
- **A** identifies information specific to LOG-ASYNC propagation.

- **S** identifies information specific to synchronous propagation.

Terms Used in This Book

The following terms are synonymous in this book:

- *File* and *data set*
- *DXT* and *DataRefresher*

  Unless a specific version or release is referenced, these terms refer to either of the following products:

  – DXT Version 2 Release 5
  – DataRefresher Version 1 or higher
- Databases that have been *quiesced* or set to *READONLY status*.

  In all cases, these terms refer to either or both of the following:

  – Any propagatable database, except for DEDBs, that has been set to READONLY status.
  – DEDBs that have been taken offline with a /DBR command

References to DataRefresher and DXT in this book refer to only host activities. This book assumes that you will use batch and command statements, *not* the DataRefresher workstation component.

Selector and *Receiver* (capitalized) refer to the IMS DPROP Selector and Receiver features. However, *selector* and *receiver* (not capitalized) refer to user-created functions.

IMS DPROP books use the term “child” instead of the term “dependent.” For example, IMS DPROP books use the terms “child table” and “child rows” instead of DB2 terms “dependent table” and “dependent rows.” The term “child” is used so that terms for IMS and DB2 are similar.

What You Should Know

This book assumes you understand what data propagation is and the business reasons for propagating data. Information on these topics is in *IMS DPROP An Introduction*.

This book also assumes you have a basic understanding of IMS, DB2, and DataRefresher concepts and functions.

What is in This Book

The Version 3.1 Customization Guide provides information on how to write exit routines for your IMS DPROP system. It contains sample segment, field, and propagation exit routines that you can use. It also describes how to design and develop programs required to implement asynchronous propagation. The chapters are as follows:

- Chapter 1, “Introduction” on page 1
- Chapter 2, “Segment Exit Routines” on page 17

- Chapter 3, “Field Exit Routines” on page 110
- Chapter 4, “Propagation Exit Routines” on page 153
- Chapter 5, “DB2 Data Capture Subexit Routine” on page 259
- Chapter 6, “EKYRESLB Dynamic Allocation Exit Routine” on page 297
- Chapter 7, “TSMF Callable Interface” on page 314
- Chapter 8, “EMF Callable Interface” on page 319
- Chapter 9, “User-Implemented Asynchronous Data Propagation (USER-ASYNC)” on page 322

It also includes the following appendixes:

- Appendix A, “Calling the Trace Module” on page 343
- Appendix B, “Sample Segment Exit Control Blocks” on page 352
- Appendix C, “Sample Field Exit Control Blocks” on page 368
- Appendix D, “Sample Propagation Exit Control Blocks” on page 381

How to Read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this book:

Arrow symbols

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

►►───	Indicates the beginning of a statement.
───►	Indicates that the statement syntax is continued on the next line.
►───	Indicates that a statement is continued from the previous line.
───►◄	Indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the ►─── symbol and end with the ───► symbol.

Conventions

- Keywords, their allowable synonyms, and reserved parameters, appear in uppercase for MVS and OS/2 platforms, and lowercase for UNIX platforms. These items must be entered exactly as shown.
- Variables appear in lowercase italics (for example, *column-name*). They represent user-defined parameters or suboptions.
- When entering commands, separate parameters and keywords by at least one blank if there is no intervening punctuation.
- Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs) and numbers exactly as given.
- Footnotes are shown by a number in parentheses, for example, (1).
- A ƀ symbol indicates one blank position.

Required items

Required items appear on the horizontal line (the main path).

```
►►──REQUIRED_ITEM──────────────────────────────────────────────────────►◄
```

Optional Items

Optional items appear below the main path.

```
►►──REQUIRED_ITEM──┬───────────────┬───────────────────────────────────►◄
                   └─optional_item─┘
```

If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

```
                   ┌─optional_item─┐
►►──REQUIRED_ITEM──┴───────────────┴───────────────────────────────────►◄
```

Multiple required or optional items

If you can choose from two or more items, they appear vertically in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.

```
►►──REQUIRED_ITEM──┬─required_choice1─┬────────────────────────────────►◄
                   └─required_choice2─┘
```

If choosing one of the items is optional, the entire stack appears below the main path.

```
►►──REQUIRED_ITEM──┬──────────────────┬────────────────────────────────►◄
                   ├─optional_choice1─┤
                   └─optional_choice2─┘
```

Repeatable items

An arrow returning to the left above the main line indicates that an item can be repeated.

```
                   ┌─────────────────┐
►►──REQUIRED_ITEM──▼─repeatable_item─┴─────────────────────────────────►◄
```

If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
                   ┌─,───────────────┐
►►──REQUIRED_ITEM──▼─repeatable_item─┴─────────────────────────────────►◄
```

A repeat arrow above a stack indicates that you can specify more than one of the choices in the stack.

Default keywords

IBM-supplied default keywords appear above the main path, and the remaining choices are shown below the main path. In the parameter list following the syntax diagram, the default choices are underlined.

```
                   ┌─default_choice──┐
►►──REQUIRED_ITEM──┼─────────────────┼─────────────────────────────────►◄
                   ├─optional_choice─┤
                   └─optional_choice─┘
```

IMS-specific syntax information

Fragments

Sometimes a diagram must be split into fragments. The fragments are represented by a letter or fragment name, set off like this: | A |. The fragment follows the end of the main diagram. The following example shows the use of a fragment.

```
►►──STATEMENT──item 1──item 2──┤ A ├──────────────────────────────────────►◄
```

A:

```
├──┬─item 3─┬──KEYWORD──┬─item 5─┬──────────────────────────────────────┤
   └─item 4─┘           └─item 6─┘
```

Substitution-block

Sometimes a set of several parameters is represented by a substitution-block such as <**A**>. For example, in the imaginary /VERB command you could enter `/VERB LINE 1`, `/VERB EITHER LINE 1`, or `/VERB OR LINE 1`.

```
►►──/VERB──┬─────┬──LINE──line#──────────────────────────────────────────►◄
           └─<A>─┘
```

where <A> is:

```
►►──┬─EITHER─┬───────────────────────────────────────────────────────────►◄
    └─OR─────┘
```

Parameter endings

Parameters with number values end with the symbol '#', parameters that are names end with 'name', and parameters that can be generic end with '*'.

```
►►──/MSVERIFY──┬─MSNAME──msname─┬────────────────────────────────────────►◄
               └─SYSID──sysid#──┘
```

The MSNAME keyword in the example supports a name value and the SYSID keyword supports a number value.

Chapter 1. Introduction

This chapter introduces the routines you can use to customize IMS DPROP. These routines are:

- Segment exit routine
- Field exit routine
- Propagation exit routine
- DB2 Data Capture subexit routine
- EKYRESLB Dynamic Allocation exit routine
- Timestamp marker facility callable interface
- User Asynchronous programs

Information about coding the programs in high-level languages is also included. The rest of this book describes the programs in detail.

Segment, Field, and Propagation Exit Routines

If DPROP mapping and conversion capabilities do not meet your needs, you can use the following exit routines for special situations:

- Segment exit routines
- Field exit routines
- Propagation exit routines

These routines can be written in either Assembler language or one of the following high-level languages: COBOL, PL/I, and C. DPROP support for exit routines written in high-level languages requires Language Environment/370 (LE/370) Version 1 Release 2. See *IMS DPROP An Introduction* for a description of the software requirements for the LE/370 environment.

Segment and Field exit routines complement the generalized mapping logic of the RUP and HUP. They perform special data formatting that the RUP and HUP do not support. When called, a Segment exit routine reformats an entire IMS segment, while a Field exit routine reformats individual fields in a segment.

Figure 1 on page 2 illustrates the sequence in which the Segment exit routine, Field exit routines, and DPROP conversion routines are invoked by the RUP and HUP.

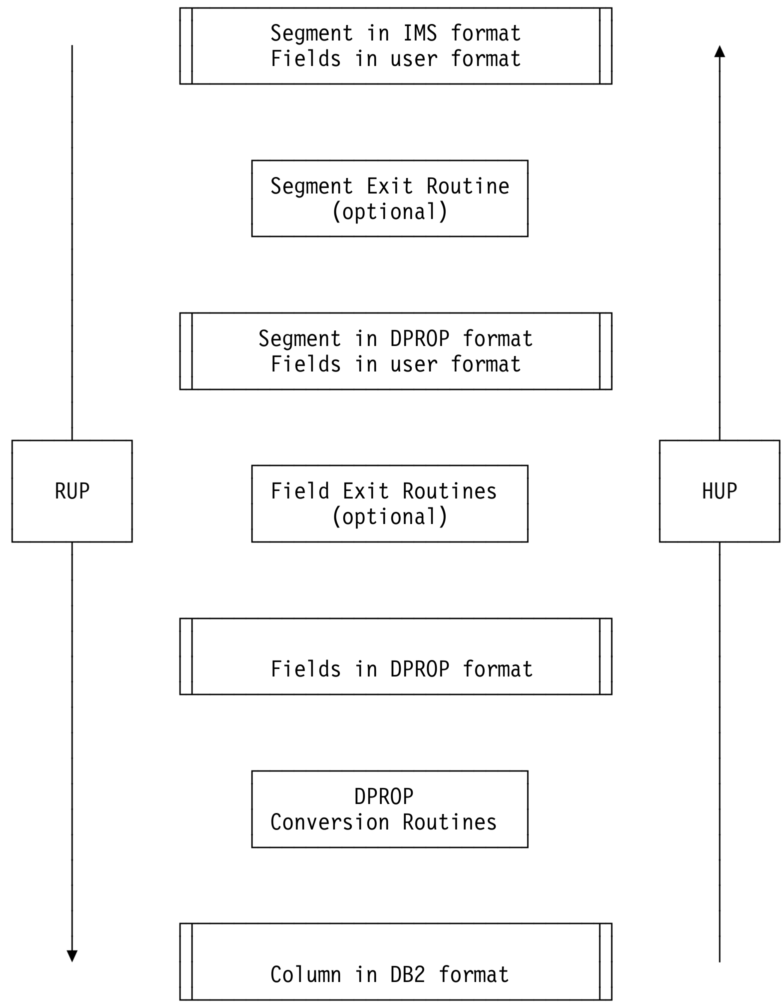

Figure 1. Sequence of Conversion by Segment and Field Exit Routines

You may find that your mapping or propagation requirements cannot be handled by combining the generalized mapping logic of DPROP with Segment and Field exit routines. In this case, you may want to use a Propagation exit routine, which lets you substitute your own mapping logic for the generalized mapping logic of the RUP and HUP.

DPROP calls exit routines in both synchronous and LOG-ASYNC modes. During synchronous propagation, the RUP and HUP can call the exit routines from both IMS batch and dependent regions. For generalized mapping cases, Segment and Field exit routines are also called during execution of the Consistency Check utility (CCU) and DPROP DL/I Load utilities (DLU).

If you extract data with DataRefresher, the Segment and Field exit routines are also called by DataRefresher.

Segment Exit Routine

The Segment exit routine is generally used to map an IMS segment between an IMS database format that DPROP does not support and a DPROP-supported format.

The Segment exit routine can change the format and positions of data fields in an IMS segment. It cannot change the format and position of keys, including the concatenated key, nor can it change the format and position of any fields mapped to the primary key of the related DB2 row.

Segment exit routines can:

- Map IMS segments that have fields with variable starting positions into a segment format where starting positions are fixed.
- Clean up data, such as data stored in redefined areas of IMS segments.
- Selectively suppress propagation based on selection criteria programmed into the exit routine. For IMS-to-DB2 propagation, it is preferable, where possible, to selectively suppress propagation by defining a WHERE clause during PR definition.

Segment exit routines used with PRTYPE=L (limited function) PRs are called only for IMS-to-DB2 propagation and must, therefore, support only IMS-to-DPROP mapping.

Segment exit routines used with PRTYPE=E (extended function) PRs must support both IMS-to-DPROP and DPROP-to-IMS mapping, even if the PRTYPE=E PR specifies MAPDIR=HR. This is because your Segment exit routine can be called during CCU and DLU processing to do DPROP-to-IMS mapping. The conversion done during DPROP-to-IMS mapping should be the opposite of the conversion done during IMS-to-DPROP mapping.

If you are using DataRefresher to extract IMS data, the Segment exit routine is also called by DataRefresher as a data type exit routine.

For additional information about the Segment exit routine, see Chapter 2, “Segment Exit Routines” on page 17. For information about data type exit routines, see the appropriate DataRefresher or DXT documentation.

Field Exit Routine

The Field exit routine is generally used to map a field between its IMS database format (referred to as a user format) and a DPROP-supported format.

Field exit routines are used:

- For IMS segment fields that have special formats not supported by DPROP, and that cannot be converted by the DPROP conversion routines. Examples of such fields are:
  - Date and time formats other than USA, ISO, EUR, and JIS, which must be converted into a standard format
  - Unsigned, packed numeric fields
  - Encoded data, such as a two-byte state code that is to be expanded

- When the format of the IMS field cannot be directly converted by DPROP to the format of the DB2 column, such as converting a character format to a numeric format, or converting a character field to a DBCS field.
- To convert some values in an IMS field to a DB2 null value.
- To change the contents or restructure the data in the field before storing it in the corresponding DB2 table.
- To alter the contents of a key field.
- When performing DB2-to-IMS propagation, to convert the value of a numeric DB2 column into a packed or zoned IMS field having a sign code other than the “preferred” sign codes `X'C'` and `X'D'`.

Field exit routines used with PRTYPE=L PRs are only called for IMS-to-DB2 propagation and must therefore only support user-to-DPROP mapping. They are called to convert an IMS field from your format in the IMS database to the format supported by and defined to DPROP.

Field exit routines used with PRTYPE=E PRs must support both user-to-DPROP mapping and DPROP-to-user mapping, even if the PRTYPE=E PR specifies MAPDIR=HR. This is because your Field exit routine can be called during CCU and DLU processing to do DPROP-to-user mapping. The conversion done during DPROP-to-user mapping should be the opposite of the conversion done during user-to-DPROP mapping.

If you are using DataRefresher to extract IMS data, the Field exit routine is also called by DataRefresher as a data type exit routine.

For additional information about the Field exit routine, see Chapter 3, “Field Exit Routines” on page 110. For information about data type exit routines, see the appropriate DataRefresher or DXT documentation.

Propagation Exit Routine

If the DPROP generalized mapping cases cannot be used for propagation, you can supply your own mapping in a Propagation exit routine. Propagation exit routines must provide all necessary mapping logic, build the SQL* calls needed for propagation to DB2, and build the IMS calls needed for propagation to IMS. Neither DataRefresher nor the DLU call Propagation exit routines during the extract/load phase.

For additional information about the Propagation exit routine, see Chapter 4, “Propagation Exit Routines” on page 153.

Propagation Exit Routine or IMS Data Capture Exit Routine

Using Propagation exit routines to propagate data from DPROP has some advantages over propagating data using an IMS Data Capture exit routine that you write. These advantages include:

- Propagation debugging support provided by DPROP
- Centralized error handling through the RUP and HUP
- Simplified operation of propagation since DPROP can be used to suspend and restart propagation

- Protection against unintentional updates during the extract/load phase of propagation
- Centralized control point for PR definitions (the DPROP directory tables)
- Common process for managing the data propagation environment for both generalized and user mapping cases

Overview of RUP and Exit Routine Processing

For each updated segment occurrence, the RUP is called once by the IMS Data Capture function. A particular segment type can be propagated by zero, one, or several PRs. The number of PRs can be zero if you changed the DBD with an EXIT= keyword, but have not yet generated PRs.

If the updated segment type is propagated by multiple PRs, the RUP will sequentially process these PRs within a single call by the IMS Data Capture function.

For each PR, the RUP checks the PR status to determine if the PR should be processed. Inactive PRs are not processed. Then the RUP determines if the PR specifies a generalized or user mapping case.

For a PR belonging to a generalized mapping case:

1. The RUP calls the optional Segment exit routine. The Segment exit routine converts the IMS segment from its IMS database format to the format supported by and defined to DPROP.

   For some PRs (for example, those defined with a WHERE clause, those propagating IMS segments that contain embedded structures, or those attempting to avoid unnecessary SQL updates by specifying AVU=Y), the Segment exit routine is called twice by the RUP during replace operations: once to convert the segment before replacement, and a second time to convert the segment after replacement.
2. For each field requiring it, the RUP calls the appropriate optional Field exit routine. The Field exit routine converts the field from its user format to the format supported by and defined to DPROP.
3. The RUP converts each field into its DB2 column format.
4. The RUP issues the propagating SQL statement by calling the appropriate SQL update module, which was generated when the PR was defined.

For a PR using a Propagation exit routine (user mapping), the RUP calls the Propagation exit routine. The Propagation exit routine is responsible for all required mapping, conversions, and propagating SQL statements.

Figure 2 on page 6 shows RUP processing for a generalized mapping case, including the relationship with Segment and Field exit routines. Figure 3 on page 6 shows RUP processing for user mapping with a Propagation exit routine.

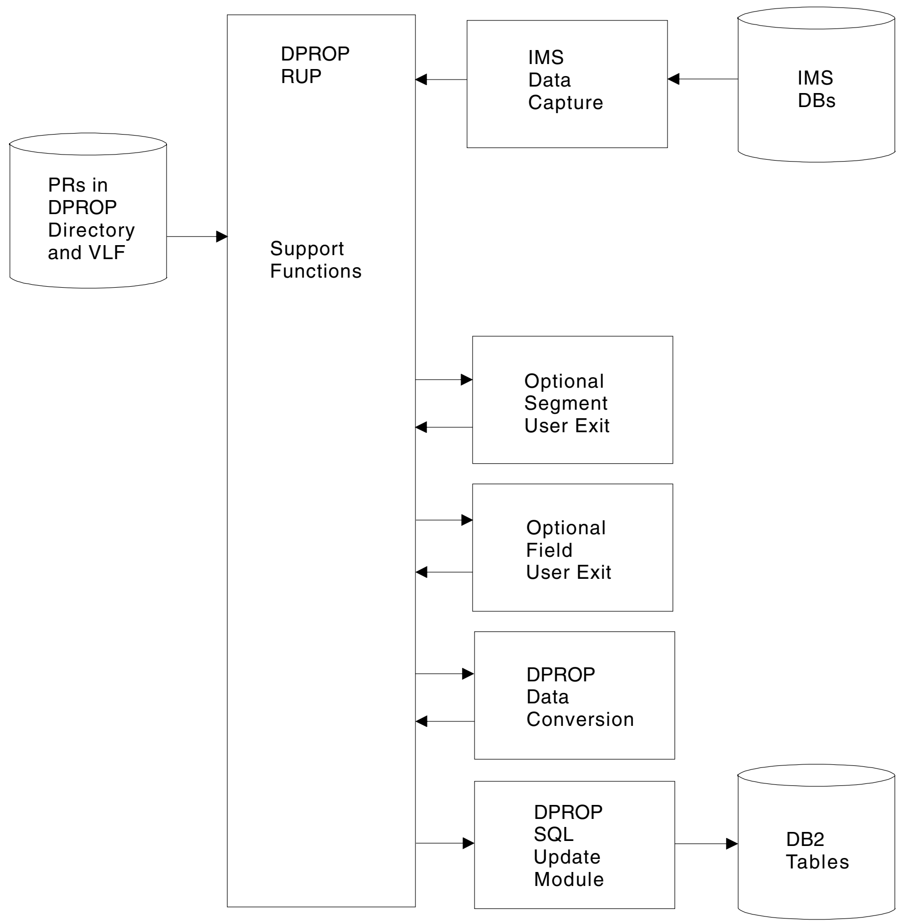

Figure 2. RUP Processing for Generalized Mapping

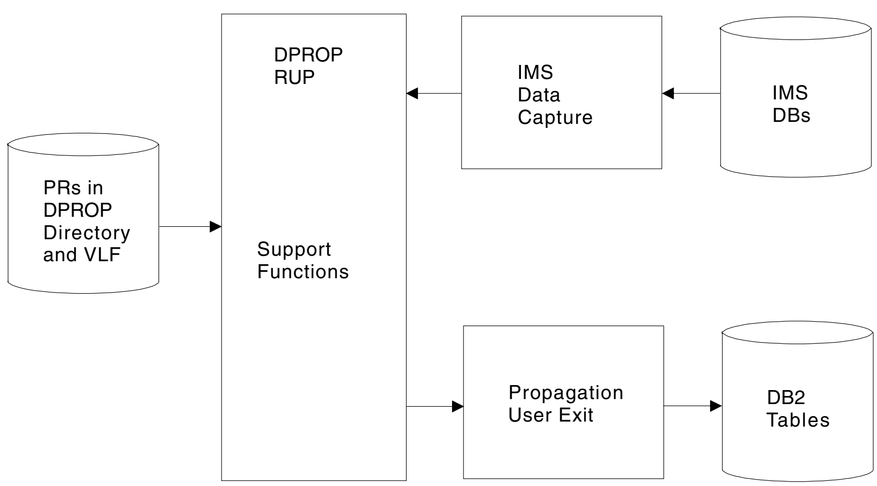

Figure 3. RUP Processing for User Mapping

Overview of the HUP and Exit Routine Processing

The HUP runs as the IBM-supplied DB2 Data Capture exit routine. The HUP is called when the DB2 Data Capture function detects that an SQL update changes rows of tables that have been defined with the DATA CAPTURE parameter and when DB2 tracing for MONITOR CLASS(6) is active. You must also set the DB2 system parameter DPROP SUPPORT to 2 or 3, otherwise no call back to IMS occurs.

The HUP obtains all changed rows of these tables from DB2. Then, based on your PR definitions, the HUP determines whether and how the changed rows of a particular table should be propagated.

The HUP checks the PR status to determine if the PR should be processed. Inactive PRs are not processed. Then the HUP determines if the PR specifies a generalized or user mapping case.

For a PR belonging to a generalized mapping case:

1. The HUP converts each DB2 column into the DPROP-supported field format that you specified in the PR definition.
2. For each field requiring it, the HUP calls the appropriate optional Field exit routine. The Field exit routine converts the field from the format supported by and defined to DPROP into its user format.
3. Then the HUP builds the IMS segment search arguments (SSAs) required to access the target IMS database segment.
4. If the IMS target segment needs to be replaced or deleted, the HUP issues an IMS GHU (get hold unique) call to retrieve the segment.

   If the IMS target segment to be replaced is processed by a Segment exit routine, and some fields are not propagated, the HUP initially calls the Segment exit routine. The Segment exit routine must convert the retrieved IMS segment from its IMS database format into the format you defined to DPROP. The conversion done by your Segment exit routine should be the same as the conversion done during RUP calls for IMS-to-DB2 propagation. This processing is used to merge nonpropagated fields in the original IMS segment with the updated fields propagated from DB2.
5. If the target IMS segment will be replaced or inserted, the HUP builds the new segment image. If you have not specified use of a Segment exit routine, the segment image has the format of the IMS database segment. Otherwise, the segment image has the format you defined to DPROP.

   For IMS segments that do not contain propagated internal segments, the HUP builds the image of the IMS segment. In the other cases, the HUP builds the image of either the internal or containing segment.
6. The HUP calls the optional Segment exit routine.
   - If the IMS segment **does not** contain propagated internal segments, the Segment exit routine converts the IMS segment from the format supported by and defined to DPROP to the IMS database format. The conversion done by your exit routine should be the reverse of the mapping done during RUP calls for IMS-to-DB2 propagation.

- If the IMS segment **does** contain propagated internal segments, the Segment exit routine must merge the internal/containing segment formatted by DPROP to the existing IMS segment that was previously retrieved by the HUP (see item 4 on page 7).

7. The HUP issues the DL/I update calls that propagate the DB2 change.

For a PR using a Propagation exit routine (user mapping), the HUP calls the Propagation exit routine. The Propagation exit routine does all required mapping, conversions, and propagation of DL/I update calls.

After propagation of the changed DB2 row, the HUP calls your optional DB2 Data Capture subexit routine.

Figure 4 shows HUP processing for a generalized mapping case, including the relationship with Field and Segment exit routines. Figure 5 on page 9 shows HUP processing for user mapping with a Propagation exit routine.

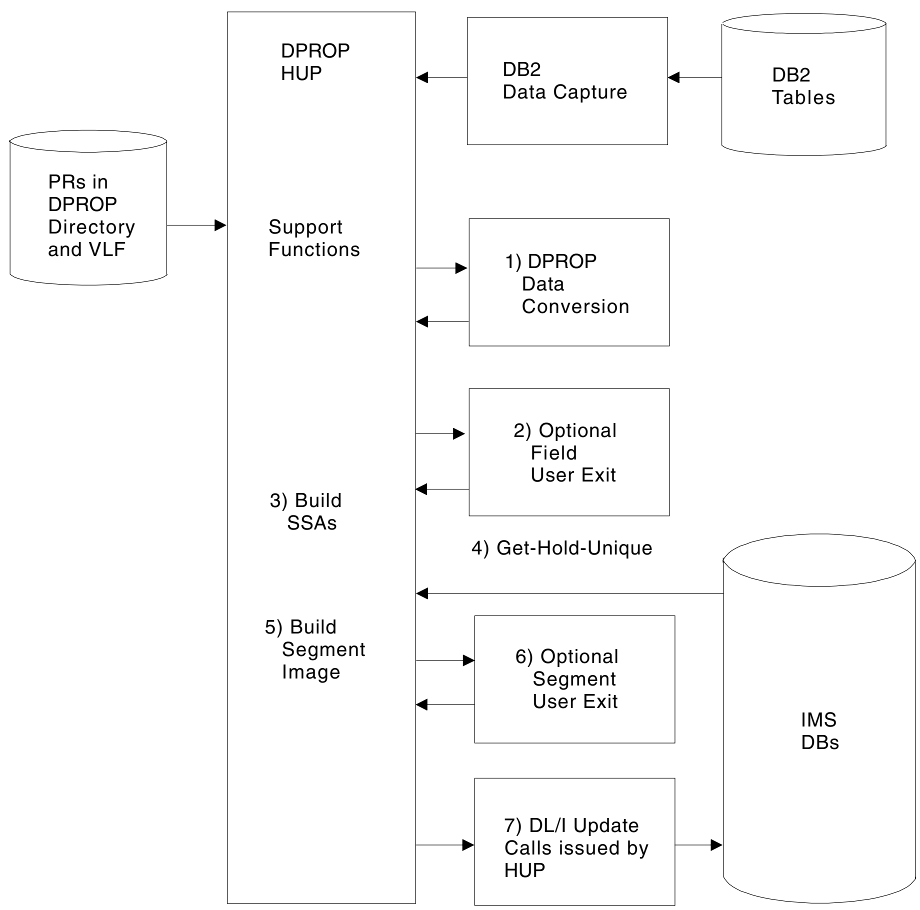

Figure 4. HUP Processing for Generalized Mapping Logic

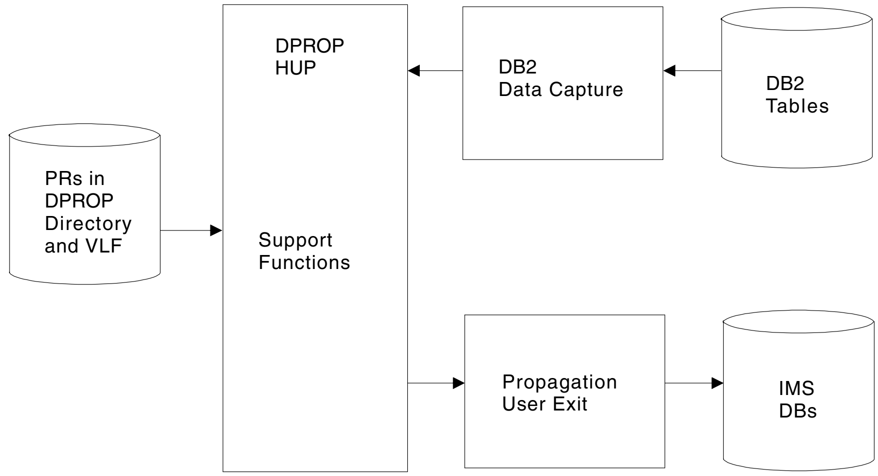

Figure 5. HUP Processing for User Mapping

Error Handling Logic of Exit Routines

The exit routines may encounter error situations. For example, a field defined as numeric may contain nonnumeric data. In this case, the exit routines should use the error handling logic of the RUP and HUP. This practice has the following advantages:

- The error option (ERROPT) in effect is used when the exit routine encounters an error.
- Errors are traced and placed on an audit trail for later review if desired.

To take advantage of the error handling logic of the RUP and HUP, your exit routine should:

- Signal propagation failures to RUP/HUP using a return code in the provided interface control block. Your exit routine should not issue an abend if you want to use the error handling logic of the RUP/HUP.
- Provide error or warning messages in the interface control block to help diagnose the problem. Your exit routine should not issue messages directly.

To change versions of an exit routine, the job step from which the exit routine is called must be stopped and restarted with the new version available.

Exit Routine Relationship to DataRefresher

This section describes the relationship between DataRefresher and the exit routines.

Segment and Field Exit Routines

DPROP calls Segment and Field exit routines during propagation. DataRefresher calls them during the extract/load when extracts are done by DataRefresher. This lets you have identical mapping for extracts done by DataRefresher and propagation done by DPROP. DataRefresher calls these routines data exits and data type exits, respectively.

There are some special restrictions and requirements for exit routines called by DPROP. For example, while the interface control blocks to the exit routines are identical for DPROP and DataRefresher, DPROP does not initialize all of the fields in the control blocks. Another example of these restrictions is that no SYSPRINT DCB is furnished to the exit routine by DPROP. The additional restrictions and requirements are discussed with each type of exit routine in that routine's chapter of this book.

Propagation Exit Routines

DataRefresher does not support Propagation exit routines. If you are using Propagation exit routines for user mapping, DataRefresher will not call your Propagation exit during the extract/load phase. If you determine that you can use the mapping capabilities of DataRefresher for the extract/load, the mapping logic of your Propagation exit routine must be compatible with that of DataRefresher Otherwise, you write your own extract program providing the same mapping logic as your Propagation exit routine.

DB2 Data Capture Subexit Routine

If your installation requires that the HUP coexist with another generalized DB2 Data Capture exit routine, consider writing a DB2 Data Capture subexit routine. Instead of having two DB2 Data Capture exit routines (which is not supported by DB2), you would:

- Use the HUP as a DB2 Data Capture exit routine, and
- Define to DPROP the “other” generalized exit routine as a DB2 Data Capture subexit routine (its name is defined during DPROP installation).

The purpose of the subexit routine is usually not DB2-to-IMS propagation. Instead, its purpose is usually to:

- Propagate changed DB2 rows to other tables, or
- Perform other generalized functions, such as auditing changed DB2 rows.

DPROP calls your subexit routine when the DB2 Data Capture function calls the HUP. DPROP calls the subexit routine even if you have not defined a PR and even if propagation has been emergency stopped.

The HUP calls your subexit routine once for each changed row and gives it both the data and the description of the changed row. The HUP calls your subexit routine **after** processing of all DPROP PRs. However, your subexit routine is **not** called when the HUP issues a rollback of the unit of work or an abend. This is not a problem since, in this case, the SQL update can be considered nonexistent.

You can write your DB2 Changed Data Capture subexit routine in a high-level language, such as C, COBOL, and PL/I.

For additional information about the DB2 Data Capture Subexit routine, see Chapter 5, “DB2 Data Capture Subexit Routine” on page 259.

EKYRESLB Dynamic Allocation Exit Routine

You can write an EKYRESLB Dynamic Allocation exit routine to dynamically allocate the APF-authorized library containing DPROP load modules. You can do this if the following methods of allocation are inappropriate for your installation:

- Allocation using an //EKYRESLB DD statement in the JCL of propagating job steps and DPROP utility job steps
- Dynamic allocation by DPROP to a data set name specified during DPROP installation.

Note that you cannot write the EKYRESLB Dynamic Allocation exit routine in a high-level language.

For more information about EKYRESLB Dynamic Allocation exit routine, see Chapter 6, “EKYRESLB Dynamic Allocation Exit Routine” on page 297.

General Considerations for Exit Routines

When called during synchronous propagation, the exit routines execute in the same environment as the propagating application program. The exit routines can issue the same IMS calls and SQL statements as the application. However, IMS and DB2 updates issued by the exit routines are not propagated. A possible exception to this would be IMS updates issued during DB2-to-IMS propagation; they can be propagated asynchronously if LOG-ASYNC propagation is based on the IMS Asynchronous Data Capture function. This is because IMS and DB2 calls issued from the IMS or DB2 Data Capture function do not result in recursive calls to Data Capture exit routines. To the IMS and DB2 Data Capture functions, DPROP's exits appear to run as an extension to the RUP and HUP.

The exit routines must not perform functions incompatible with the environments in which they execute. For example, they should not write to MVS data sets from IMS message processing regions. The exit routines should also avoid using services that can impact the performance of propagating application programs. Examples could include the OPEN macro issued from IMS message processing regions.

DPROP Release 2 supports exit routines both written in Assembler language and with high-level language compilers supporting LE/370 Version 1 Release 2. Exit routines must receive and return control in AMODE 31, but their execution RMODE can be ANY. The addresses of parameters passed by DPROP to the exit routines are 31 bit, and the parameters are usually located above the 16MB line.

TSMF Callable Interface

When you are using DPROP LOG-ASYNC propagation, you must set an initial start time to run the Selector. Subsequently, the Selector can determine its own start and stop times. Alternatively, you can specify Group/database start times and group stop times for each Selector run. For details on Selector start and stop times, refer to the appropriate *Administrators Guide* for your propagation mode.

The timestamp marker facility (TSMF) allows you to specify timestamp markers (TSMs) to be used by the Selector for group/database start times and group stop times. The TSMF can be invoked as a batch job. It can also be invoked through a callable interface to allow a user application to insert a stop TSM in the Selector control file for a specified propagation group. The TSMF callable interface is described in detail in Chapter 7, "TSMF Callable Interface" on page 314.

Note: The TSMF Callable Interface can only be used to create group stop times.

EMF Callable Interface

MQ-ASYNC supports both a Near Real Time Propagation and a Point-In-Time Propagation.

For a Point-In-Time Propagation with MQ-ASYNC, you create Event Markers on the Source System. Each Event Marker identifies a particular Source System Point In Time. The Event Markers are transmitted in MQSeries messages, together with the IMS DB Changes, in First-In-First-Out order to the Apply Programs on the Target System.

An Apply Program can be instructed to stop its processing when it reads an MQSeries message containing a specific Event Marker. When an Apply Program is stopped in this way, the content of the target DB2 tables reflects the Source System Point-in-Time that has been identified by the creation of the Event Marker.

Usually, the Event Markers are created by running the IMS DPROP 'Capture System Utility (CUT)'. As an alternative, Event Markers can also be created by application programs through use of a callable interface.

User Asynchronous Programs

IMS DataPropagator Version 3 implements LOG-ASYNC IMS-to-DB2 propagation. You can implement user LOG-ASYNC IMS-to-DB2 propagation in one of the following ways:

- Using the IMS Asynchronous Data Capture function. In this case, segment updates are written by this function to the IMS log. Later, programs you write (often called the "selector") gather and select the changed data from the IMS log data sets. Another program you write (often called the *receiver*) reads the changed data and calls DPROP with it. Propagation is then done either by DPROP (if you used the generalized mapping cases) or by your Propagation exit routine (if you used user mapping). See Chapter 9, "User-Implemented Asynchronous Data Propagation (USER-ASYNC)" on page 322 for additional information.

- Using a Data Capture exit routine you write (often called the *sender*). In this case, segment updates are written by your routine to the:

  – IMS log
  – IMS full-function database
  – DEDB sequential dependent segments
  – MVS flat file

  Another program you write (often called the receiver) does the same processing as described in the preceding paragraph.

The LOG-ASYNC sender and receiver programs used in these two implementations are described and illustrated in Chapter 9, "User-Implemented Asynchronous Data Propagation (USER-ASYNC)" on page 322.

Coding Exit Routines in High Level Languages

All DPROP User exit routines can be written in Assembler. In addition, the following types of exit routines can be written in COBOL, PL/I, and C:

- Segment exit routines
- Field exit routines
- Propagation exit routines
- DB2 Data Capture subexit routine

DPROP support for user exit routines written in high-level languages (HLL) requires the following:

- LE/370 must be installed, and LE/370 modules must be available (through //STEPLIB, //JOBLIB, LINKLIB, or LPA concatenation) to the job steps where the exits are executed.
- Exit routines written in COBOL must be compiled with the SAA AD/Cycle COBOL/370 Version 1 Release 1 (or later releases) compiler.
- Exit routines written in PL/I must be compiled with the LE/370 Version 1 Release 1 (or later releases) compiler.
- Exit routines written in C must be compiled with the LE/370 Version 1 Release 1 (or later releases) compiler.

Preinitializing an HLL Environment

DPROP uses LE/370 support for preinitialization to call exit routines written in high-level languages. LE/370 preinitialization allows the HLL environment to be initialized once and to perform multiple executions of HLL exit routines using this preinitialized environment.

DPROP initialization triggers the preinitialization of the HLL environment. DPROP uses an interface to LE/370 that is similar to the CEEPIPI environment created with an INIT-SUB call. The resulting LE/370 enclave is used to execute all HLL DPROP exit routines.

Then, when one of your exit routines needs to be called, DPROP determines whether it was compiled with one of the previously listed HLL compilers, and proceeds as follows:

- If it was compiled, DPROP calls your exit routine through the CEEPIPI interface module. Your exit routines are called in the LE/370 enclave used for the HLL DPROP exit routines, as opposed to the LE/370 enclave of your propagating application programs. LE/370 treats your exit routines as subroutines, as opposed to main programs.
- If your exit routines were not compiled with one of the above compilers, DPROP assumes that they are Assembler exit routines, and calls them according to Assembler conventions.
- If LE/370 is installed and available through the //JOBLIB, //STEPLIB, linklist, or LPA concatenation, do not provide HLL exit routines compiled with compilers other than those listed above, because the results are unpredictable.

Specifying LE/370 Runtime Options

It is possible that both your exit routines and your applications will be coded in high-level languages. In most cases, each will use LE/370 runtime libraries and options. In such a situation, there are two enclaves: one for the exit, and one for the application. Note that the two enclaves operate under a different set of rules:

- The *exit* operates under the rules for a routine invoked through CEEPIPI preinitialization.
- The *application* operates under the LE/370 rules for a main routine.

Additionally, the enclaves are separate in terms of storage use. Each enclave has its own storage, based on installation defaults or overriding runtime options. For performance and storage use, ensure that each enclave has sufficient storage allocations.

The //EKYLEOPT DD Statement

DPROP allows you to specify LE/370 runtime options for the HLL DPROP exit routines' enclave. You can provide the runtime options in the //EKYLEOPT DD statements of the various DPROP job steps (described in *IMS DPROP Reference*). DPROP provides these runtime options to CEEPIPI during the **INIT SUB** call. Runtime options provided in //EKYLEOPT override the installation defaults that were defined in the CEEDOPT LE/370 module.

You cannot link runtime options with a particular exit, as you can with an application.

The TRAP Runtime Option

In an environment where both your exit routines and applications are coded in high-level languages, you should be aware of issues concerning the LE/370 TRAP runtime option. This runtime option is a combined ESTAE/ESPIE setting that you can set ON or OFF. If the TRAP option is set ON, a DPROP abend could be *trapped* by the LE/370 ESTAE/ESPIE mechanism. However, because LE/370 is not in control, a 4036 abend will likely result. In this case, rerun the failing situation with the TRAP option set to OFF to find the underlying abend.

For more information on LE/370, see OS/390 Language Environment Programming Guide. For more information on diagnosing DPROP problems, see *IMS DPROP Diagnosis*.

LE/370 and DPROP Installation

During DPROP installation, your DPROP system administrator can create a dummy, IEFBR14-type, CEEPIPI load module in your DPROP RESLIB. This is done if DPROP installation occurs before LE/370 installation. Creating the dummy CEEPIPI module prevents a large number of CSV003I messages informing you that the CEEPIPI module was not found. If the dummy CEEPIPI module was copied to the DPROP RESLIB during DPROP installation and you install LE/370 at a later time, do one of the following to enable DPROP to support LE/370:

- Delete the dummy, IEFBR14-type, CEEPIPI load module from the DPROP RESLIB. The LE/370 load modules must be available to DPROP (through //JOBLIB, //STEPLIB, LINKLIB, and LPA concatenation).
- Concatenate the load module library containing LE/370 modules ahead of the DPROP RESLIB in the //JOBLIB, //STEPLIB, LINKLIB or LPA concatenation.

Additional Requirements and Recommendations For COBOL

DPROP exit routines written in COBOL must be compiled with the **RENT** option.

Additional Requirements and Recommendations For PL/I

The PROCEDURE statement for a PL/I exit must include processing characteristics, as shown in the following example:

```
OPTIONS(FETCHABLE REENTRANT)
```

Refer to *IBM SAA AD/Cycle PL/I MVS & VM Language Reference* for more information.

The sample exit in "Third Sample Segment Exit Routine" on page 97 and the control blocks in Appendix B, "Sample Segment Exit Control Blocks" on page 352, Appendix C, "Sample Field Exit Control Blocks" on page 368, and Appendix D, "Sample Propagation Exit Control Blocks" on page 381 were coded with source code between columns 2 and 72. Column 1 is used for carriage control. Page ejects are inserted to make compiled sample listings more readable. To set this up in your exit routines, specify the MARGINS compiler option when compiling a PL/I exit, as follows:

```
MARGINS(2,72,1)
```

Refer to *IBM SAA AD/Cycle PL/I MVS & VM Programming Guide* for more information.

Additional Requirements and Recommendations For C

To establish correct linkage, all C language exits must include the following PRAGMA:

```
#PRAGMA LINKAGE(exitname, FETCHABLE)
```

Where:

EXITNAME
Is the name of the user exit (field, segment, or propagation).

The sample propagation exit routine in “Sample Exit Routine Source Code” on page 235 uses LE/370 callable services (CEETDLI). To call LE/370 callable services, add the following INCLUDE statement to your C language source code:

```
#INCLUDE "LEAWI.H"
```

Refer to *OS/390 Language Environment Programming Reference* for additional information about LE/370 callable services.

Carriage control within the sample exit in “Sample Exit Routine Source Code” on page 235 and in the control blocks in Appendix B, “Sample Segment Exit Control Blocks” on page 352, Appendix C, “Sample Field Exit Control Blocks” on page 368, and Appendix D, “Sample Propagation Exit Control Blocks” on page 381 was forced using PRAGMA:

```
#PRAGMA PAGE(1)
```

Page ejects were inserted to make compiled sample listings more readable.

Chapter 2. Segment Exit Routines

The RUP and HUP call a Segment exit routine as part of DPROP's generalized mapping logic processing. This exit routine is required for TYPE=E PRs that propagate IMS segments containing internal segments; it is optional for other PRs.

A Segment exit routine can be used to reformat or change the segment data during propagation. The RUP's or HUP's generalized mapping logic can take care of most situations, but if your data is stored in an unusual way or in some form that the RUP or HUP cannot handle, consider writing a Segment exit routine.

A Segment exit routine converts a segment between an IMS database format that DPROP does not support and the DPROP-supported format that you define in your PR. This is further referenced as:

- ***IMS-to-DPROP mapping*** or ***normal call*** when your exit routine is called to convert the segment from its IMS database format to the DPROP format. Calls to your exit routine for IMS-to-DPROP mapping are generated primarily by the RUP as part of IMS-to-DB2 propagation, and under some circumstances also by the HUP as part of DB2-to-IMS propagation.
- ***DPROP-to-IMS mapping*** or ***reverse call*** when your exit routine is called to convert the segment from its DPROP format to the IMS database format. Calls to your exit routine for DPROP-to-IMS mapping are only generated by the HUP as part of DB2-to-IMS propagation.

The conversion performed during DPROP-to-IMS mapping must be the reverse of the conversion performed during IMS-to-DPROP mapping.

Segment exit routines used with TYPE=L or TYPE=F PRs must support IMS-to-DPROP mapping; they do not need to support DPROP-to-IMS mapping.

Segment exit routines used with TYPE=E PRs must support both IMS-to-DPROP mapping and DPROP-to-IMS mapping, even if the TYPE=E PR specifies MAPDIR=HR. This is because the HUP may call your Segment exit routine and request DPROP-to-IMS mapping during CCU and DLU processing.

A Segment exit routine can be used to:

- Reorganize IMS segments whose fields have variable start positions into a format in which the fields have fixed start positions DPROP does not directly support fields with variable start positions.
- Clean up data stored in an unusual way, or reorganize it before propagation to DB2.
- Suppress the propagation of certain data changes. This subject is discussed in more detail in “Selective Suppression of Data Propagation” on page 44.
- Support propagation of IMS segments containing internal segments (mapping case 3).

  The IMS-to-DPROP mapping logic of your Segment exit routine will typically perform one or more of the following functions:

  – Artificially construct, in the internal segments, ID fields that uniquely identify each occurrence of the internal segment

– Artificially construct, in the containing IMS segment, a counter field that counts the number of occurrences of an internal segment type within the containing segment (for internal segments whose number of occurrences varies).

The DPROP-to-IMS mapping logic of your Segment exit routine must assemble the IMS segment as it is expected by your IMS applications. The assembly is performed from the containing segment and from multiple internal segments.

Your Segment exit routine does not need to distinguish between propagated and nonpropagated fields; it always receives a complete segment.

You can use a Segment exit routine to change the format, position, or content of fields in a segment before it is propagated to DB2 or stored in the IMS database. ***Do not*** change the format, position, or content of the segment's key, concatenated key, or any field that maps to the primary DB2 key. Changing these fields results in an error.

If you need to convert field formats that DPROP does not directly support, consider using a Field exit routine instead of (or in combination with) Segment exit routines. Field exit routines are described in Chapter 3, "Field Exit Routines" on page 110.

If you are using a Segment exit routine, the definitions of the field format and position that you provide to DPROP apply to the DPROP segment format. DPROP does not require definitions for the IMS database format of the segment.

All your exit routines can be written in Assembler, or in COBOL, PL/I, or C. DPROP support for exit routines written in high-level languages requires LE/370 Version 1 Release 2. For synchronous propagation, the RUP and HUP call your exits in both IMS batch and online dependent regions accessing DB2. For LOG-ASYNC propagation, the RUP calls your exit routines in an MVS batch environment. During user asynchronous propagation, depending on your implementation, the RUP calls your exit routines in IMS batch and dependent regions accessing DB2, or in a non-IMS DB2/TSO or CAF environment. The RUP and HUP also call your exits during execution of the CCU and DLU.

The DataRefresher term for segment exits is *data exits*. If you are using DataRefresher to extract the IMS data, DataRefresher calls your exit routines during extraction so that the mapping performed during extraction and data propagation is the same.

As shown in Figure 1 on page 2, your Segment exit routine is called by DPROP in the following contexts:

1. During HR propagation, the RUP first calls your Segment exit routine for IMS-to-DPROP mapping, immediately after the segment has been passed by the DL/I Data Capture.

   Your Segment exit routine must convert the segment from its IMS database format (as it is in the IMS database) to the DPROP format that you specified during PR definition.

   After calling your Segment and Field exit routines, the RUP converts the field formats that you specified in your PR definition to the format of the DB2 columns and issues SQL statements (INSERT, UPDATE, or DELETE) to update the DB2 table.

2. During RH propagation, the HUP first converts the format of the DB2 columns into the field format that you specified in the PR definition. Then it calls your optional Field exit routines.

   The HUP then calls your Segment exit routine for DPROP-to-IMS mapping, just before performing the update of the IMS database. The Segment exit routine must convert the segment from its DPROP format to its IMS database format.

   Your exit routine does not need to distinguish between propagated and nonpropagated fields; it always receives a complete segment from the HUP.

   The HUP uses the following logic to provide a complete segment when only a subset of the fields are propagated: The HUP retrieves the existing IMS segment (if it exists) from the IMS database and calls your Segment exit routine to perform IMS-to-DPROP mapping of the existing IMS segment. If the IMS segment does not exist, the HUP initializes a segment in its DPROP format by setting nonpropagated fields to the default value associated with their data type (for example, zero or blank) or to binary zeroes (for space in the segment that was not explicitly defined to DPROP as fields). The HUP then merges the updated DB2 data with nonpropagated fields of the existing or initialized IMS segment; this results in a complete segment in its DPROP format.

Providing Required Mapping Logic in Segment Exits

The mapping logic provided by a Segment exit routine is usually straightforward, especially if the Segment exit routine does not support IMS segments containing internal segments that are propagated by mapping case 3 PRs. In this case, the Segment exit routine must convert the segment between its IMS database format (as it is in the IMS database) and its DPROP format (as defined to DPROP during PR definition).

Mapping Logic for IMS Segments With No Internal Segments

This section describes the mapping logic for IMS segments that do not contain internal segments.

IMS-to-DPROP Mapping

For IMS-to-DPROP mapping, when your exit routine is entered, a buffer contains the segment in its IMS format. Your exit routine must convert the segment to its DPROP format, and place it in another buffer.

DPROP-to-IMS Mapping

For DPROP-to-IMS mapping, when your segment routine is entered, a buffer contains the segment in its DPROP format, as it was mapped (according to the PR definition) by DPROP from the changed DB2 row. Your exit routine must convert the segment to its IMS format, and place it in another buffer.

The segment, in its DPROP format, is built by DPROP before calling your exit routine, as follows:

- For propagated fields, the value of the DB2 column is converted to the IMS field's DPROP format.

  If Field exit routines were defined, the Field exit routines are called to convert the fields from their DPROP format to their user format.

- Nonpropagated IMS fields are initialized in the DPROP segment format as follows:
  - For replace and delete operations, nonpropagated fields are initialized with their current value.
  - For insert operations, nonpropagated fields are initialized to the default value associated with their data type (for example, zero or blank) or to binary zeroes (for space in the segment that was not explicitly defined to DPROP as fields).

Mapping Logic for IMS Segments

This section discusses mapping logic for IMS segments with internal segments.

When designing a Segment exit routine for IMS segments containing one or more internal segment types, consider the following:

1. Each internal segment type is propagated by a mapping case 3 PR to/from a different table.

   Fields of the IMS segment that are not located in any internal segment can be propagated by a mapping case 1 or 2 PR to/from another table, if performing DB2-to-IMS propagation, propagation of these other fields is required.

   Specify use of the same Segment exit routine when you define all these PRs. This is to avoid propagation failures resulting from inconsistent mapping. Consequently, your segment exit routine will typically be called during the processing of both mapping case 3 and mapping case 1 or 2 PRs.
2. The Segment exit routine is specified at the level of the IMS segment, not at the level of the internal segment. The output of the Segment exit routine is therefore an entire IMS segment, not an individual internal segment.
3. A Segment exit routine is required for the propagation of an IMS segment containing internal segments with TYPE=E PRs. It is optional for TYPE=L and TYPE=F PRs.
4. During IMS-to-DB2 propagation your segment exit routine will be called for IMS-to-DPROP mapping.

   During DB2-to-IMS propagation, your Segment exit routine is called primarily for DPROP-to-IMS mapping, and, in some circumstances, for IMS-to-DPROP mapping.
5. When called for IMS-to-DPROP mapping, your Segment exit routine always gets as input the entire IMS segment in its IMS format. Your Segment exit routine must then return the entire IMS segment in its DPROP format.
6. When called for DPROP-to-IMS mapping, your Segment exit routine must distinguish between two cases:
   1. Sometimes, your exit routine is called during the processing of a mapping case 3 PR propagating a table change to an occurrence of an internal segment. In this case your Segment exit routine gets both of the following as input:
      - The internal segment, in its DPROP format, as mapped by DPROP from the changed DB2 row
      - The entire IMS segment, in its IMS format as it exists in the database before propagation

Your Segment exit routine must then return the modified IMS segment, in its IMS format. This is done by merging the changed internal segment occurrence in the pre-existing IMS segment.

b. Other times, your exit routine is called during the processing of the mapping case 1 or 2 PR propagating a table change to the containing IMS segment. In this case your Segment exit routine gets both of the following as input:

   - The IMS segment, in its DPROP format, as mapped by DPROP from the changed DB2 row
   - The entire IMS segment (if it exists), in its IMS format as it exists in the database before the propagation

   Your Segment exit routine must then return the modified IMS segment in its IMS format.

IMS-to-DPROP Mapping

For IMS-to-DPROP mapping, when your Segment exit routine is entered, a buffer contains the segment in its IMS format. Your exit routine must convert the segment to its DPROP format and place it in another buffer.

Make sure that the IMS-to-DPROP mapping logic of your exit routine creates a DPROP segment format that matches the PR definition. Be sure that:

- For each internal segment type defined as having a variable number of occurrences, the containing segment in its DPROP format has a count field. If such a count field does not exist in the IMS format, your exit routine must construct the count field in the DPROP format.
- Each internal segment type contains one or more ID fields that uniquely identify each occurrence of the internal segment type within its containing segment. If the ID fields do not exist in the IMS format, your exit routine must construct the ID fields in the DPROP format.
- The start position of the first occurrence of an internal segment type and the length of each internal segment occurrence exactly match the PR definitions.

DPROP-to-IMS Mapping

Sometimes, your Segment exit routine is called for the processing of a mapping case 3 PR propagating a table change to an internal segment. Other times, your Segment exit routine is called for the processing of the mapping case 1 or mapping case 2 PR propagating a table change to the containing IMS segment. Your Segment exit routine must provide logic for both types of calls (as explained in the information on page 36, your exit routine can distinguish between the two types of calls by testing the value provided by DPROP in the DAXSEGT field).

Mapping logic when propagating to an internal segment: This section discusses mapping logic when propagating to an internal segment with a mapping case 3 PR.

A mapping case 3 PR propagates a table change to an internal segment. When propagating this table change to IMS, DPROP provides the following information when entering your exit routine:

- The internal segment, in its DPROP format, as DPROP mapped it (according to the mapping case 3 PR definition) from the changed DB2 row. See below for a description of how DPROP builds it.
- The before-change IMS segment, in its IMS format, as it exists in the IMS database before propagation.

Your exit routine must merge (insert, delete, or replace) the changed internal segment occurrence into the existing IMS segment, and construct the changed IMS segment in its IMS format.

Before calling your exit routine, DPROP builds the internal segment that is provided as input to your exit routine. DPROP builds the segment as follows:

- For propagated fields, the value of the DB2 column is converted to the DPROP format of the IMS field.

  If Field exit routines were defined, the Field exit routines are called to convert the fields from their DPROP format to their user format.
- Nonpropagated IMS fields of the changed internal segment occurrence are initialized in the DPROP segment format as follows:
  - For replace and delete operations, they are initialized with their current value.
  - For insert operations, they are initialized with the default value associated with their data type (for example, zero or blank); or with binary zeroes (for space in the segment that was not explicitly defined to DPROP as fields).

Mapping logic when propagating to containing segment: This section discusses mapping logic when propagating to the containing segment with a mapping case 1 or mapping case 2 PR.

A mapping case 1 or mapping case 2 PR propagates a table change to a containing IMS segment. When called to propagate this table change to IMS, your exit routine receives the following information from DPROP upon entry:

- The containing IMS segment, in its DPROP format, as mapped by DPROP from the changed DB2 row according to the mapping case 1 or 2 PR definition. See below for a description of how DPROP builds it.
- The before-change IMS segment, in its IMS format, as it exists in the IMS database before propagation (it is provided only if the DB2 change is a replace or delete).

Your exit routine must return the new or changed IMS segment, in its IMS format, in another area.

The containing IMS segment, in its DPROP format, provided as input to your exit routine, is built as follows by DPROP before calling your exit routine:

- For fields that are propagated by the mapping case 1 or 2 PR (but not for fields propagated by mapping case 3 PRs), the value of the DB2 column is converted to the DPROP format of the IMS field.

  If Field exit routines were defined, the Field exit routines are called to convert the fields from their DPROP format to their user format.
- Fields that are not propagated by the mapping case 1 or 2 PR are initialized in the DPROP segment format as follows:

- For replace and delete operations, they are initialized with their current value.
- For insert operations, they are initialized with binary zeroes for fields located in internal segments and for space in the segment that was not explicitly defined to DPROP as fields. Other fields that are not propagated by the mapping case 1 or 2 PR are initialized with the default value associated with their data type (for example, zero or blank).

How To Write A Segment Exit Routine

This section describes some guidelines and requirements for writing a Segment exit routine to be used with DPROP. If DataRefresher uses your exit routine for data extraction, it must also conform to these requirements.

As mentioned above, your exit routine can be written in Assembler, COBOL, PL/I, or C when LE/370 Version 1 Release 2 is installed. When the RUP and HUP call your Segment exit routine, they pass the following four parameters to the exit:

- An Interface Control Block
- An IMS DB segment buffer
- A DPROP segment buffer
- A 64-byte anchor area

Note: When calling your exit routine for DPROP-to-IMS mapping, DPROP provides to your exit routine one additional segment buffer. This additional buffer contains the before-change and existing IMS segment in its IMS format. This additional buffer is not provided as a call parameter; instead, the buffer is pointed to by the DAXIDDSB field of the interface control block.

If your exit routine is written in Assembler, register 1 contains the address of the list of parameter addresses. This list is four fullwords long and contains the addresses of the parameters in the order listed above. If your exit routine is written in a high-level language supported by LE/370 Version 1 Release 2, then it must include the appropriate mapping definitions to access the four parameters being passed to it.

Interface Control Block

Figure 7 on page 28 shows the structure of the interface control block, EKYRCDAX, that is passed to your Segment exit routine. There is one interface control block per exit routine, lasting the duration of the exit in virtual storage. The following table lists:

- The fields most useful to your exit routine
- What the fields are used for
- Their displacement into the control block DSECT

Figure 6 (Page 1 of 2). Interface Control Block Parameters for Segment Exits

Field	Used For	Displacement
DAXCALL	Call function	X'20'
DAXDBNM	Name of IMS database currently in use	X'9C'
DAXSEGM	Name of physical segment type	X'4C'

Figure 6 (Page 2 of 2). Interface Control Block Parameters for Segment Exits

Field	Used For	Displacement
DAXDLEN	Length of the IMS DB segment buffer	X'7C'
DAXFLEN	Length of DPROP segment buffer	X'80'
DAXPROGM	Name of calling program	X'90'
DAXTRANS	A description of the DB or TM environment	X'8C'
DAXKFBAD	Address of fully concatenated key	X'74'
DAXKFBLN	Length of fully concatenated key	X'78'
DAXDPRCT	Type of update	X'174'
DAXISEGM	Name of segment to be processed	X'178'
DAXIDDSB	Pointer to buffer with before-change IMS DB segment	X'180'
DAXRETC	Return code that your exit provides	X'1A4'
DAXSMESG	Exit message text	X'1A8'
DAXSCRT1	Exit work space (128 bytes)	X'220'
DAXENTRD	Indicates the exit routine has been entered	X'1A2'
DAXINCTL	Indicates the exit routine has control	X'1A3'

The interface control block has the same structure as the control block DataRefresher passes to its data exits. A more complete description of these fields is included in the copy of the control block DSECT shown in Figure 7 on page 28.

Of the fields listed above, the following can be changed by your Segment exit routine:

- DAXRETC
- DAXSMESG
- DAXSCRT1
- DAXENTRD
- DAXINCTL

Altering any of the other fields in the control block causes an error.

IMS DB Segment Buffer

The IMS DB Segment Buffer contains the segment in its IMS format.

- When performing IMS-to-DPROP mapping, DPROP or DataRefresher provides the IMS segment to your Segment exit routine in this buffer. Until now, there has been no processing of the segment, so it appears as it does in the IMS database.

Your Segment exit routine must not modify this buffer when it is called to perform IMS-to-DPROP mapping.

- When performing DPROP-to-IMS mapping, your Segment exit routine must provide the segment to DPROP in this buffer. The segment must be provided in its IMS format. This is the same IMS format as provided by DPROP to your exit routine when performing IMS-to-DPROP mapping.

  When called for DPROP-to-IMS mapping, this buffer is empty at entry to the Segment exit routine.

See "Buffers and Variable-Length Segments" for notes about variable length segment.

DPROP Segment Buffer

The DPROP Segment buffer contains the segment in the DPROP-supported format that you identified during your PR definition.

- ***When performing IMS-to-DPROP mapping***, your exit must place the transformed segment into this buffer before returning to DPROP. When your exit routine returns, DPROP reads this buffer to get the transformed segment.

  You can return the segment in either fixed-length or variable-length format, depending on what you specified in your PR. This does not depend on whether the segment was fixed or variable in the IMS DB segment buffer. See "Buffers and Variable-Length Segments" for notes on variable-length segments.

- ***When performing DPROP-to-IMS mapping***, the HUP provides in this buffer the segment to your Segment exit routine. This segment is in the DPROP-supported format that you specified during the PR definition; it is in either fixed-length or variable-length format.

  1. For an IMS segment containing imbedded structures, the segment is either the containing IMS segment or one of the internal segments whose name can be found in the DAXISEGM field of the interface control block. Refer to "DPROP-to-IMS Mapping" on page 21 for more details.

  2. For all other segment types, this buffer contains the complete IMS segment; the name in the DAXISEGM field is the same as the name of the physical segment type in DAXSEGM.

  This buffer always contains an entire internal or IMS segment that has:

  – Propagated fields mapped from the changed DB2 table row

  – Nonpropagated fields either mapped from an existing IMS segment image or set to their initial values

Buffers and Variable-Length Segments

- For variable-length IMS segments, the first two bytes contain the length field, followed by the segment data. The number in the length field includes the length of the segment data plus the two bytes of the length field itself. For example, if the segment data is 18 bytes long, the length field is set at 20 bytes, or `X'14'`.

  To understand how DPROP sets the length of IMS segments during DB2-to-IMS propagation, refer to the appropriate *Administrators Guide* for your propagation mode.

- For variable-length internal segments, the length of the internal segment is not necessarily in the first two bytes. Instead, the length of a variable-length internal segment is provided by the HUP to your Segment exit routine in the DAXFLEN field. The HUP determines the length of the internal segment based on the PR definition. Remember that during PR definition, you specify the length of a variable-length internal segment on the NEXT=fieldname+n keyword of the SEGMENT statement of the DXTPCB (for PRs defined with DataRefresher), or the fieldname+n value of the NEXT column of the DPRISEG table (for PRs defined without DataRefresher).

Before-Change IMS DB Segment Buffer

The Before-Change IMS DB segment buffer exists only when the Segment exit routine is called to perform DPROP-to-IMS mapping and there is an existing segment in the IMS database. This is the case for:

- All segment types when the IMS update to be performed is a DLET or REPL
- Internal segment types for all types of updates

The Before-Change IMS DB segment buffer contains the IMS DB segment (in its IMS format) as it currently exists in the IMS database, before propagation of the DB2 change. This buffer is pointed to by the DAXIDDSB field of the interface control block.

Your Segment exit routine must not modify this buffer.

Although your exit always receives this buffer when there is an existing IMS segment image, it is only important when performing DPROP-to-IMS mapping of an IMS segment containing internal segments. In the other cases, it is recommended that your exit routine ignore this buffer.

See “Buffers and Variable-Length Segments” on page 25 for notes on variable-length segments.

64-Byte Anchor Area

DPROP gives you 64 bytes as a general-purpose storage area. Each exit routine has its own unique anchor area. You can use it for whatever you want. Initially, the area is set to all binary zeros, and DPROP (or DataRefresher if you are using it) never changes it again.

The anchor area exists in virtual storage, and remains yours for the duration of the exit, as follows:

- For IMS batch and BMP regions, the anchor area lasts for the duration of the application program.
- For MPP regions, the anchor area lasts for the duration of the IMS Program Controller Subtask. This can span multiple MPP executions.
- For CCU and DLU executions, the anchor area lasts for the duration of the job step.
- For LOG-ASYNC propagation and user asynchronous propagation, the anchor area lasts for the duration of the MVS task being used by the receiver program to call the RUP.

Interface Control Block DSECT

You can generate the following DSECT in your assembler exit routine by coding the EKYRCDAX macro statement. For HLL exit routines, you can include or copy one of the following members to map the Segment exit routine Interface Control Block:

EKYRCDXC For exit routines written in COBOL
EKYRCDXP For exit routines written in PL/I
EKYRCDXK For exit routines written in C

Figure 7 on page 28 shows the interface control block, followed by detailed descriptions of its fields.

```
          EKYRCDAX
+****************** START OF CONTROL BLOCK SPECIFICATION **************/
+*                                                                    */
+*       CONTROL BLOCK NAME:                                          */
+*         EKYRCDAX (DAX)                                             */
+*                                                                    */
+*       DESCRIPTIVE NAME:                                            */
+*         DPROP SEGMENT EXIT INTERFACE BLOCK                         */
+*                                                                    */
+*                                                                    */
+*********************************************************************
+*                                                                    *
+*         THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
+*                                                                    *
+*         5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
+*         ALL RIGHTS RESERVED.                                       *
+*                                                                    *
+*         U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
+*         USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
+*         GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
+*                                                                    *
+*         LICENSED MATERIALS - PROPERTY OF IBM.                      *
+*                                                                    *
+*********************************************************************
+*                                                                    */
+*       STATUS: V1 R2 M0                                             */
+*                                                                    */
+*       FUNCTION:                                                    */
+*         THIS IS THE CONTROL BLOCK USED TO INTERFACE BETWEEN        */
+*              - DPROP OR DXT                                        */
+*              AND                                                   */
+*              - A USER'S SEGMENT EXIT ROUTINE (THESE USER           */
+*                EXIT ROUTINES ARE CALLED BY DXT 'USER DATA          */
+*                EXIT ROUTINES')                                     */
+*                                                                    */
+*         THERE IS ONE DAX CONTROL BLOCK FOR EACH SEGMENT            */
+*         EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT         */
+*         IN VIRTUAL STORAGE.                                        */
+*         FOR SYNCH PROPAGATION IN MPP REGIONS:                      */
+*         - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER       */
+*           SUBTASK.                                                 */
+*         FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR            */
+*         CCU AND DLU PROCESSING, AND FOR ASYNCH PROPAGATION         */
+*         (DEPENDING ON HOW AYSNCH PROPAGATION IS IMPLEMENTED):      */
+*         - THIS IS THE DURATION OF THE JOBSTEP.                     */
+*                                                                    */
+*-------------------------------------------------------------------*/
+*       IMPORTANT NOTES:                                             */
+*       ================                                             */
+*       - SINCE THE SAME USER EXIT ROUTINE CAN BE INVOKED BOTH       */
+*         BY DPROP AND BY DXT: CHANGES TO THIS CONTROL BLOCK MUST    */
+*         BE COORDINATED BETWEEN DPROP DEVELOPMENT AND DXT           */
+*         DEVELOPMENT.                                               */
+*                                                                    */
+*       - FIELDS MARKED IN THE COMMENT WITH '***DXT ONLY***'         */
+*         HAVE NO MEANING, WHEN THE SEGMENT USER EXIT                */
+*         ROUTINE IS INVOKED BY DPROP.                               */
+*-------------------------------------------------------------------*/
+*                                                                    */
+*       MODULE TYPE= MACRO                                           */
+*          PROCESSOR= ASSEMBLER H                                    */
+*                                                                    */
+*       INNER CONTROL BLOCKS: NONE                                   */
+*                                                                    */
+*       MACROS USED FROM MACRO LIBRARY: NONE                         */
+*                                                                    */
```

Figure 7 (Part 1 of 6). Interface Control Block for a Segment Exit Routine

```
                                 67+*       CHANGE ACTIVITY:                                                 */
                                 68+*                      KMP0057  12/13/90                                 */
                                 69+*                      KMP0060  02/08/91 COPYRIGHT INFORMATION           */
                                 70+*                      KMPREL2  03/20/91                                 */
                                 71+*                                                                        */
                                 72+******************** END OF CONTROL BLOCK SPECIFICATION **************/

000000                           74+DAX      DSECT
                        00000    75+DVRDAX   EQU   *                    LABEL FOR DXT COMPATIBILITY
                                 76+*-------------------------------------------------------------------*
                                 77+*   THIS SECTION OF THE CB MAY NOT BE MODIFIED BY EXIT               *
                                 78+*-------------------------------------------------------------------*
000000                           79+DAXPFX   DS    0CL32                PREFIX OF CONTROL BLOCK
000000                           80+DAXTNAME DS    CL8                  EYE CATCHER: "DVRXCDAX"
000008                           81+DAXRSVD  DS    CL24                 RESERVED FOR DXT INTERNAL USE
                                 82+*
000020                           83+DAXPFXE  DS    0CL448               PREFIX EXTENSION
000020                           84+DAXCALL  DS    CL2                  TYPE OF CALL TO EXIT:
                                 85+*                                     =C'NO' - NORMAL CALL,
                                 86+*                                        ISSUED TO CONVERT DATA FROM
                                 87+*                                        'IMS DATABASE FORMAT' TO
                                 88+*                                        'DPROP/DXT' FORMAT
                                 89+*                                     =C'RV' - REVERSE CALL
                                 90+*                                        ISSUED TO CONVERT
                                 91+*                                        DATA FROM:
                                 92+*                                        'DPROP/DXT' FORMAT
                                 93+*                                        TO
                                 94+*                                        'IMS DATABASE FORMAT'
                                 95+*                 ***DXT ONLY***      =C'RE' - RETURN CALL, ISSUED
                                 96+*                                        INSTEAD OF NEXT REQUEST FOR
                                 97+*                                        NEW DATA AT REQUEST OF EXIT
                                 98+*                                        (SEE DAXRETC VALUE 4)
                                 99+*                 ***DXT ONLY***      =C'ED' - END-OF-DATA CALL
                                100+*                                        ISSUED BY DXT.
                                101+*
000022                          102+DAXDATYP DS    CL2                  TYPE OF DATA BEING PASSED--
                                103+*                                     =C'DL' - DL/I DATA
                                104+*                 ***DXT ONLY***      =C'PS' - PHYSICAL SEQUENTIAL
                                105+*                 ***DXT ONLY***      =C'VK' - VSAM KSDS DATA
                                106+*                 ***DXT ONLY***      =C'VE' - VSAM ESDS DATA
                                107+*                 ***DXT ONLY***      =C'GD' - GDI RECRD DATA
                                108+*
000024                          109+DAXFIL   DS    CL32                 NAME OF FILE OR PCB FROM WHICH
                                110+*                                   DATA IS BEING PASSED
                                111+*
000044                          112+DAXPSB   DS    CL8                  NAME OF PSB IF TYPE IS "DL"
                                113+*
00004C                          114+DAXSEGM  DS    CL32                 NAME OF SEGMENT IF TYPE IS "DL"
                                115+*                                   IF CALLER IS DPROP:
                                116+*                                      - NAME OF PHYSICAL SEGM.
                                117+*                                   IF CALLER IS DXT:
                                118+*                                      - NAME OF SEGM. SPECIFIED
                                119+*                                        IN THE USED DBD (DBD CAN
                                120+*                                        BE A PHYSICAL OR LOGICAL
                                121+*                                        DBD)
                                122+*
00006C                          123+DAXPCBAD DS    AL4    ***DXT ONLY***  PTR TO PCB IF TYPE IS "DL"
                                124+*
000070                          125+DAXPCBLS DS    AL4    ***DXT ONLY***  PTR TO LIST OF DEM'S PCBS,
                                126+*                                   IF DEM IS A DL/I DEM
```

Figure 7 (Part 2 of 6). Interface Control Block for a Segment Exit Routine

```
                            127+*
000074                      128+DAXKFBAD DS    AL4                  PTR TO SEGMENT'S FULLY
                            129+*                                   CONCAT KEY (IF DL/I).
                            130+*                                   IF CALLER IS DPROP:
                            131+*                                      - 0, IF 'NOKEY' HAS BEEN
                            132+*                                        SPECIFIED ON EXIT=
                            133+*                                        OF DBDGEN.
                            134+*
000078                      135+DAXKFBLN DS    F                    LENGTH OF SEGM'S FULLY
                            136+*                                   CONCAT KEY (IF DL/I)
                            137+*                                   IF DPROP: 0, IF 'NOKEY' HAS BEEN
                            138+*                                             SPECIFIED ON EXIT=
                            139+*                                             OF DBDGEN.
                            140+*
00007C                      141+DAXINLN  DS    0F
00007C                      142+DAXDLEN  DS    F                    LENGTH OF IMS DB SEGMENT BUFFER
                            143+*
000080                      144+DAXOUTLN DS    0F
000080                      145+DAXFLEN  DS    F                    LENGTH OF DPROP SEGMENT BUFFER
                            146+*
000084                      147+DAXSYSPR DS    AL4   ***DXT ONLY*** POINTER TO SYSPRINT DCB (EXIT
                            148+*                                   MAY WISH TO RECORD INFORMATION
                            149+*                                   IN SYSPRINT VIA "PUT"--
                            150+*                                   DCB FACTS: LRECL=121,
                            151+*                                      NO CARRIAGE CONTROL CHAR
                            152+*
000088                      153+DAXENVT  DS    0CL12                ENVIRONMENT SUBFIELDS
000088                      154+DAXOPSYS DS    CL4                  OPERATING SYSTEM:
                            155+*                                    =C'ESA ' IF MVS/ESA
                            156+*                ***DXT ONLY***      =C'XA  ' IF MVS/XA
                            157+*                ***DXT ONLY***      =C'MVS ' IF MVS
                            158+*
00008C                      159+DAXTRANS DS    CL4                  DB/DC ENVIRONMENT:
                            160+*                                    =C'BAT ' IF IMS BATCH/BMP
                            161+*                                    =C'MPP ' IF IMS MPP
                            162+*                                    =C'IFP ' IF FAST PATH
                            163+*                                    =C'CICS' IF CICS
                            164+*                                    =C'    ' IF NONE OF ABOV.
                            165+*
000090                      166+DAXPROGM DS    CL4                  CALLING PROGRAM:
                            167+*                                    =C'DXT ' IF DataRefresher
                            168+*                                    =C'DPRS' IF DPROP SYNCH PROP
                            169+*                                    =C'DPRA' IF DPROP ASYNCH PROP
                            170+*                                    =C'DPRC' IF DPROP CCU  PROP
                            171+*                                    =C'DPRL' IF DPROP DLU
                            172+*
000094                      173+DAXEXIT  DS    CL8                  NAME OF THIS EXIT ROUTINE
                            174+*
00009C                      175+DAXDBNM  DS    CL8                  NAME OF IMS DATABASE
                            176+*                                   IF CALLER IS DPROP:
                            177+*                                    - NAME OF PHYSICAL DBD.
                            178+*                                   IF CALLER IS DXT:
                            179+*                                    - NAME OF USED DBD (CAN BE
                            180+*                                      NAME OF A PHYSICAL OR
                            181+*                                      LOGICAL DBD)
                            182+*
0000A4                      183+DAXDPRPN DS    CL24                 RESERVED
                            184+*
0000BC                      185+DAXASGNO DS    F     ***DXT ONLY*** NUMBER OF DAXASEGS ARRAY
                            186+*                                   ELEMENTS CONTAINING
                            187+*                                   ANCESTOR SEGM INFORMATION
```

Figure 7 (Part 3 of 6). Interface Control Block for a Segment Exit Routine

```
                                     188+*
0000C0                               189+DAXASEGS DS    15CL12 ***DXT ONLY***  ARRAY OF ANCESTOR SEGMS,
                                     190+*                                     ONLY FOR DL/I SEGM EXIT,
                                     191+*                                     IN ORDER FROM ROOT TO
                                     192+*                                     PARENT SEGMENT (EACH
                                     193+*                                     ARRAY ELEMENT IS MAPPED
                                     194+*                                     BY DAXANCTR DSECT, BELOW)
                                     195+*
000174                               196+DAXRSVD1 DS    CL46                   RESERVED FOR DXT USE
0001A2                        00174  197+         ORG   DAXRSVD1               REDEFINE THIS AREA
                                     198+*
000174                               199+DAXDPRCT DS    CL4' ' --DPROP ONLY-- IF CALLER IS DPROP:
                                     200+*                                     - EXIT IS CALLED TO PROCESS:
                                     201+*                                       'ISRT': A DL/I OR DB2 INSERT
                                     202+*                                       'DLET': A DL/I OR DB2 DELETE
                                     203+*                                       'REPL': A DL/I OR DB2 REPLACE
                                     204+*                                               (AFTER-REPLACE IMAGE)
                                     205+*                                    IF CALLER IS DXT:
                                     206+*                                     - NOT USED
000178                               207+DAXREPL  DS    C' '   --DPROP ONLY-- IF CALLER IS DPROP AND IF
                                     208+*                                    DAXDPRCT IS 'REPL':
                              000C1  209+DAXREPLA EQU   C'A'                     'A': AFTER-REPLACE  IMAGE
                              000C2  210+DAXREPLB EQU   C'B'                     'B': BEFORE-REPLACE IMAGE

000179                               212+DAXSEGT  DS    C' '   --DPROP ONLY-- IF CALLER IS DPROP:
                                     213+*                                     - TYPE OF SEGMENT PROCESSED:
                              000E4  214+DAXSEGTU EQU   C'U'                     'U': UPDATED IMS SEGMENT
                              000C1  215+DAXSEGTA EQU   C'A'                     'A': ANCESTOR OF UPDATED SEGM
                              000C9  216+DAXSEGTI EQU   C'I'                     'I': INTERNAL SEGMENT

00017A                               218+DAXPSUP  DS    C' '   --DPROP ONLY-- IF CALLER IS DPROP, DESCRIPTION
                                     219+*                                    WHETHER PROPAGATION-SUPPRESSION
                                     220+*                                    IS ALLOWED:
                              000D5  221+DAXPSUPN EQU   C'N'                     'N': SUPPRESSION NOT ALLOWED
                              000E8  222+DAXPSUPY EQU   C'Y'                     'Y': SUPPRESSION ALLOWED

00017B                               224+         DS    C' '                  RESERVED
                                     225+*
00017C                               226+DAXISEGM DS    CL8' ' --DPROP ONLY-- IF CALLER IS DPROP:
                                     227+*                                     - FOR RH PROPAGATION
                                     228+*                                       NAME OF SEGMENT TO
                                     229+*                                       PROCESS. SAME AS PHYS.
                                     230+*                                       IMS SEGNAME IN DAXSEGM
                                     231+*                                       IF NOT MAPPING CASE 3
                                     232+*                                       ENTITY (INTERNAL)
                                     233+*                                       SEGMENT IN PROCESS.
                                     234+*                                    IF CALLER IS DXT:
                                     235+*                                     - NOT USED
000184                               236+DAXIDDSB DS    A      --DPROP ONLY-- IF CALLER IS DPROP:
                                     237+*                                     - FOR RH PROPAGATION
                                     238+*                                       POINTER TO THE BUFFER
                                     239+*                                       CONTAINING THE 'BEFORE-CHANGE'
                                     240+*                                       IMS DATABASE SEGMENT.
                                     241+*                                       BUFFER CONTAINS THE
                                     242+*                                       BEFORE IMAGE OF THE
                                     243+*                                       IMS SEGMENT IF:
                                     244+*                                       - DAXDPRCT EQ REPL, OR
                                     245+*                                       - DAXDPRCT EQ DLET, OR
                                     246+*                                       - DAXSEGT EQ DAXSEGTI
                                     247+*                                         (INTERNAL SEGMENT OF
                                     248+*                                         MAPPING CASE 3)
                                     249+*                                       OR CONTAINS ALL BINARY
                                     250+*                                       ZEROES IN OTHER CASES.
                                     251+*                                       BUFFER IS READ ONLY
                                     252+*                                       FOR THE EXIT ROUTINE.
```

Figure 7 (Part 4 of 6). Interface Control Block for a Segment Exit Routine

```
000188                    253+DAXIDDSL DS    A      --DPROP ONLY-- IF CALLER IS DPROP:
                          254+*                                     - FOR RH PROPAGATION
                          255+*                                       LENGTH OF THE 'BEFORE-CHANGE'
                          256+*                                       IMS DB SEGMENT POINTED-TO
                          257+*                                       BY DAXIDDSB.
00018C           001A2    258+         ORG
                          259+*                                    POINT TO THE END OF DAXRSVD1
                          260+*-------------------------------------------------------------------*
                          261+*   THE NEXT GROUP OF FIELDS MAY BE MODIFIED BY THE EXIT ROUTINE     *
                          262+*-------------------------------------------------------------------*
0001A2                    263+DAXENTRD DS    CL1                    SET BY EXIT ROUTINE TO
                          264+*                                     C'X', INDICATES
                          265+*                                     THAT EXIT HAS BEEN ENTERED
                          266+*
0001A3                    267+DAXINCTL DS    CL1                    SET BY EXIT ROUTINE TO
                          268+*                                     C'X', INDICATES
                          269+*                                     THAT EXIT IS IN CONTROL
                          270+*
0001A4                    271+DAXRETC  DS    F                      RETURN CODE--
                          272+*                                       VALUE SET HERE BY EXIT,
                          273+*
                          274+*                                     RETURN CODE VALUES...
                 00000    275+DAXRCOK  EQU   0                        = 0 - NORMAL, OUTPUT
                          276+*                                             DATA RETURNED
                          277+*
                 00004    278+DAXRCOKR EQU   4      ***DXT ONLY***    = 4 - NORMAL, OUTPUT
                          279+*                                             DATA RETURNED,
                          280+*                                             DXT SHOULD
                          281+*                                             RETURN TO EXIT FOR NEXT
                          282+*                                             OCCURRENCE OF THIS RECORD
                          283+*                                             OR SEGMENT
                          284+*
                 00008    285+DAXRCNQ  EQU   8                        = 8 - IF CALLER IS DPROP:
                          286+*                                             DPROP WILL SUPPRESS
                          287+*                                             THE PROPAGATION OF
                          288+*                                             THE CHANGED DL/I DATA
                          289+*                                           - IF CALLER IS DXT:
                          290+*                                             DXT SHOULD NOT
                          291+*                                             CONSIDER DATA TO
                          292+*                                             BE ELIGIBLE FOR
                          293+*                                             EXTRACT
                          294+*
                 0000C    295+DAXRCERB EQU   12                       =12   ERROR
                          296+*                                           - IF CALLER IS DPROP:
                          297+*                                             PROPAGATION FAILURE.
                          298+*                                             DPROP/RUP WILL
                          299+*                                             GO THROUGH ITS USUAL
                          300+*                                             ERROR HANDLING LOGIC.
                          301+*                                           - IF CALLER IS DXT:
                          302+*                                             DXT SHOULD
                          303+*                                             TERMINATE BATCH
                          304+*
                 00010    305+DAXRCERD EQU   16                       =16   ERROR
                          306+*                                           - IF CALLER IS DPROP:
                          307+*                                             RUP WILL ABEND
                          308+*                                           - IF CALLER IS DXT:
                          309+*                                             DXT SHOULD
                          310+*                                             TERMINATE DEM EXECUTION
                          311+*
```

Figure 7 (Part 5 of 6). Interface Control Block for a Segment Exit Routine

```
0001A8                         312+DAXSMESG DS    CL64                 TEXT OF MESSAGE PASSED
                               313+*                                   FROM EXIT ROUTINE TO DPROP/DXT.
                               314+*                                   ALL BLANKS MEANS NO MESSAGE.
                               315+*                                 - IF CALLER IS DPROP:
                               316+*                                   MSG WILL BE WRITTEN TO
                               317+*                                   VARIOUS DESTINATIONS ACCORDING
                               318+*                                   TO USUAL DPROP/RUP ERROR HANDLING
                               319+*                                   LOGIC IN MESSAGE EKYR980I OR
                               320+*                                   EKYR981E.
                               321+*                                 - IF CALLER IS DXT:
                               322+*                                   TEXT OF MESSAGE WILL BE
                               323+*                                   WRITTEN TO
                               324+*                                   SYSPRINT DATA SET IN MESSAGE
                               325+*                                   DVRA0_50.
                               326+*                                   (UNDERSCORE IS REPLACED
                               327+*                                   BY ONE OF SEVERAL DIGITS)
                               328+*                                   HAS EFFECT FOR ALL CALLS.
                               329+*
0001E8                         330+DAXDPRPM DS    CL24                 STORAGE RESERVED FOR DATA EXIT
                               331+*
000200                         332+DAXRSVD2 DS    CL32                 RESERVED FOR DXT USE
000220                         333+DAXSCRT1 DS    CL128                WORK SPACE (SCRATCHPAD)
                               334+*                                   MAY BE USED BY EXIT
                               335+*                                   ROUTINE AS DESIRED
                               336+*
                       002A0   337+DAXEND   EQU   *                    END OF DAX DSECT
                       002A0   338+DAXLEN   EQU   *-DAX                LENGTH OF DAX DSECT
                               339+*********************************************************************
                               340+*
                               341+*    DAXANCTR DSECT   ***DXT ONLY***
                               342+*                     MAPS THE ARRAY ELEMENTS OF DAXASEGS
                               343+*
                               344+*********************************************************************
000000                         345+DAXANCTR DSECT ,      ***DXT ONLY***
000000                         346+DAXASGNM DS    CL8    ***DXT ONLY***   ANCESTOR SEGM NAME
                               347+*
000008                         348+DAXASGAD DS    AL4    ***DXT ONLY***   ANCESTOR SEGM ADDRESS
                               349+*
                               350          END
```

Figure 7 (Part 6 of 6). Interface Control Block for a Segment Exit Routine

Interface Control Block Field Descriptions

The following list contains detailed descriptions of the fields in the interface control block. The primary descriptions given are for DPROP unless otherwise indicated. Additional descriptions are given for DataRefresher.

Some of the fields are not useful to your exit routine when DPROP calls it. These fields are for DataRefresher only, both in the interface control block and below.

DAXTNAME Contains the constant **DVRXCDAX**, used to identify the control block in a storage dump.

DAXCALL The call function that describes what action your exit routine must perform. This field can have the following values:

NO Normal (IMS-to-DPROP mapping). The exit routine is called to convert the segment from its IMS to its DPROP (or DataRefresher) format described in the PR definition. **NO** calls can be generated by both the RUP and HUP during propagation, and by DataRefresher DEM during extract.

RV Reverse (DPROP-to-IMS mapping). The exit routine is called to convert the segment from the DPROP (or DataRefresher) format described in the PR definition to the IMS format. **RV** calls are generated only by the HUP.

ED End of data (*DataRefresher only*). The exit routine is called to perform end of data summary functions. A DataRefresher user must request this function with a EODCALL=Y keyword on the SEGMENT statement of the DXTPSB.

RE Return (*DataRefresher only*). The exit routine is called during data extract after returning with a return code of 4. DPROP does not support return code 4 and return calls. Your exit must not use return code 4 with DataRefresher if the extracted data is propagated using a generalized mapping case. This results in different mapping for the DataRefresher extract and propagation by DPROP, which causes inconsistencies in the propagated data.

DAXDATYP Contains the constant **DL**, indicating that the data being mapped is an IMS segment.

DAXFIL If called by DPROP, the name of the DBPCB used for the updating IMS call. This field is filled for both NO and RV calls. If called by DataRefresher, it contains the name of the DBPCB used for extract.

DAXPSB If called by DPROP, the name of the PSB used for the program that modified the IMS data. This field is only specified for NO calls. If called by DataRefresher, it contains the name of the PSB used for DataRefresher DEM.

DAXDBNM If called by DPROP, the name of the physical IMS DBD.

If called by DataRefresher, the name of the DBD referenced in the PCB of the PSB used for the DataRefresher DEM. In this case, the DBD can be either physical or logical. For DataRefresher users, if the exit routine is called for segments propagated by DPROP, it is recommended that you refer to *physical* DBDs in the PCB.

DAXSEGM If called by DPROP, the name of the IMS segment type as specified in the physical IMS DBD.

If called by DataRefresher, it is the segment type found on the SEGMENT statement of the DXTPSB. For DataRefresher users, it is recommended that you specify the same segment names on the SEGMENT statements as in the physical IMS DBD.

DAXKFBAD The address of the segment's fully concatenated key. Remember that your exit routine must not modify this key. The address can be zero if the segment has no fully concatenated key, or if the key was not supplied to the RUP (for example, if the NOKEY option was used in the EXIT= keyword of the DBD).

DAXKFBLN The length of the fully concatenated key. The length can be zero if the segment has no fully concatenated key, or if the key was not supplied to the RUP.

DAXDLEN If called by DPROP for a NO call (IMS-to-DPROP mapping), the length of the IMS DB segment.

If called by DPROP for an RV call (DPROP-to-IMS mapping), this field contains the length of the IMS DB segment buffer. For RV calls, this buffer contains the result of Segment exit routine processing; do not store a segment in the IMS DB segment buffer that is longer than the length specified in DAXDLEN. This can cause storage overlays and unpredictable results. If the segment in its IMS format is a variable-length segment, the segment exit must store the actual length of the segment in the first two bytes of the IMS segment buffer.

If called by DataRefresher, this field contains the length specified in the BYTES= keyword of the SEGMENT statement in the CREATE DXTPSB control statement. For variable length IMS DB segments, the actual length is found in the first two bytes of the buffer.

DAXFLEN If called by DPROP for a NO call (IMS-to-DPROP mapping), the length of the DPROP segment buffer. For NO calls, this buffer contains the result from Segment exit routine processing; do not store a segment in the DPROP segment buffer that is longer than the length specified in DAXFLEN. This can cause storage overlays and unpredictable results. If the PR defines the segment in its DPROP format as a variable-length segment, the actual length of the segment must be stored by the segment exit in the first two bytes of the DPROP segment buffer.

If called by DPROP for an RV call (DPROP-to-IMS mapping), DAXFLEN contains the length of the segment in its DPROP format.

DAXOPSYS Contains the constant **ESA**, indicating that the program is running in an MVS environment.

DAXTRANS Contains a value describing the environment in which the exit routine is called. This field can have the following values:

BAT IMS Batch or BMP environment
MPP IMS MPP environment
IFP IMS Fast Path environment
CICS CICS environment

If the exit is called in an environment other than those listed above, the value consists of blanks.

DAXPROGM Contains information about the calling program, either DPROP or DataRefresher. This field can have the following values:

DPRS Called by DPROP during synchronous propagation
DPRA Called by DPROP during LOG-ASYNC propagation and user asynchronous propagation
DPRC Called by DPROP during CCU execution
DPRL Called by DPROP during DLU execution
DataRefresher Called by DataRefresher

DAXEXIT The load module name of the Segment exit routine.

DAXDPRCT Contains a value describing the type of IMS or DB2 update performed. This field can contain the values **ISRT**, **REPL**, and **DLET** for the insert, replace, or delete of an IMS segment or DB2 row, respectively. The field is set only when DPROP calls the exit routine. If the exit is called for CCU or DLU processing, the value of the field is set to ISRT because the DPROP logic simulates an insert during CCU and DLU processing.

DAXREPL This field is only set when the exit routine is called by DPROP during processing of a Replace. The field specifies whether the exit routine is being called to process the after-image (**A**) of the segment, or the before-image (**B**) of the segment.

DAXSEGT This field describes which type of segment is being processed.

For NO calls (IMS-to-DPROP mapping), the possible values are:

U The segment being processed is the IMS segment being updated.

A The segment being processed is a physical ancestor of the IMS segment being updated. The value can be set to **A** when processing a PR propagating path-data located in an ancestor segment.

For RV calls (DPROP-to-IMS mapping), the possible values are:

U The segment located in the DPROP segment buffer is the containing IMS segment.

I The segment located in the DPROP segment buffer is an internal segment. The value can be set to **I** when processing mapping case 3 PRs. The name of the internal segment type being processed is located in DAXISEGM.

The field is set only when the exit routine is called by DPROP.

DAXPSUP This field indicates whether the Segment exit routine can request suppression of data propagation during its current call. This field is set only when the exit routine is called by DPROP.

If your Segment exit routine is designed to support propagation suppression, it must test this field to determine if it can suppress propagation.

N The exit routine cannot request suppression of data propagation.

Y The exit routine can request suppression of data propagation.

This field is set to Y only if:

- The PR definition specified PROPSUP=Y.
- Other conditions are met (for example, if the current call of the exit is not for the before-image of a segment).

DAXISEGM This field is only set for RV calls (DPROP-to-IMS mapping) and contains the name of the segment to be processed. For internal segments (mapping case 3), this is the name of an internal segment; for any other case, it contains the name of the physical IMS segment and is the same as in the DAXSEGM field above.

DAXIDDSB This field is only set for RV calls (DPROP-to-IMS mapping) and contains a pointer to a buffer, or zero. The buffer contains the before-change IMS DB segment (in its IMS format). The size of the before-change IMS DB segment is provided in DAXIDDSL.

The buffer must not be modified by your exit routine.

This pointer is only present if the type of update in DAXDPRCT is either REPL or DLET, or if the segment to be processed is a mapping case 3 internal segment.

The buffer pointed to by DAXIDDSB is only important when performing DPROP-to-IMS mapping of an IMS segment containing internal segments. In this case, your exit routine requires the following two inputs:

- The DPROP segment buffer. It contains either an internal segment (if it is the target table of an internal segment that has changed) or the IMS segment (in all other cases) in its DPROP format.
- The buffer pointed to by DAXIDDSB. It contains the before-change copy of the IMS segment, as stored in the IMS DB.

Your Segment exit routine must then use this input to assemble the after-change copy of the IMS segment (in its IMS format). The assembled IMS segment must be returned in the IMS DB segment Buffer.

If the IMS segment is variable length, the first two bytes contain the length field, followed by the segment data. The number in the length field includes the length of the segment data plus the two bytes of the length field itself.

DAXIDDSL The length of the segment in the before-change IMS DB segment buffer.

The next two fields are switches that can be useful for problem determination. DPROP and DataRefresher do not require your exit routine to set these fields. However, they can help you determine where a problem occurred if you have an ABEND. DPROP and DataRefresher set these fields to blanks before calling your exit routine for the first time.

DAXENTRD Exit-entered flag.

As you enter your exit routine, set this field to **X**. DPROP does not change this field again, so if a problem occurs, you can determine if your exit has been entered.

DAXINCTL Exit-in-control flag.

You can also set this field to **X**, indicating that your exit routine has control. When DPROP regains control, it resets this field to blank, so you can determine if your exit routine has control when an ABEND occurs.

The next two fields can be used along with the RUP's and HUP's error handling logic. For more information on return codes and error handling techniques, see “Return Codes and Error Handling Techniques” on page 39.

DAXRETC The return code that the exit routine provides when returning to its caller. This field is set to zero when the exit routine is called.

DAXSMESG User-provided error message. It is set to blanks when the exit routine is called. When the exit routine returns, if the field is not blank, DPROP or DataRefresher writes the contents of the field.

DPROP prefaces the message with the number EKYR980I or EKYR981E, and writes the message according to its usual error handling logic (for example, to the OS/VS console, trace data set, //EKYPRINT, or the Audit trail). DataRefresher prefaces the message with the number DVRA_50, (where _ is one of several possible digits) and writes the message to the //SYSPRINT data set.

DAXSCRT1 An exit routine work space for your own use; for example, to save information across calls to the exit routine. Before the first call to your exit routine, DPROP initializes this space to binary zeros, and does not modify it again.

Exit Routine Processing

Using the information given above, your Segment exit routine can copy the propagated data from a buffer, transform it, and return it using another buffer.

When called for IMS-to-DPROP mapping (with NO in DAXCALL), your Segment exit routine can read the segment from the IMS DB segment buffer and return it in the DPROP segment buffer after transformation.

When called for DPROP-to-IMS mapping (with RV in DAXCALL), your Segment exit routine reads the segment from the DPROP segment buffer and returns it in the IMS DB segment buffer after transformation.

There are, however, some restrictions and guidelines to follow when developing your exit routine:

- When DPROP calls it, your exit routine always gets control in AMODE 31, and must return control in AMODE 31. Keywords DPROP passes to your exit are usually located above the 16MB line. The exit routine is loaded above or below the 16MB line, depending on the RMODE attribute of the exit load module.

  It is recommended that you code and link-edit your program as reentrant. To simplify programming, DPROP provides work spaces in your exit routines, in the interface control block, and the 64-byte anchor area.

- If your exit routine is written in Assembler language, DPROP uses standard OS/VS conventions when calling your exit routine.

  - Register 1 points to the parameter list described above.
  - Register 13 contains the address of a register save area.
  - Register 14 contains the return address.
  - Register 15 contains the entry point address of the exit routine.

  Upon entry, the exit routine must save the register contents into the save area that the caller provides. If your exit routine calls other routines that use standard MVS linkage conventions, it must also provide a save area of its own. The exit routine must return to its caller using normal OS/VS conventions after restoring the registers. A return code must be provided in the interface control block, not in register 15.

- Your Segment exit routine must never change the content or the displacement of the key field of the propagated IMS segment. Do not change the fully concatenated key, the address of which is in DAXKFBAD in the interface control block. and when called for DPROP-to-IMS mapping, your exit routine must **not** change the DPROP segment buffer,

- When called for IMS-to-DPROP mapping, your exit routine must **not** change the IMS DB segment buffer, which contains a copy of the propagated segment.
- If you map an IMS field that is not in the fully concatenated key to a column of the DB2 primary key, observe the following rules:

  For a TYPE=E PR, your exit routine must not change the content or displacement of this field.

  If the DPROP format of the segment is variable length, this field must be contained in the DPROP format of the segment that your exit routine returns to DPROP during IMS-to-DPROP mapping.
- Because the exit routine for synchronous propagation runs in the same environment as the propagating application program, it can generate the same type of IMS calls and SQL statements as the application program. However, for LOG-ASYNC propagation and user asynchronous propagation using the TSO-Attach or CAF-Attach, the exit routines do not execute in an IMS environment, and cannot generate IMS calls. Therefore, it may be preferable to generate only SQL statements.

  If your exit generates IMS calls, then use the AIB interface described in *IMS/ESA Application Programming: DL/I Calls*, which allows your exit routine to generate calls without the address of the IMS PCBs.

  During synchronous propagation, IMS and DB2 update calls, made from within your exit routine, are not propagated synchronously (but can be propagated asynchronously, if you implement LOG-ASYNC propagation or user asynchronous propagation).

  Exclude the PCBs your exit routine uses from the list passed to the application program upon entry. You can avoid changing the application program if you need to add PCBs that your exit routine uses exclusively. Refer to *IMS/ESA Utilities Reference: System* for more details.
- A Segment exit routine must not perform functions that are not supported by the environment in which it is running. For example, an exit routine running in an MPP region must not WRITE to OS files; also, the exit routine must not generate STIMER macros in an IMS environment.

  For performance reasons, your exit routine should generate static rather than dynamic SQL statements. Avoid using functions that have a detrimental impact on the performance of the propagating program (such as performing an OPEN and CLOSE on an OS/VS file each time the exit routine is called).

Return Codes and Error Handling Techniques

This section discusses how to return from your exit routine to DPROP, including return codes and error handling techniques.

Return Codes

The following list describes the return codes that you can set when returning from your Segment exit routine to the RUP or HUP. To set the return code, place it in the DAXRETC field in the interface control block. The RUP and HUP read this field when they regain control.

Returning with any code other than those on the list is considered an error and results in an ABEND, regardless of the error option (that is, even with ERROPT=IGNORE in effect).

0 Used for normal returns.

4 This return code is not supported by DPROP. Returning this code to the RUP or HUP causes it to ABEND. While DataRefresher supports this return code, your exit must not return this code to DataRefresher while processing a segment that is propagated by the DPROP generalized mapping logic. This can result in DataRefresher mapping during the extract that is not consistent with the DPROP mapping during propagation. This can result in propagation failures.

8 This return code causes DPROP to suppress propagation of the changed data segment. The exit must be specifically allowed to use this code during PR generation. The exit routine must not return with return code 8 when DAXPSUP was set to N (for example, the Segment exit routine is processing the before-image of a segment, or processing an ancestor of the changed IMS segment). For more information about suppressing data propagation, see "Selective Suppression of Data Propagation" on page 44.

If your exit routine uses this return code with DataRefresher, the current occurrence of the segment is not extracted.

12 DPROP interprets this return code as a failure indication. This prevents propagation of the changed data, and DPROP proceeds with its error logic.

If ERROPT=BACKOUT is in effect, for synchronous propagation, the RUP or HUP backs out the propagating application. If ERROPT=BACKOUT is in effect for LOG-ASYNC propagation, the Receiver terminates with an error message. For user asynchronous propagation, CCU or DLU execution, the RUP and HUP return to the caller with an error. DPROP uses its error reporting logic to write diagnosis information.

If ERROPT=IGNORE is in effect, the RUP and HUP do not perform propagation, and return to the caller without performing a backout and without providing any error indication to the caller. However, if this occurs during CCU or DLU execution, the RUP and HUP return to the CCU or DLU with an error. DPROP uses its error reporting logic to write diagnosis information.

If the exit returns to DataRefresher with this return code, DataRefresher terminates the extract requests currently being processed.

16 Return code 16 signals a severe error. DPROP does not propagate the changed data, but generates an ABEND. Returning this code to DataRefresher causes it to terminate the DEM.

Error Handling Techniques

When your exit routine encounters an error, It is strongly recommended that your exit routine take advantage of the standard error handling logic of DPROP. In the interface control block, you can supply a return code in DAXRETC, and an error message in DAXSMESG. You must not return an error message in DAXSMESG without providing an error return code (12 or 16), because this can create many console messages.

By supplying DPROP with an error return code and message, you gain many advantages. When an exit returns with an error return code, the RUP and HUP trace or snap the data and the control blocks involved in the interface. The exits are included in the standardized error handling scheme of DPROP. This scheme:

- Determines the difference between ERROPT=BACKOUT and ERROPT=IGNORE

- Is different for propagation and CCU or DLU execution
- Protects against sending too many messages to the MVS consoles

DPROP writes your error message using its standard message writing logic: WTO, trace data set (the IMS log, the //EKYLOG data set, or the //EKYTRACE data set), and AUDIT trail.

If the exit routine generates its own messages or ABENDs, the RUP and HUP cannot include the exit routine in their standardized error handling, or guard against sending numerous messages to the MVS consoles. Therefore, it is not recommended that your exit routine generate its own messages or ABENDs when an error occurs.

Saving Information Across Calls

You can save information across calls to the exit routine. Save the information either in the 64-byte anchor area or in the DAXSCRT1 field of the interface control block. If these areas are not large enough, you can generate a GETMAIN and save the address of the storage in either of these areas.

DPROP and DataRefresher treat the interface control block in slightly different ways. In DPROP, there is one interface control block per exit routine (lasting for the duration of the MVS task), while in DataRefresher, there is one interface control block per segment type (lasting for the duration of the extract request). If DataRefresher and DPROP call the exit routine, and are sensitive to this difference, you can use the anchor area to save information across calls. DPROP and DataRefresher handle this area the same way: there is one anchor area per exit, and it lasts for the duration of the exit in virtual storage.

Updating Your Segment Exit Routine

DPROP does not provide any online change logic to replace an existing load module copy of your segment exit routine with a new version of the load module. If you need to change your exit routine, then stop the affected IMS regions, DPROP asynchronous Receiver or any user asynchronous receiver programs before performing the change. A change of the exit routine without stopping the IMS regions or receiver programs can cause unpredictable results. For example, some MPP regions can use the new version of the exit routine, while other regions use the old version. After the change, you can restart the IMS regions.

Tracing Your Exit Routine

DPROP provides a trace facility that can assist you in detecting errors in your exit routines. You can activate the DPROP trace facility by providing a TRACE control statement in the //EKYIN data set of the job step where your exit routine runs. For synchronous propagation, you can also activate tracing by calling the SCU with a TRACE ON control statement.

If you include debug level 2 on the TRACE or TRACE ON statements, the trace output includes the changed IMS segment and the propagating SQL statements for HR propagation, or the changed DB2 row and the propagating DL/I call, including the IMS segment data, for RH propagation.

If you include debug level 4 on the TRACE or TRACE ON statements, each time the exit routine returns to DPROP, the trace output includes:

- The contents of the interface control block
- The IMS DB segment buffer
- The DPROP segment buffer
- The 64-Byte anchor area
- For DPROP-to-IMS mapping, the old image of the IMS segment (located in the buffer pointed to by DAXIDDSB)

This information is automatically included in the RUP or HUP trace information when a propagation failure occurs, even if you have not activated the DPROP trace.

If you include debug level 8 on the TRACE or TRACE ON statements, the trace output includes a record of each call to, and each return from, an exit routine.

Two other debugging aids, located in the interface control block, are:

- The *exit-entered* flag
- The *exit-in-control* flag

In a dump, these flags help you determine if your exit routine is in control at the time of a failure.

Differences Between Exit Routine Calls From DPROP or DataRefresher

This section summarizes the differences between calling your Segment exit routine from DPROP and calling it from DataRefresher.

- DPROP does not call the exit routine with ED or RE calls.
- DPROP does not support return code 4. If DataRefresher/DEM calls the exit routine for a segment that is propagated using generalized mapping logic, the exit routine must not return a return code 4 to DataRefresher/DEM.
- When DPROP calls the exit routine, there is one interface control block *per exit routine*, lasting for the duration of the IMS Program Controller MVS Subtask. When DataRefresher calls it, there is one interface control block *per segment type*, lasting for the duration of the extract request.

  If your exit routine must save information across calls and is sensitive to this difference, you can use the 64-byte anchor area to save information. This area is treated the same by both DPROP and DataRefresher.
- The following fields in the interface control block are not set when DPROP calls the exit routine; a brief explanation of any consequences this has is included.

  DAXPCBAD The exit routine cannot access the DB PCB used for the updating IMS calls.

  DAXPCBLS The exit routine cannot access the list of DB PCBs. If the exit routine needs to issue IMS calls, it must use the IMS AIB interface.

  DAXSYSPR The exit routine cannot write to the SYSPRINT file.

  DAXASGNO This is the number of array elements in DAXASEGS.

  DAXASEGS This is the array of names of ancestor segments.
- The RV call is generated *only* by DPROP for DPROP-to-IMS transformation and is not used by DataRefresher.

- DAXDPRCT is only set when DPROP calls the exit routine.
- DAXISEGM is only set when DPROP calls the exit routine for DPROP-to-IMS mapping.
- DPROP support for exit routines written in high-level languages requires LE/370 Version 1 Release 2.

Refer to the appropriate DataRefresher or DXT documentation for information about DataRefresher trace facilities.

Telling DPROP About Your Segment Exit Routine

This section discusses how to inform DPROP that you want to use a Segment exit routine. The procedure depends on how you are entering your PR.

PRs Entered Through DataRefresher UIM

If you are entering the PR through DataRefresher UIM, you must provide the following keyword operands on the SEGMENT statement of the DXTPSB:

- Specify the load module name of the exit routine on the **EXIT=** keyword.
- Specify the fixed or maximum length of the segment, in its DPROP format, on the **XBYTES=** keyword.
- Specify whether the DPROP segment format is fixed with **FORMAT=F**, or variable with **FORMAT=V**.

You can also specify that your exit routine be allowed to suppress propagation of an update by returning a return code of 8. You specify this by coding a PROPSUP=Y value on the MVGUPARM keyword of the DataRefresher SUBMIT control statement. Specifying PROPSUP=N prohibits your exit routine from returning a return code of 8.

PRs Entered Into the MVG Input Tables

If you are entering your PR information directly into the Mapping Verification and Generation (MVG) input tables, without using DataRefresher, you use the DPRISEG (or SEG) table to inform DPROP about your exit routine. The SEG table is one of the MVG input tables. There are three columns in the table that you must specify:

SEGEXIT The name of your Segment exit routine. It can be up to eight characters long. It must be alphanumeric, and begin with an alphabetic character.

SEGEXITL The length, in bytes, of the segment in its DPROP format. The length must be specified as an integer. If the segment length is variable, use the maximum length.

SEGEXITF The format of the segment in its DPROP format. If the segment is fixed length, place an **F** in this column. If the segment is variable length, place a **V** in this column.

SEGEXITL and SEGEXITF describe the segment in the DPROP segment buffer and in the DPROP-supported format. This format is for IMS-to-DPROP mapping the output of the exit routine, and for DPROP-to-IMS mapping the input to your exit routine.

You must specify values for all three of these columns to use your exit routine. If either the segment length or the format is entered, the MVGU checks to make sure you have also entered the name of the exit routine.

To specify that your exit routine be allowed to suppress propagation of an update by returning a return code of 8, use the PROPSUP column of the DPRIPR MVG input table. Place a **Y** in the column to allow suppression. Place an **N** in the column to prohibit suppression.

Selective Suppression of Data Propagation

Your Segment exit routine can selectively suppress data propagation. Propagation is suppressed when your exit routine returns a return code of 8 to DPROP. This means that your exit can analyze the changed data segment and, based on your requirements, tell DPROP whether or not to propagate the change to your DB2 table or IMS database. For example, this can be used to suppress the propagation of IMS delete calls, turning your propagated copy into a kind of archive that contains data for longer periods than the source data. If you use a return code of 8, DPROP does not propagate the data, but continues with its normal processing. This section describes how to set up selective suppression.

Your Segment exit routine must not return with return code 8 when DAXPSUP was set to N (for example, because the Segment exit routine is processing the before-image of a segment, or processing an ancestor of the changed IMS segment).

To indicate that you want to allow a return code of 8 to be used, you must specify the PROPSUP parameter as PROPSUP=Y. The default for this parameter is PROPSUP=N, which means that a return code of 8 is not allowed.

If you are using DataRefresher to code your PRs, specify this PROPSUP parameter in the MAPUPARM operand of the DataRefresher SUBMIT statement. If you are using the MVG input tables to code your PRs, the PROPSUP parameter is specified in the MVGIPR Table.

When you specify PROPSUP=Y, it is recorded in the mapping table. This can be useful for problem determination. If the database administrator (DBA) finds an inconsistency between IMS and DB2 data, the DBA can check the mapping table to see if it is caused by a return code of 8 from a Segment exit routine.

Be *very* careful when using selective suppression. The inconsistencies that it creates can result in future propagation failures. For example, an SQL INSERT can fail if the original DELETE statement was not propagated to DB2. Also, selective suppression can make the CCU useless, because the IMS and DB2 data are no longer consistent.

You can retain some of the usefulness of the CCU with the USE keyword in the CCU CHECK statement. If you are suppressing delete calls before they are propagated to your DB2 table, you can create a view of the DB2 table that excludes the undeleted rows during the CCU read phase. For more information, see *IMS DPROP Reference*.

For HR propagation, you can also selectively suppress propagation through definition of a WHERE clause during PR definition. If you can choose between specifying a WHERE clause and suppressing with a Segment exit routine, choose the WHERE clause approach. Using the WHERE clause does not cause inconsistencies, and does not restrict the usefulness of the CCU.

Mapping case 2 propagates multiple segment types to or from one table. Suppression of propagation of the entity segment does not automatically suppress propagation of the extension segments (RH propagation of the delete of the entity segment is an exception; this also suppresses deletion of the extension segments). Therefore, if you provide a Segment exit routine that suppresses the propagation of the entity segment, you must also provide Segment exit routines that suppress the propagation of the extension segments. This is important for avoiding propagation failures.

First Sample Segment Exit Routine

Figure 8 on page 46 is an example of a Segment exit routine. The Segment exit routine transforms a segment between its IMS format and its DPROP format. The IMS format contains fields with variable start positions. In the DPROP format, all of the fields have fixed start positions.

In this example, the first two fields in the IMS format of the segment are fixed length and contain the segment key. The last three fields, however, are variable length, containing a last name, first name, and city. It is assumed that each of the variable length fields has a maximum length, and a variable start position within the segment in its IMS format.

When it receives a changed IMS data segment and is called for IMS-to-DPROP mapping, the exit routine transforms it into a DPROP-supported format, and returns the segment to the RUP for propagation to DB2.

When it is called for DPROP-to-IMS mapping, the exit routine transforms the DPROP format into the IMS format and returns the segment to the HUP for propagation to IMS.

The source code shown in Figure 8 on page 46 is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYESE1A. Following the source code are definitions related to the sample Segment exit routine.

```
********** START OF SPECIFICATIONS *************************************
*    MODULE NAME = EKYESE1A                                            *
*                                                                      *
*    DESCRIPTIVE NAME = SAMPLE 'SEGMENT USER EXIT ROUTINE'             *
*                                                                      *
*    STATUS: V1 R2 M0                                                  *
*                                                                      *
*    FUNCTION = EKYESE1A IS A SAMPLE DPROP                             *
*               'SEGMENT USER EXIT ROUTINE' AND ILLUSTRATES            *
*               THE TRANSFORMATION OF A SEGMENT LAYOUT BETWEEN ITS:    *
*                    - 'DL/I DB FORMAT'                                *
*                    - 'DPROP FORMAT'.                                 *
*                                                                      *
*               EKYESE1A ILLUSTRATES ONE OF THE MOST TYPICAL USAGE     *
*               OF DPROP SEGMENT USER EXITS: THE TRANSFORMATION OF:    *
*                 - A VARIABLE LENGTH DL/I SEGMENT WITH FIELDS         *
*                   HAVING VARIABLE START POSITIONS                    *
*                 INTO                                                 *
*                 - A SEGMENT LAYOUT WHERE ALL FIELDS HAVE A FIXED     *
*                   START POSITION.                                    *
*               (DL/I SEGMENTS WITH FIELDS HAVING VARIABLE START       *
*               POSITIONS CAN BE SUPPORTED BY THE 'GENERALIZED         *
*               MAPPING LOGIC' OF DPROP V1R2, ONLY IF A SEGMENT        *
*               USER EXIT ROUTINE TRANSFORMS THE SEGMENT INTO A        *
*               FORMAT WHERE ALL FIELDS HAVE A FIXED START POSITION.   *
*               IF DB2 TO IMS OR TWO WAY PROPAGATION IS IN EFFECT,     *
*               THEN THE SEGMENT USER EXIT ROUTINE MUST ALSO BE ABLE   *
*               TO TRANSFORM SUCH A SEGMENT FROM A FORMAT WHERE ALL    *
*               FIELDS HAVE FIXED START POSITION (THE DPROP FORMAT)    *
*               TO A FORMAT WITH VARIABLE START POSITIONS (THE DL/I    *
*               SEGMENT FORMAT))                                       *
*                                                                      *
*               THIS SAMPLE ASSUMES THAT IN ITS DL/I DB FORMAT:        *
*                 1) THE FIRST PORTION OF THE SEGMENT HAS A            *
*                    FIXED FORMAT CONTAINING THE KEY OF THE            *
*                    SEGMENT.                                          *
*                                                                      *
*                 2) THE SECOND PORTION OF THE SEGMENT CONSISTS OF     *
*                    THREE ADJACENT PAIR OF:                           *
*                          (LENGTH FIELD,VARIABLE LENGTH FIELD)        *
*                    FOR THE FAMILY-NAME, FIRST-NAME, AND CITY.        *
*                                                                      *
*                    WITH THE EXCEPTION OF THE FIRST PAIR OF           *
*                    LENGTH FIELD AND VARIABLE LENGTH FIELD:           *
*                    THESE PAIR OF LENGTH FIELDS AND VARIABLE LENGTH   *
*                    FIELDS HAVE A VARIABLE START POSITION.            *
*                                                                      *
*                    EACH VARIABLE LENGTH FIELD IS ASSUMED TO HAVE A   *
*                    SPECIFIC MAXIMUM LENGTH.                          *
*                                                                      *
*              THE FIGURE BELOW PROVIDES AN OVERVIEW OF                *
*              THE TRANSFORMATION PERFORMED BY THIS SAMPLE EXIT.       *
*                                                                      *
*              THE LEFT-HAND SIDE DESCRIBES THE SEGMENT SEG1 IN ITS    *
*              DL/I DB FORMAT. THE FIGURE PROVIDES FOR EACH FIELD      *
*              LOCATED IN THE SEGMENT:                                 *
*                - THE FIELD NAME                                      *
*                - THE FORMAT OF THE FIELD                             *
*                  'H' STANDS FOR 'HALFWORD BINARY' FORMAT.            *
*                  'C' STANDS FOR 'FIXED LENGTH CHARACTER FORMAT'      *
*                  'VC' STANDS FOR 'VARIABLE LENGTH CHARACTER FORMAT'  *
*                - THE FIXED START POSITION OF THE FIELD (IF THE       *
*                  FIELD HAS A FIXED START POSITION) OR 'V' IF         *
*                  THE FIELD HAS A VARIABLE START POSITION.            *
```

Figure 8 (Part 1 of 23). First Sample Segment Exit Routine (Assembler)

```
*                                                                    *
*            THE RIGHT-HAND SIDE DESCRIBES THE SEGMENT SEG1 IN ITS  *
*            DPROP FORMAT. THE FIGURE PROVIDES FOR EACH FIELD       *
*            LOCATED IN THE SEGMENT:                                *
*              - THE FIELD NAME                                     *
*              - THE FORMAT OF THE FIELD                            *
*                'C' STANDS FOR 'FIXED LENGTH CHARACTER FORMAT'     *
*                'VC' STANDS FOR 'VARIABLE LENGTH CHARACTER FORMAT' *
*              - THE FIXED START POSITION OF THE FIELD WITHIN THE   *
*                DPROP FORMAT OF THE SEGMENT.                       *
*                                                                    *
*            *--------------------*    *---------------------*      *
*            ‘ SEGMENT IN ITS     ‘    ‘  SEGMENT IN ITS     ‘      *
*            ‘ VARIABLE-LENGTH    ‘    ‘  FIXED-LENGTH       ‘      *
*            ‘ DL/I DB FORMAT     ‘    ‘  DPROP FORMAT       ‘      *
*            *--------------------*    *---------------------*      *
*                                                                    *
*            *--------*----*-----*    *--------*-----*------*      *
*            ‘FLD NAME‘ FLD ‘ FLD ‘    ‘FLD NAME‘ FLD ‘ FLD  ‘      *
*            ‘        ‘ FMT ‘START‘    ‘        ‘ FMT ‘START ‘      *
*            *--------*----*-----*    *--------*-----*------*      *
*            ‘SEG1LL  ‘  H  ‘  1  ‘    ‘SEG1LL  ‘  H  ‘  1   ‘      *
*            ‘KEYFLD1 ‘  C  ‘  3  ‘<-->‘KEYFLD1 ‘  C  ‘  3   ‘      *
*            ‘KEYFLD2 ‘  C  ‘  5  ‘<-->‘KEYFLD2 ‘  C  ‘  5   ‘      *
*            ‘        ‘     ‘     ‘    ‘        ‘     ‘      ‘      *
*            ‘FAMILY_L‘  H  ‘ 11  ‘<-->‘FAMILY_L‘  H  ‘  11  ‘      *
*            ‘FAMILY  ‘  VC ‘ 13  ‘<-->‘FAMILY  ‘  VC ‘  13  ‘      *
*            ‘        ‘     ‘     ‘    ‘        ‘     ‘      ‘      *
*            ‘FIRST_L ‘  H  ‘  V  ‘<-->‘FIRST_L ‘  H  ‘  43  ‘      *
*            ‘FIRST   ‘  VC ‘  V  ‘<-->‘FIRST   ‘  VC ‘  45  ‘      *
*            ‘        ‘     ‘     ‘    ‘        ‘     ‘      ‘      *
*            ‘CITY_L  ‘  H  ‘  V  ‘<-->‘CITY-L  ‘  H  ‘  65  ‘      *
*            ‘CITY    ‘  VC ‘  V  ‘<-->‘CITY    ‘  VC ‘  67  ‘      *
*            *--------*----*-----*    *--------*-----*------*      *
*                                                                    *
*            PLEASE REFER TO THE DSECTS TOWARDS THE BOTTOM OF THIS  *
*            MODULE IN ORDER TO FIND ALL THE DETAILS ABOUT THE      *
*            'DL/I DB FORMAT' AND THE 'DPROP FORMAT' OF SEG1.       *
*                                                                    *
*                                                                    *
*    NOTES =                                                         *
*            EKYESE1A IS CALLED:                                    *
*                                                                    *
*              - FOR TRANSFORMATION OF THE SEGMENT FROM ITS DL/I    *
*                DB FORMAT WITH VARIABLE FIELD START POSITIONS      *
*                TO ITS FORMAT SUPPORTED BY DXT/DPROP WITH FIXED    *
*                FIELD START POSITIONS (NORMAL CALL TYPE INDICATED  *
*                BY 'NO' IN DAXCALL FIELD OF THE DAX AREA):         *
*                  - BY DXT (DURING EXTRACT OF THE DL/I DATA).      *
*                  - BY DPROP DURING:                               *
*                    - SYNCH/ASYNCH IMS-TO-DB2 PROPAGATION          *
*                    - SYNCH DB2-TO-IMS PROPAGATION                 *
*                    - CCU EXECUTION                                *
*                    - DLU EXECUTION                                *
*                                                                    *
*              - FOR TRANSFORMATION OF THE SEGMENT FROM ITS FORMAT  *
*                SUPPORTED BY DXT/DPROP WITH FIXED FIELD START      *
*                POSITIONS TO ITS FORMAT ON THE DL/I DATABASE WITH  *
*                VARIABLE FIELD START POSITITIONS (REVERSE CALL     *
*                TYPE INDICATED BY 'RV' IN DAXCALL FIELD OF DAX):   *
*                  - BY DPROP DURING:                               *
*                    - SYNCH DB2-TO-IMS PROPAGATION                 *
*                    - CCU EXECUTION OF REPAIR FILE GENERATION      *
*                    - DLU EXECUTION                                *
*                                                                    *
*                                                                    *
```

Figure 8 (Part 2 of 23). First Sample Segment Exit Routine (Assembler)

```
*                                                                      *
*       DEPENDENCIES = NONE                                            *
*                                                                      *
*       RESTRICTIONS = NONE                                            *
*       REGISTER CONVENTIONS=                                          *
*                    R13= ADDRESS OF SAVE AREA                         *
*                    R12= MODULE BASE REGISTER                         *
*                    R8 = ADDRESS OF ANCHOR AREA                       *
*                    R7 = ADDRESS OF SEGMENT IN DPROP FORMAT           *
*                    R6 = ADDRESS OF SEGMENT IN DL/I DB FORMAT         *
*                    R5 = ADDRESS OF DAX                               *
*                    R2 = CURRENT ADDRESS WITHIN SEGMENT IN ITS        *
*                         DL/I DB FORMAT                               *
*       PATCH LABEL  = - (NONE)                                        *
*                                                                      *
*    MODULE TYPE = PROCEDURE                                           *
*       PROCESSOR = ASSEMBLER                                          *
*       MODULE SIZE = APPROXIMATELY 1400 BYTES                         *
*       ATTRIBUTES = REENTRANT                                         *
*       RMODE      = ANY                                               *
*       AMODE      = 31                                                *
*                                                                      *
*    ENTRY POINT = EKYESE1A                                            *
*       PURPOSE = SEE FUNCTION                                         *
*       LINKAGE = STANDARD OS/VS ASSEMBLER LINKAGE CONVENTIONS.        *
*                                                                      *
*    INPUT  : R1 = POINTING TO A STANDARD PARAMETER ADDRESS LIST.      *
*                  1ST PARAMETER: ADDRESS OF DAX (DAX IS THE           *
*                                 EXIT INTERFACE CONTROL BLOCK)        *
*                  2ND PARAMETER: ADDRESS OF SEGMENT IN DL/I FORMAT    *
*                  3RD PARAMETER: ADDRESS OF SEGMENT IN DPROP FORMAT   *
*                  4TH PARAMETER: ADDRESS OF ANCHOR AREA PRESERVED     *
*                                 ACROSS CALLS TO THIS EXIT.           *
*                                                                      *
*    OUTPUT : THE SEGMENT FORMAT TRANSFORMATION HAS BEEN DONE          *
*                                                                      *
*    EXIT-NORMAL=                                                      *
*       STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.                   *
*       RETURN CODES = 0                                               *
*                                                                      *
*    EXIT-ERROR=                                                       *
*       STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.                   *
*       RETURN CODE  = 12: INVALID DATA IN DL/I SEGMENT                *
*                          (INVALID LENGTH OF SEGMENT,                 *
*                           INVALID FIELD LENGTH,                      *
*                           FIELD NOT TOTALLY WITHIN SEGMENT).         *
*                    = 16: SHOULD-NOT-OCCUR ERRORS                     *
*                          (INVALID CALL FUNCTION,                     *
*                           PARAMETER AREA TOO SMALL,                  *
*                           INVALID SEGMENT NAME).                     *
*                                                                      *
*                                                                      *
*    ABEND CODE OF EKYESE1A = NONE                                     *
*       ABEND REASON CODES  = NONE                                     *
*                                                                      *
*    ERROR MESSAGES ISSUED BY EKYESE1A                                 *
*            EKYESE0E: CALL FUNCTION NOT SUPPORTED                     *
*            EKYESE1E: UNSUPPORTED DBD OR SEGNAME                      *
*            EKYESE2E: 3RD PARAMETER HAS INCORRECT LENGTH              *
*            EKYESE3E: INVALID SEGMENT LENGTH                          *
*            EKYESE4E: FAMILY FIELD DOES NOT FIT WITHIN SEGMENT        *
*            EKYESE5E: LENGTH OF FAMILY FIELD IS INVALID               *
*            EKYESE6E: FIRST-NAME FIELD DOES NOT FIT WITHIN SEGMENT    *
*            EKYESE7E: LENGTH OF FIRST-NAME FIELD IS INVALID           *
*            EKYESE8E: CITY FIELD DOES NOT FIT WITHIN SEGMENT          *
*            EKYESE9E: LENGTH OF CITY FIELD IS INVALID                 *
*                                                                      *
```

Figure 8 (Part 3 of 23). First Sample Segment Exit Routine (Assembler)

```
*                                                                  *
*    EXTERNAL REFERENCES                                           *
*                                                                  *
*       ROUTINES=      = NONE                                      *
*                                                                  *
*       DATA AREAS     = SEE CONTROL BLOCKS                        *
*                                                                  *
*       CONTROL BLOCKS = DAX   INTERFACE CB FOR SEGMENT EXIT ROUTINE  *
*                                                                  *
*    MACROS CODED IN MODULE= NONE                                  *
*                                                                  *
*    MACROS USED FROM MACRO LIBRARY=                               *
*           SAVE     - SAVE REGISTERS                              *
*           GETMAIN  - OS/VS GETMAIN                               *
*                                                                  *
*           EKYRCDAX - INTERFACE CB FOR SEGMENT EXIT ROUTINE       *
*                                                                  *
*                                                                  *
*    TABLES= NONE                                                  *
*                                                                  *
*    INCLUDE CODE FROM LIBRARY= NONE                               *
*                                                                  *
*    CHANGE ACTIVITY= NONE                                         *
*                                                                  *
************* END OF SPECIFICATIONS **********************************
************ LOGIC OF EKYESE1A **************************************
*                                                                 *
*                                                                 *
*    MAIN-LINE LOGIC:                                             *
*    ================                                             *
*                                                                 *
*    1) MODULE ENTRY LOGIC:                                       *
*    ----------------------                                       *
*          - PROVIDE REGISTER EQUATES                             *
*          - GENERATE A MODULE SAVE-ID                            *
*          - SAVE REGISTERS AND ESTABLISH MODULE BASE REGISTER    *
*          - LOAD ADDRESSES OF CALL PARAMETERS                    *
*          - SET 'MODULE ENTERED' AND 'MODULE IN CONTROL' FLAGS   *
*            INTO DAX.                                            *
*          - IF FIRST INVOCATION OF THE EXIT:                     *
*              - GETMAIN AN AREA CONTAINING AMONG OTHER           *
*                A MODULE SAVE-AREA AND MODULE-WORKSPACE.         *
*              - SAVE ADDRESS OF GETMAINED AREA.                  *
*              - CLEAR THE GETMAINED AREA.                        *
*          - CHAIN MODULE SAVE-AREA AND SAVE-AREA OF CALLER.      *
*                                                                 *
*          NOTE: SINCE THE SAMPLE EXIT DOES NOT CALL OTHER        *
*          FUNCTIONS, IT DOES NOT REALLY NEED A SAVE-AREA AND     *
*          DOES NOT REALLY NEED TO GETMAIN AN AREA.               *
*          EKYESE1A NEVERTHELESS PROVIDES THIS LOGIC,             *
*          WITH THE HOPE, THAT THIS COULD HELP IBM CUSTOMERS      *
*          FOR THE DEVELOPMENT OF MORE COMPLEX, REAL-LIFE,        *
*          SEGMENT USER EXITS.                                    *
*                                                                 *
*    2) VERIFY INFORMATION PROVIDED BY CALLER                     *
*    ----------------------------------------                     *
*          - VERIFY THAT THE EXIT IS INVOKED TO PROPAGATE THE     *
*            RIGHT DBD/SEGNAME.                                   *
*          - VERIFY THAT THE AREA CALL PARAMETER USED FOR         *
*            THE SEGMENT IN ITS DPROP FORMAT HAS THE EXPECTED LENGTH.*
*          - VERIFY THAT THE REQUESTED CALL FUNCTION IS SUPPORTED.  *
*            AND BRANCH ACCORDING TO CALL FUNCTION:               *
*            -- FOR A 'NO' CALL: PERFORM TRANSFORMATION           *
*                                DL/I DB FORMAT ---> DPROP FORMAT  *
*            -- FOR A 'RV' CALL: PERFORM TRANSFORMATION           *
*                                DPROP FORMAT   ---> DL/I DB FORMAT  *
```

Figure 8 (Part 4 of 23). First Sample Segment Exit Routine (Assembler)

```
*                                                                    *
*    3) TRANSFORM THE SEGMENT: DL/I DB FORMAT ---> DPROP FORMAT      *
*    --------------------------------------------------------        *
*       A)   PROCESS THE FIRST, FIXED-FORMAT PORTION OF THE          *
*            SEGMENT:                                                *
*                                                                    *
*              - VERIFY THAT THE DL/I FORMAT OF THE SEGMENT IS       *
*                LONG ENOUGH TO CONTAIN THE FIRST, FIXED-FORMAT      *
*                PORTION OF THE SEGMENT.                             *
*              - MOVE THE FIELDS IN THE FIRST, FIXED-FORMAT          *
*                PORTION OF THE SEGMENT INTO THE 'DPROP FORMAT' OF   *
*                THE SEGMENT.                                        *
*                                                                    *
*       B)   PROCESS THE SECOND, VARIABLE-FORMAT PORTION OF THE      *
*            SEGMENT:                                                *
*              - INITIALIZE THE 'CURRENT POINTER' WITHIN THE         *
*                DL/I FORMAT TO THE START OF THE VARIABLE-FORMAT     *
*                PORTION.                                            *
*                                                                    *
*              - FOR EACH FIELD IN THE VARIABLE-FORMAT PORTION OF    *
*                THE SEGMENT:                                        *
*                -- VALIDATE THE FIELD LENGTH:                       *
*                    ---FIELD LENGTH SHOULD BE POSITIVE              *
*                    ---FIELD LENGTH SHOULD NOT EXCEED A SPECIFIC    *
*                       MAXIMAL LENGTH.                              *
*                    ---FIELD SHOULD BE TOTALLY WITHIN THE SEGMENT.  *
*                                                                    *
*                -- COPY THE FIELD TO THE 'DPROP FORMAT' OF THE      *
*                   SEGMENT.                                         *
*                                                                    *
*    4) TRANSFORM THE SEGMENT: DPROP FORMAT ---> DL/I DB FORMAT      *
*    ----------------------------------------------------------      *
*       A)   PROCESS THE FIRST, FIXED-FORMAT PORTION OF THE          *
*            SEGMENT:                                                *
*                                                                    *
*              - VERIFY THAT THE DL/I FORMAT OF THE SEGMENT IS       *
*                LONG ENOUGH TO CONTAIN THE FIRST, FIXED-FORMAT      *
*                PORTION OF THE SEGMENT.                             *
*              - MOVE THE FIELDS IN THE FIRST, FIXED-FORMAT          *
*                PORTION OF THE SEGMENT INTO THE 'DPROP FORMAT' OF   *
*                THE SEGMENT.                                        *
*                                                                    *
*       B)   PROCESS THE SECOND, VARIABLE-FORMAT PORTION OF THE      *
*            SEGMENT:                                                *
*              - INITIALIZE THE 'CURRENT POINTER' WITHIN THE         *
*                DL/I FORMAT TO THE START OF THE VARIABLE-FORMAT     *
*                PORTION.                                            *
*                                                                    *
*              - FOR EACH FIELD IN THE VARIABLE-FORMAT PORTION OF    *
*                THE SEGMENT:                                        *
*                -- VALIDATE THE FIELD LENGTH:                       *
*                    ---FIELD LENGTH SHOULD BE POSITIVE              *
*                    ---FIELD LENGTH SHOULD NOT EXCEED A SPECIFIC    *
*                       MAXIMAL LENGTH.                              *
*                    ---FIELD SHOULD BE TOTALLY WITHIN THE SEGMENT.  *
*                                                                    *
*                -- COPY THE FIELD TO THE 'DPROP FORMAT' OF THE      *
*                   SEGMENT.                                         *
*                                                                    *
*                                                                    *
*    5) RETURN LOGIC                                                 *
*    ----------------                                                *
*          - RESTORE REGISTERS OF THE CALLER                         *
*          - RETURN TO THE CALLER.                                   *
*                                                                    *
```

Figure 8 (Part 5 of 23). First Sample Segment Exit Routine (Assembler)

```
                                        332 *                                                                       *
                                        333 *    ERROR LOGIC                                                        *
                                        334 *    ===========                                                        *
                                        335 *                                                                       *
                                        336 *          - FORMAT AN ERROR MESSAGE INTO DAX                           *
                                        337 *          - SET RETURN CODE INTO DAX                                   *
                                        338 *          - RETURN TO THE CALLER.                                      *
                                        339 *                                                                       *
                                        340 ************ END-OF-LOGIC *********************************************
                                        342 **********************************************************************
                                        343 **********************************************************************
                                        344 **********************************************************************
                                        345 ****                                                              ****
                                        346 ****          MODULE ENTRY LOGIC                                  ****
                                        347 ****                                                              ****
                                        348 **********************************************************************
                                        349 **********************************************************************
                                        350 **********************************************************************

000000                                  352 EKYESE1A START
                                        353 *
                                        354 EKYESE1A AMODE 31              EXIT EXPECTS TO BE CALLED IN AMODE-31
                                        355 EKYESE1A RMODE ANY             EXIT CAN BE LOADED ANYWHERE
                                        356 *
                                        357 *---------------------------------------------------------------------*
                                        358 *        DEFINITION OF REGISTER EQUATES                               *
                                        359 *---------------------------------------------------------------------*
                                        360 *
                                 00000  361 R0       EQU   0
                                 00001  362 R1       EQU   1
                                 00002  363 R2       EQU   2               CURRENT POSITION WITHIN DL/I DB FMT
                                 00003  364 R3       EQU   3
                                 00004  365 R4       EQU   4
                                 00005  366 R5       EQU   5               A(DAX)
                                 00006  367 R6       EQU   6               A(SEGMENT IN DL/I DB FORMAT)
                                 00007  368 R7       EQU   7               A(SEGMENT IN DPROP FORMAT)
                                 00008  369 R8       EQU   8               A(ANCHOR AREA)
                                 00009  370 R9       EQU   9
                                 0000A  371 R10      EQU   10
                                 0000B  372 R11      EQU   11              BAS REGISTER TO CALL SUBROUTINES
                                 0000C  373 R12      EQU   12              MODULE BASE REGISTER
                                 0000D  374 R13      EQU   13              A(SAVEAREA)
                                 0000E  375 R14      EQU   14
                                 0000F  376 R15      EQU   15

                                        378 *---------------------------------------------------------------------*
                                        379 *        GENERATE SAVE-ID CONSISTING OF EXIT NAME,                    *
                                        380 *        COMPILATION DATE AND COMPILATION TIME.                       *
                                        381 *---------------------------------------------------------------------*

                                        383          LCLC  &SAVEID
                                        384 &SAVEID  SETC  'EKYESE1A DPR120'.'-'.'&SYSDATE'.'-'.'&SYSTIME'

                                        386 *---------------------------------------------------------------------*
                                        387 *        SAVE REGISTERS AND ESTABLISH MODULE BASE REGISTER            *
                                        388 *---------------------------------------------------------------------*

                                        390          SAVE  (14,12),,&SAVEID SAVE REGISTERS
000000 47F0 F024                 00024  391+         B     36(0,15)                       BRANCH AROUND ID
000004 1E                               392+         DC    AL1(30)                        LENGTH OF IDENTIFIER
000005 C5D2E8C5E2C5F1C1                 393+         DC    CL8'EKYESE1A'                  IDENTIFIER
00000D 40C4D7D9F1F2F060                 394+         DC    CL8' DPR120-'                  IDENTIFIER
000015 F0F361F2F361F9F3                 395+         DC    CL8'03/23/93'                  IDENTIFIER
00001D 60F1F04BF4F6                     396+         DC    CL6'-10.46'                    IDENTIFIER
000023 00
000024 90EC D00C                 0000C  397+         STM   14,12,12(13)                   SAVE REGISTERS
```

Figure 8 (Part 6 of 23). First Sample Segment Exit Routine (Assembler)

```
000028 18CF                    399          LR    R12,R15          R12=ENTRY POINT OF THIS EXIT
                         00000  400          USING EKYESE1A,R12    ESTABLISH BASE REGISTER

                                402 *----------------------------------------------------------------------*
                                403 *        LOAD ADDRESS OF CALL PARAMETERS                              *
                                404 *----------------------------------------------------------------------*

00002A 9858 1000         00000  406          LM    R5,R8,0(R1)     LOAD ADDRESS OF FOUR CALL PARAMETERS
                         00000  407          USING DAX,R5          R5=BASE FOR INTERFACE CONTROL BLOCK
                         00000  408          USING DL1_SEG1,R6     R6=A(SEGMENT IN ITS DL/I DB FORMAT)
                         00000  409          USING DPR_SEG1,R7     R7=A(SEGMENT IN ITS DPROP FORMAT)
                         00000  410          USING ANCHOR,R8       R8=A(ANCHOR AREA)

                                412 *----------------------------------------------------------------------*
                                413 *        SET IN THE INTERFACE BLOCK THE                               *
                                414 *        'EXIT ENTERED' AND 'EXIT IN CONTROL' FLAGS.                  *
                                415 *----------------------------------------------------------------------*

00002E 92E7 51A2      001A2     417          MVI   DAXENTRD,C'X'   SET 'EXIT ENTERED'
000032 92E7 51A3      001A3     418          MVI   DAXINCTL,C'X'   SET 'EXIT IN CONTROL'

                                420 *----------------------------------------------------------------------*
                                421 *        IF THIS IS THE FIRST INVOCATION:                             *
                                422 *         - GETMAIN AN AREA CONTAINING                                *
                                423 *             -- OUR SAVE-AREA                                        *
                                424 *             -- MODULE-WORKSPACE                                     *
                                425 *         - CLEAR THE GETMAINED AREA WITH BINARY ZEROES               *
                                426 *----------------------------------------------------------------------*

000036 58B0 8000         00000  428          L     R11,ANCHOR_PTR  R11=A(GETMAINED AREA)
00003A 12BB                     429          LTR   R11,R11         IS THIS ADDRESS ZERO?
00003C 4770 C068         00068  430          BNZ   NOTFIRST        ...NO>>>FIRST TIME PROCESSING DONE

                                432          GETMAIN RU,LV=GETML,LOC=ANY GETMAIN AN AREA
000040                          433+         CNOP  0,4
000040 47F0 C04C         0004C  434+         B     *+12-4*0-2*0       BRANCH AROUND DATA
000044 0000005E                 435+         DC    A(GETML)           LENGTH
000048 00                       436+IHB0002F DC    AL1(0)             RESERVED
000049 00                       437+         DC    AL1(0)             RESERVED
00004A 00                       438+         DC    AL1(0)             SUBPOOL
00004B 72                       439+         DC    BL1'01110010'                 MODE BYTE         @G860P30
00004C 5800 C044         00044  440+         L     0,*-8+2*0          LOAD LENGTH
000050 58F0 C048         00048  441+         L     15,IHB0002F                       LOAD GETMAIN PARMS
000054 1B11                     442+         SR    1,1                     ZERO RESERVED REG 1
000056 0A78                     443+         SVC   120                     ISSUE GETMAIN SVC

000058 18B1                     445          LR    R11,R1          R11=A(GETMAINED AREA)
00005A 50B0 8000         00000  446          ST    R11,ANCHOR_PTR  SAVE ADDRESS GETMAINED AREA

00005E 1801                     448          LR    R0,R1           SET UP
000060 4110 005E         0005E  449          LA    R1,GETML        ...FOR A
000064 1BFF                     450          SR    R15,R15         ...ZEROING
000066 0E0E                     451          MVCL  R0,R14          ...MVCL
000068                          452 NOTFIRST DS    0H

                                454 *----------------------------------------------------------------------*
                                455 *        CHAIN TOGETHER OUR SAVE-AREA AND THE HIGHER LEVEL SAVEAREA   *
                                456 *        AND LOAD INTO R13 THE ADDRESS OF OUR SAVE-AREA               *
                                457 *----------------------------------------------------------------------*

000068 50BD 0008         00008  459          ST    R11,8(R13)      CHAIN OUR SAVE-AREA INTO HIGHER
00006C 50DB 0004         00004  460          ST    R13,4(R11)      CHAIN HIGHER SAVE-AREA INTO OUR
000070 18DB                     461          LR    R13,R11         R13=A(OUR SAVE-AREA)
                         00000  462          USING GETM,R13        ESTABLISH BASE REGISTER FOR WORKAREA
```

Figure 8 (Part 7 of 23). First Sample Segment Exit Routine (Assembler)

```
                                ************************************************************************
                                ************************************************************************
                                ************************************************************************
                                ****                                                                ****
                                ****          VERIFY THAT:                                          ****
                                ****           - THE EXIT IS INVOKED TO FORMAT THE SEGMENT 'SEG1'   ****
                                ****             OF DB 'DB1'.                                       ****
                                ****           - THE AREA USED TO CONTAIN THE SEGMENT IN ITS        ****
                                ****             DPROP FORMAT HAS THE EXPECTED LENGTH.              ****
                                ****           - THE EXIT IS INVOKED WITH A SUPPORTED               ****
                                ****             CALL FUNCTION.                                     ****
                                ****                                                                ****
                                ************************************************************************
                                ************************************************************************
                                ************************************************************************

                                *----------------------------------------------------------------------*
                                *        VERIFY, THAT THE EXIT IS CALLED FOR THE TRANSFORMATION OF     *
                                *        THE CORRECT DBDNAME AND SEGMENT-NAME.                         *
                                *----------------------------------------------------------------------*

000072 D507 509C C358 0009C 00358          CLC   DAXDBNM,=CL8'DB1'   EXPECTED DBDNAME?
000078 4770 C28E            0028E          BNE   INVDBSEG            ...NO>>>THIS IS AN ERROR
00007C D507 504C C360 0004C 00360          CLC   DAXSEGM(8),=CL8'SEG1'  EXPECTED SEGMENT-NAME?
000082 4770 C28E            0028E          BNE   INVDBSEG            ...NO>>>THIS IS AN ERROR

                                *----------------------------------------------------------------------*
                                *        VERIFY, THAT THE EXIT IS CALLED WITH A SUPPORTED              *
                                *        CALL FUNCTION.                                                *
                                *----------------------------------------------------------------------*

000086 D501 5020 C3F0 00020 003F0          CLC   DAXCALL,=C'NO'      'NORMAL CALL'?
00008C 4780 C09E            0009E          BE    CALLNO              ...YES>>>B
000090 D501 5020 C3F2 00020 003F2          CLC   DAXCALL,=C'RV'      'REVERSE CALL'?
000096 4780 C18A            0018A          BE    CALLRV              ...YES>>>B
00009A 47F0 C27A            0027A          B     INVCALL             UNSUPPORTED CALL FUNCTION
                                ************************************************************************
                                ************************************************************************
                                ************************************************************************
                                ****                                                                ****
                                ****     'NORMAL CALL' TO TRANSFORM THE SEGMENT                     ****
                                ****     FROM ITS 'DL/I DB FORMAT' INTO ITS 'DPROP FORMAT'          ****
                                ****                                                                ****
                                ************************************************************************
                                ************************************************************************
                                ************************************************************************

00009E                          CALLNO   DS    0H

                                *----------------------------------------------------------------------*
                                *        VERIFY, THAT THE 3RD PARAMETER HAS THE EXPECTED LENGTH        *
                                *----------------------------------------------------------------------*

00009E D503 5080 C3B8 00080 003B8          CLC   DAXFLEN,=A(DPR_SEG1L) EXPECTED LENGTH OF PARAMETER?
0000A4 4770 C2A2            002A2          BNE   INVPARL             ...NO>>>THIS IS AN ERROR

                                ***********************************************************************
                                *       PROCESS THE FIRST, FIXED-FORMAT PORTION OF THE                *
                                *       SEGMENT:                                                      *
                                *                                                                     *
                                *       - VERIFY THAT THE SEGMENT IN ITS DL/I FORMAT IS LARGE         *
                                *         ENOUGH TO CONTAIN KEYFLD1 AND KEYFLD2.                      *
                                *       - MOVE KEYFLD1 AND KEYFLD2 TO THE DPROP-FORMAT OF THE SEGM.   *
                                ***********************************************************************
```

Figure 8 (Part 8 of 23). First Sample Segment Exit Routine (Assembler)

```
0000A8 D501 6000 C3F4 00000 003F4   530         CLC   DL1_SEG1LL,=AL2(DL1_FIXEDL) SEGMENT LARGE ENOUGH?
0000AE 47D0 C2B6            002B6   531         BNH   INVSEGL                     ...NO>>>THATS AN ERROR
0000B2 D201 7000 C3F6 00000 003F6   532         MVC   DPR_SEG1LL,=AL2(DPR_SEG1L)  SET LENGTH OF DPROP_SEG
0000B8 D201 7002 6002 00002 00002   533         MVC   DPR_KEYFLD1,DL1_KEYFLD1     MOVE KEYFLD1
0000BE D205 7004 6004 00004 00004   534         MVC   DPR_KEYFLD2,DL1_KEYFLD2     MOVE KEYFLD2


                                    536 *********************************************************************
                                    537 *       INITIALIZE 'CURRENT POINTER' WITHIN DL/I FORMAT            *
                                    538 *                                                                   *
                                    539 *       (IT IS REGISTER 2, WHICH IS USED AS CURRENT PTR            *
                                    540 *       WITHIN DL/I FORMAT)                                         *
                                    541 *********************************************************************

0000C4 4120 600A            0000A   543         LA    R2,DL1_SEG1VAR        R2=CURRENT ADDR IN DL1 FMT


                                    545 *********************************************************************
                                    546 *                                                                   *
                                    547 *        FOR EACH FIELD IN THE VARIABLE-FORMAT PORTION OF THE      *
                                    548 *        SEGMENT:                                                   *
                                    549 *           - VALIDATE THE FIELD LENGTH:                            *
                                    550 *                ---FIELD LENGTH SHOULD BE POSITIVE                 *
                                    551 *                ---FIELD LENGTH SHOULD NOT EXCEED A SPECIFIC      *
                                    552 *                   MAXIMAL LENGTH.                                 *
                                    553 *                ---FIELD SHOULD BE TOTALLY WITHIN THE SEGMENT.    *
                                    554 *                                                                   *
                                    555 *           - COPY THE FIELD AND ITS LENGTH-FIELD TO THE           *
                                    556 *             DPROP-FORMAT OF THE SEGMENT                           *
                                    557 *                                                                   *
                                    558 *********************************************************************

                                    560 *-------------------------------------------------------------------*
                                    561 *       PROCESS FAMILY NAME FIELD                                   *
                                    562 *-------------------------------------------------------------------*

                                    564 *
                                    565 ***           CHECK LENGTH FIELD
                                    566 *
0000C8 48F2 0000            00000   567         LH    R15,0(R2)             R15=LENGTH OF VC FIELD
0000CC 12FF                         568         LTR   R15,R15               LENGTH FIELD POSITIVE?
0000CE 47D0 C2DE            002DE   569         BNP   INVFAM2               ...NO>>>THATS AN ERROR
0000D2 55F0 C3BC            003BC   570         CL    R15,=A(30)            FAMILY FIELD LONGER THAN 30?
0000D6 4720 C2DE            002DE   571         BH    INVFAM2               ...YES>>>THATS AN ERROR
                                    572 *
                                    573 ***           CHECK THAT THE FIELD IS TOTALLY WITHIN THE SEGM
                                    574 *
0000DA 4102 F002            00002   575         LA    R0,2(R2,R15)          R0=A(END OF VC FIELD)+1
0000DE 4810 6000            00000   576         LH    R1,DL1_SEG1LL         R1=LENGTH OF SEGMENT
0000E2 4111 6000            00000   577         LA    R1,DL1_SEG1(R1)       R1=A(END OF SEGMENT)+1
0000E6 1901                         578         CR    R0,R1                 FLD TOTALLY WITHIN SEG?
0000E8 4720 C2CA            002CA   579         BH    INVFAM1               ...NO>>>THATS AN ERROR
                                    580 *
                                    581 ***           MOVE LENGTH FIELD INTO DPROP-FORMAT
                                    582 *
0000EC D201 700A 2000 0000A 00000   583         MVC   DPR_FAMILY_L,0(R2)    MOVE LENGTH FIELD
                                    584 *
                                    585 ***           MOVE VC FIELD INTO DPROP-FORMAT
                                    586 *
0000F2 4102 0002            00002   587         LA    R0,2(R2)              R0=START FOR MVCL
0000F6 181F                         588         LR    R1,R15                R1=LENGTH FIR MVCL
0000F8 BF18 C3F8            003F8   589         ICM   R1,8,=C' '            PADDING BLANK FOR MVCL
0000FC 41E0 700C            0000C   590         LA    R14,DPR_FAMILY        R14=TARGET ADDRESS FOR MVCL
000100 41F0 001E            0001E   591         LA    R15,30                R15=TARGET LENGTH FOR MVCL
000104 0EE0                         592         MVCL  R14,R0                MOVE THAT FIELD
```

Figure 8 (Part 9 of 23). First Sample Segment Exit Routine (Assembler)

```
                                   593 *
                                   594 ***      ADJUST 'CURRENT POINTER WITHIN DL/I DB FORMAT'
                                   595 *
000106 1820                        596          LR    R2,R0                  R2=START OF NEXT FIELD


                                   598 *-------------------------------------------------------------------*
                                   599 *        PROCESS FIRST NAME FIELD                                   *
                                   600 *-------------------------------------------------------------------*

                                   602 *
                                   603 ***            CHECK LENGTH FIELD
                                   604 *
000108 48F2 0000            00000  605          LH    R15,0(R2)              R15=LENGTH OF VC FIELD
00010C 12FF                        606          LTR   R15,R15                LENGTH FIELD POSITIVE?
00010E 47D0 C306            00306  607          BNP   INVFRST2               ...NO>>>THATS AN ERROR
000112 55F0 C3C0            003C0  608          CL    R15,=A(20)             FIRST FIELD LONGER THAN 20?
000116 4720 C306            00306  609          BH    INVFRST2               ...YES>>>THATS AN ERROR
                                   610 *
                                   611 ***            CHECK THAT THE FIELD IS TOTALLY WITHIN THE SEGM
                                   612 *
00011A 4102 F002            00002  613          LA    R0,2(R2,R15)           R0=A(END OF VC FIELD)+1
00011E 4810 6000            00000  614          LH    R1,DL1_SEG1LL          R1=LENGTH OF SEGMENT
000122 4111 6000            00000  615          LA    R1,DL1_SEG1(R1)        R1=A(END OF SEGMENT)+1
000126 1901                        616          CR    R0,R1                  FLD TOTALLY WITHIN SEG?
000128 4720 C2F2            002F2  617          BH    INVFRST1               ...NO>>>THATS AN ERROR
                                   618 *
                                   619 ***            MOVE LENGTH FIELD INTO DPROP-FORMAT
                                   620 *
00012C D201 702A 2000 0002A 00000  621          MVC   DPR_FIRST_L,0(R2)      MOVE LENGTH FIELD
                                   622 *
                                   623 ***            MOVE VC FIELD INTO DPROP FORMAT
                                   624 *
000132 4102 0002            00002  625          LA    R0,2(R2)               R0=START FOR MVCL
000136 181F                        626          LR    R1,R15                 R1=LENGTH FOR MVCL
000138 BF18 C3F8            003F8  627          ICM   R1,8,=C' '             PADDING BLANK FOR MVCL
00013C 41E0 702C            0002C  628          LA    R14,DPR_FIRST          R14=TARGET ADDRESS FOR MVCL
000140 41F0 0014            00014  629          LA    R15,20                 R15=TARGET LENGTH FOR MVCL
000144 0EE0                        630          MVCL  R14,R0                 MOVE THAT FIELD
                                   631 *
                                   632 ***      ADJUST 'CURRENT POINTER WITHIN DL/I DB FORMAT'
                                   633 *
000146 1820                        634          LR    R2,R0                  R2=START OF NEXT FIELD


                                   636 *-------------------------------------------------------------------*
                                   637 *        PROCESS CITY FIELD                                         *
                                   638 *-------------------------------------------------------------------*

                                   640 *
                                   641 ***            CHECK LENGTH FIELD
                                   642 *
000148 48F2 0000            00000  643          LH    R15,0(R2)              R15=LENGTH OF VC FIELD
00014C 12FF                        644          LTR   R15,R15                LENGTH FIELD POSITIVE?
00014E 47D0 C32E            0032E  645          BNP   INVCITY2               ...NO>>>THATS AN ERROR
000152 55F0 C3C4            003C4  646          CL    R15,=A(35)             CITY FIELD LONGER THAN 35?
000156 4720 C32E            0032E  647          BH    INVCITY2               ...YES>>>THATS AN ERROR
                                   648 *
                                   649 ***            CHECK THAT THE FIELD IS TOTALLY WITHIN THE SEGM
                                   650 *
00015A 4102 F002            00002  651          LA    R0,2(R2,R15)           R0=A(END OF VC FIELD)+1
00015E 4810 6000            00000  652          LH    R1,DL1_SEG1LL          R1=LENGTH OF SEGMENT
000162 4111 6000            00000  653          LA    R1,DL1_SEG1(R1)        R1=A(END OF SEGMENT)+1
000166 1901                        654          CR    R0,R1                  FLD TOTALLY WITHIN SEG?
000168 4720 C31A            0031A  655          BH    INVCITY1               ...NO>>>THATS AN ERROR
```

Figure 8 (Part 10 of 23). First Sample Segment Exit Routine (Assembler)

```
                                    656 *
                                    657 ***          MOVE LENGTH FIELD INTO DPROP-FORMAT
                                    658 *
00016C D201 7040 2000 00040 00000   659        MVC   DPR_CITY_L,0(R2)       MOVE LENGTH FIELD
                                    660 *
                                    661 ***          MOVE VC FIELD INTO DPROP-FORMAT
                                    662 *
000172 4102 0002            00002   663        LA    R0,2(R2)               R0=START FOR MVCL
000176 181F                         664        LR    R1,R15                 R1=LENGTH FOR MVCL
000178 BF18 C3F8            003F8   665        ICM   R1,8,=C' '             PADDING BLANK FOR MVCL
00017C 41E0 7042            00042   666        LA    R14,DPR_CITY           R14=TARGET ADDRESS FOR MVCL
000180 41F0 0023            00023   667        LA    R15,35                 R15=TARGET LENGTH FOR MVCL
000184 0EE0                         668        MVCL  R14,R0                 MOVE THAT FIELD

000186 47F0 C26A            0026A   670        B     RETURN
                                    672 ***********************************************************************
                                    673 ***********************************************************************
                                    674 ***********************************************************************
                                    675 ****                                                               ****
                                    676 ****     'REVERSE CALL' TO TRANSFORM THE SEGMENT                   ****
                                    677 ****     FROM ITS 'DPROP FORMAT' INTO ITS 'DL/I DB FORMAT'         ****
                                    678 ****                                                               ****
                                    679 ***********************************************************************
                                    680 ***********************************************************************
                                    681 ***********************************************************************

00018A                              683 CALLRV   DS    0H

                                    685 **********************************************************************
                                    686 *       PROCESS THE FIRST, FIXED-FORMAT PORTION OF THE               *
                                    687 *       SEGMENT:                                                     *
                                    688 *                                                                    *
                                    689 *       - VERIFY THAT THE DLI FORMAT BUFFER IS LARGE ENOUGH TO       *
                                    690 *         CONTAIN THE LARGEST POSSIBLE SEGMENT.                      *
                                    691 *       - VERIFY THAT THE SEGMENT IN ITS DPROP FORMAT IS LARGE       *
                                    692 *         ENOUGH TO CONTAIN KEYFLD1 AND KEYFLD2.                     *
                                    693 *       - MOVE KEYFLD1 AND KEYFLD2 TO THE DL/I-FORMAT OF THE SEGM.   *
                                    694 **********************************************************************

00018A D503 507C C3C8 0007C 003C8   696        CLC   DAXDLEN,=A(DL1_CITY+L'DL1_CITY-DL1_SEG1) ROOM FOR SEG?
000190 4740 C2B6            002B6   697        BL    INVSEGL                     ...NO>>>THATS AN ERROR
000194 D503 5080 C3CC 00080 003CC   698        CLC   DAXFLEN,=A(DPR_SEG1KEY+L'DPR_SEG1KEY-DPR_SEG1) IS AT
00019A 4740 C2A2            002A2   699        BL    INVPARL                     LEAST KEY HERE? ->NO, ERR
00019E D201 6002 7002 00002 00002   700        MVC   DL1_KEYFLD1,DPR_KEYFLD1     MOVE KEYFLD1
0001A4 D205 6004 7004 00004 00004   701        MVC   DL1_KEYFLD2,DPR_KEYFLD2     MOVE KEYFLD2


                                    703 **********************************************************************
                                    704 *        INITIALIZE 'CURRENT POINTER' WITHIN DL/I FORMAT             *
                                    705 *                                                                    *
                                    706 *        (IT IS REGISTER 2, WHICH IS USED AS CURRENT PTR             *
                                    707 *        WITHIN DL/I FORMAT)                                         *
                                    708 **********************************************************************

0001AA 4120 600A            0000A   710        LA    R2,DL1_SEG1VAR         R2=CURRENT ADDR IN DL1 FMT
```

Figure 8 (Part 11 of 23). First Sample Segment Exit Routine (Assembler)

```
                                       712 ********************************************************************
                                       713 *                                                                  *
                                       714 *         FOR EACH FIELD IN THE VARIABLE-FORMAT PORTION OF THE     *
                                       715 *         SEGMENT:                                                 *
                                       716 *            - VALIDATE IF THE FIELD IS REALLY PRESENT (MAY BE     *
                                       717 *              TRUNCATED IF "NULL" ON THE DB2 SIDE)                *
                                       718 *            - VALIDATE THE FIELD LENGTH:                          *
                                       719 *                 ---FIELD LENGTH SHOULD BE POSITIVE               *
                                       720 *                 ---FIELD LENGTH SHOULD NOT EXCEED A SPECIFIC     *
                                       721 *                    MAXIMAL LENGTH.                               *
                                       722 *                                                                  *
                                       723 *            - COPY THE FIELD AND ITS LENGTH-FIELD TO THE          *
                                       724 *              DL/I-FORMAT OF THE SEGMENT                          *
                                       725 *                                                                  *
                                       726 ********************************************************************

                                       728 *------------------------------------------------------------------*
                                       729 *        PROCESS FAMILY NAME FIELD                                 *
                                       730 *------------------------------------------------------------------*

                                       732 *
                                       733 ***            CHECK LENGTH FIELD
                                       734 *
0001AE 17FF                            735          XR    R15,R15                PRESET LENGTH TO ZERO
0001B0 D503 5080 C3D0 00080 003D0      736          CLC   DAXFLEN,=A(DPR_FAMILY-DPR_SEG1)  IS LENGTH FIELD HERE?
0001B6 4740 C1D8            001D8      737          BL    CALLRV10               ...NO>>>THEN USE ZERO FIELD
0001BA 48F0 700A            0000A      738          LH    R15,DPR_FAMILY_L       R15=LENGTH OF VC FIELD
0001BE 12FF                            739          LTR   R15,R15                LENGTH FIELD NEGATIVE?
0001C0 4740 C2DE            002DE      740          BM    INVFAM2                ...YES>>>THATS AN ERROR
0001C4 59F0 C3D4            003D4      741          C     R15,=A(L'DPR_FAMILY)   FAMILY FIELD LONGER THAN MAX?
0001C8 4720 C2DE            002DE      742          BH    INVFAM2                ...YES>>>THATS AN ERROR
                                       743 *
                                       744 ***            CHECK THAT THE FIELD IS TOTALLY WITHIN THE SEGM
                                       745 *
0001CC 410F 000C            0000C      746          LA    R0,DPR_FAMILY-DPR_SEG1(R15) R0=DPR_FAMILY+L'DPR_FAMILY
0001D0 5900 5080            00080      747          C     R0,DAXFLEN             FLD TOTALLY WITHIN SEG?
0001D4 4720 C2CA            002CA      748          BH    INVFAM1                ...NO>>>THATS AN ERROR
                                       749 *
                                       750 ***            STORE LENGTH FIELD INTO DL/I-FORMAT
                                       751 *
0001D8                                 752 CALLRV10 DS    0H
0001D8 40F0 2000            00000      753          STH   R15,0(0,R2)            STORE LENGTH FIELD
                                       754 *
                                       755 ***            MOVE VC FIELD INTO DL/I FORMAT
                                       756 *
0001DC 4120 2002            00002      757          LA    R2,2(0,R2)             R2=START FOR MVCL
0001E0 183F                            758          LR    R3,R15                 R3=LENGTH FOR MVCL
0001E2 41E0 700C            0000C      759          LA    R14,DPR_FAMILY         R14=SOURCE ADDRESS FOR MVCL
0001E6 0E2E                            760          MVCL  R2,R14                 MOVE THAT FIELD


                                       762 *------------------------------------------------------------------*
                                       763 *        PROCESS FIRST NAME FIELD                                  *
                                       764 *------------------------------------------------------------------*

                                       766 *
                                       767 ***            CHECK LENGTH FIELD
                                       768 *
0001E8 17FF                            769          XR    R15,R15                PRESET LENGTH TO ZERO
0001EA D503 5080 C3D8 00080 003D8      770          CLC   DAXFLEN,=A(DPR_FIRST-DPR_SEG1)  IS LENGTH FIELD HERE?
0001F0 4740 C212            00212      771          BL    CALLRV30               ...NO>>>THEN USE ZERO FIELD
0001F4 48F0 702A            0002A      772          LH    R15,DPR_FIRST_L        R15=LENGTH OF VC FIELD
0001F8 12FF                            773          LTR   R15,R15                LENGTH FIELD NEGATIVE?
0001FA 4740 C306            00306      774          BM    INVFRST2               ...YES>>>THATS AN ERROR
0001FE 59F0 C3DC            003DC      775          C     R15,=A(L'DPR_FIRST)    FIRST NAME LONGER THAN MAX?
000202 4720 C306            00306      776          BH    INVFRST2               ...YES>>>THATS AN ERROR
```

Figure 8 (Part 12 of 23). First Sample Segment Exit Routine (Assembler)

```
                                 777 *
                                 778 ***          CHECK THAT THE FIELD IS TOTALLY WITHIN THE SEGM
                                 779 *
000206 410F 002C           0002C  780         LA    R0,DPR_FIRST-DPR_SEG1(R15) R0=DPR_FIRST+L'DPR_FIRST
00020A 5500 5080           00080  781         CL    R0,DAXFLEN              FLD TOTALLY WITHIN SEG?
00020E 4720 C2F2           002F2  782         BH    INVFRST1                ...NO>>>THATS AN ERROR
                                 783 *
                                 784 ***          STORE LENGTH FIELD INTO DL/I-FORMAT
                                 785 *
000212                           786 CALLRV30 DS    0H
000212 40F0 2000           00000  787         STH   R15,0(0,R2)             STORE LENGTH FIELD
                                 788 *
                                 789 ***          MOVE VC FIELD INTO DL/I FORMAT
                                 790 *
000216 4120 2002           00002  791         LA    R2,2(0,R2)              R2=START FOR MVCL
00021A 183F                       792         LR    R3,R15                  R3=LENGTH FOR MVCL
00021C 41E0 702C           0002C  793         LA    R14,DPR_FIRST           R14=SOURCE ADDRESS FOR MVCL
000220 0E2E                       794         MVCL  R2,R14                  MOVE THAT FIELD


                                 796 *-----------------------------------------------------------------*
                                 797 *        PROCESS CITY FIELD                                       *
                                 798 *-----------------------------------------------------------------*

                                 800 *
                                 801 ***          CHECK LENGTH FIELD
                                 802 *
000222 17FF                       803         XR    R15,R15                 PRESET LENGTH TO ZERO
000224 D503 5080 C3E0 00080 003E0 804         CLC   DAXFLEN,=A(DPR_CITY-DPR_SEG1)  IS LENGTH FIELD HERE?
00022A 4740 C24C           0024C  805         BL    CALLRV50                ...NO>>>THEN USE ZERO FIELD
00022E 48F0 7040           00040  806         LH    R15,DPR_CITY_L          R15=LENGTH OF VC FIELD
000232 12FF                       807         LTR   R15,R15                 LENGTH FIELD NEGATIVE?
000234 4740 C32E           0032E  808         BM    INVCITY2                ...YES>>>THATS AN ERROR
000238 59F0 C3E4           003E4  809         C     R15,=A(L'DPR_CITY)      CITY FIELD LONGER THAN MAX?
00023C 4720 C32E           0032E  810         BH    INVCITY2                ...YES>>>THATS AN ERROR
                                 811 *
                                 812 ***          CHECK THAT THE FIELD IS TOTALLY WITHIN THE SEGM
                                 813 *
000240 410F 0042           00042  814         LA    R0,DPR_CITY-DPR_SEG1(R15)  R0=DPR_FIRST+L'DPRFIRST
000244 5900 5080           00080  815         C     R0,DAXFLEN              FLD TOTALLY WITHIN SEG?
000248 4720 C31A           0031A  816         BH    INVCITY1                ...NO>>>THATS AN ERROR
                                 817 *
                                 818 ***          STORE LENGTH FIELD INTO DL/I-FORMAT
                                 819 *
00024C                           820 CALLRV50 DS    0H
00024C 40F0 2000           00000  821         STH   R15,0(0,R2)             STORE LENGTH FIELD
                                 822 *
                                 823 ***          MOVE VC FIELD INTO DL/I FORMAT
                                 824 *
000250 4120 2002           00002  825         LA    R2,2(0,R2)              R2=START FOR MVCL
000254 183F                       826         LR    R3,R15                  R3=LENGTH FOR MVCL
000256 41E0 7042           00042  827         LA    R14,DPR_CITY            R14=SOURCE ADDRESS FOR MVCL
00025A 0E2E                       828         MVCL  R2,R14                  MOVE THAT FIELD


                                 830 *******************************************************************
                                 831 *        SETUP LENGTH OF SEGMENT IN DL/I FORMAT                   *
                                 832 *                                                                 *
                                 833 *        REGISTER 2 POINTS NOW 1 BYTE PAST LAST USED WITHIN THE   *
                                 834 *        DL/I SEGMENT BUFFER. SUBTRACTING THE BUFFER START        *
                                 835 *        ADDRESS GIVES THE LENGTH OF THE DL/I SEGMENT.            *
                                 836 *******************************************************************
```

Figure 8 (Part 13 of 23). First Sample Segment Exit Routine (Assembler)

```
00025C 41F0 6000            00000  838          LA    R15,DL1_SEG1          POINT BEGIN OF SEGMENT
000260 1B2F                        839          SR    R2,R15                COMPUTE SEGMENT LENGTH
000262 4020 6000            00000  840          STH   R2,DL1_SEG1LL         SET LENGTH OF DL1_SEG


000266 47F0 C26A            0026A  842          B     RETURN
                                   844 ***********************************************************************
                                   845 ***********************************************************************
                                   846 ***********************************************************************
                                   847 ****                                                               ****
                                   848 ****     RETURN LOGIC:                                             ****
                                   849 ****          - IF USER REQUESTED TRACING: TRACE THE PROPAGATING   ****
                                   850 ****            SQL STATEMENT.                                     ****
                                   851 ****          - RETURN TO CALLER OF EXIT                           ****
                                   852 ****                                                               ****
                                   853 ***********************************************************************
                                   854 ***********************************************************************
                                   855 ***********************************************************************
                                   856 ***********************************************************************

                                   858 *---------------------------------------------------------------------*
                                   859 *        RETURN TO CALLER OF THIS EXIT                                *
                                   860 *---------------------------------------------------------------------*

00026A                             862 RETURN   DS    0H
00026A 58DD 0004            00004  863          L     R13,4(R13)         R13=A(HIGHER SAVE-AREA)
00026E 98EC D00C            0000C  864          LM    R14,R12,12(R13)    RELOAD REGISTERS OF CALLER
000272 9601 D00F      0000F        865          OI    15(R13),X'01'      SET RETURN INDICATION
000276 1BFF                        866          SR    R15,R15            SET ZERO RETURN-CODE
000278 07FE                        867          BR    R14                RETURN LOGIC
                                   869 ***********************************************************************
                                   870 ***********************************************************************
                                   871 ***********************************************************************
                                   872 ****                                                               ****
                                   873 ****     ERROR LOGIC:                                              ****
                                   874 ****          - BUILD IN THE INTERFACE CONTROL BLOCK AN            ****
                                   875 ****            ERROR MESSAGE CONTAINING:                          ****
                                   876 ****               - A 8-BYTE MESSAGE-ID                           ****
                                   877 ****               - A DESCRIPTION OF THE TYPE OF FAILURE          ****
                                   878 ****          - SET A RETURN CODE IN THE INTERFACE CONTROL BLOCK   ****
                                   879 ****          - RETURN TO CALLER OF THE EXIT                       ****
                                   880 ****                                                               ****
                                   881 ***********************************************************************
                                   882 ***********************************************************************
                                   883 ***********************************************************************

00027A                             885 INVCALL  DS    0H
00027A D207 51A8 C368 001A8 00368  886          MVC   MSGID,=CL8'EKYESE0E'
000280 9240 51B0      001B0        887          MVI   MSGBL1,C' '
000284 D236 51B1 C3F9 001B1 003F9  888          MVC   MSGTXT,=CL55'CALL FUNCTION NOT SUPPORTED'      '
00028A 47F0 C34C            0034C  889          B     INVRC16


00028E                             891 INVDBSEG DS    0H
00028E D207 51A8 C370 001A8 00370  892          MVC   MSGID,=CL8'EKYESE1E'
000294 9240 51B0      001B0        893          MVI   MSGBL1,C' '
000298 D236 51B1 C430 001B1 00430  894          MVC   MSGTXT,=CL55'UNSUPPORTED DBD OR SEGNAME'
00029E 47F0 C34C            0034C  895          B     INVRC16


0002A2                             897 INVPARL  DS    0H
0002A2 D207 51A8 C378 001A8 00378  898          MVC   MSGID,=CL8'EKYESE2E'
0002A8 9240 51B0      001B0        899          MVI   MSGBL1,C' '
0002AC D236 51B1 C467 001B1 00467  900          MVC   MSGTXT,=CL55'3RD PARAMETER HAS INCORRECT LENGTH'
0002B2 47F0 C34C            0034C  901          B     INVRC16
```

Figure 8 (Part 14 of 23). First Sample Segment Exit Routine (Assembler)

```
0002B6                          903 INVSEGL  DS    0H
0002B6 D207 51A8 C380 001A8 00380   904          MVC   MSGID,=CL8'EKYESE3E'
0002BC 9240 51B0      001B0        905          MVI   MSGBL1,C' '
0002C0 D236 51B1 C49E 001B1 0049E   906          MVC   MSGTXT,=CL55'INVALID SEGMENT LENGTH'
0002C6 47F0 C342            00342   907          B     INVRC12


0002CA                          909 INVFAM1  DS    0H
0002CA D207 51A8 C388 001A8 00388   910          MVC   MSGID,=CL8'EKYESE4E'
0002D0 9240 51B0      001B0        911          MVI   MSGBL1,C' '
0002D4 D236 51B1 C4D5 001B1 004D5   912          MVC   MSGTXT,=CL55'FAMILY FIELD DOES NOT FIT WITHIN SEGMENT'
0002DA 47F0 C342            00342   913          B     INVRC12


0002DE                          915 INVFAM2  DS    0H
0002DE D207 51A8 C390 001A8 00390   916          MVC   MSGID,=CL8'EKYESE5E'
0002E4 9240 51B0      001B0        917          MVI   MSGBL1,C' '
0002E8 D236 51B1 C50C 001B1 0050C   918          MVC   MSGTXT,=CL55'LENGTH OF FAMILY FIELD IS INVALID'
0002EE 47F0 C342            00342   919          B     INVRC12


0002F2                          921 INVFRST1 DS    0H
0002F2 D207 51A8 C398 001A8 00398   922          MVC   MSGID,=CL8'EKYESE6E'
0002F8 9240 51B0      001B0        923          MVI   MSGBL1,C' '
0002FC D236 51B1 C543 001B1 00543   924          MVC   MSGTXT,=CL55'FIRST-NAME FIELD DOES NOT FIT WITHIN SEGMENC
                                                     T'
000302 47F0 C342            00342   925          B     INVRC12


000306                          927 INVFRST2 DS    0H
000306 D207 51A8 C3A0 001A8 003A0   928          MVC   MSGID,=CL8'EKYESE7E'
00030C 9240 51B0      001B0        929          MVI   MSGBL1,C' '
000310 D236 51B1 C57A 001B1 0057A   930          MVC   MSGTXT,=CL55'LENGTH OF FIRST-NAME FIELD IS INVALID'
000316 47F0 C342            00342   931          B     INVRC12


00031A                          933 INVCITY1 DS    0H
00031A D207 51A8 C3A8 001A8 003A8   934          MVC   MSGID,=CL8'EKYESE8E'
000320 9240 51B0      001B0        935          MVI   MSGBL1,C' '
000324 D236 51B1 C5B1 001B1 005B1   936          MVC   MSGTXT,=CL55'CITY FIELD DOES NOT FIT WITHIN SEGMENT'
00032A 47F0 C342            00342   937          B     INVRC12


00032E                          939 INVCITY2 DS    0H
00032E D207 51A8 C3B0 001A8 003B0   940          MVC   MSGID,=CL8'EKYESE9E'
000334 9240 51B0      001B0        941          MVI   MSGBL1,C' '
000338 D236 51B1 C5E8 001B1 005E8   942          MVC   MSGTXT,=CL55'LENGTH OF CITY FIELD IS INVALID'
00033E 47F0 C342            00342   943          B     INVRC12


000342                          945 INVRC12  DS    0H
000342 D203 51A4 C3E8 001A4 003E8   946          MVC   DAXRETC,=F'12'      SET RETURN CODE 12 (ERROR)
000348 47F0 C26A            0026A   947          B     RETURN

00034C                          949 INVRC16  DS    0H
00034C D203 51A4 C3EC 001A4 003EC   950          MVC   DAXRETC,=F'16'      SET RETURN CODE 16 (SEVERE ERROR)
000352 47F0 C26A            0026A   951          B     RETURN
000358                          953          LTORG
000358 C4C2F14040404040             954                =CL8'DB1'
000360 E2C5C7F140404040             955                =CL8'SEG1'
000368 C5D2E8C5E2C5F0C5             956                =CL8'EKYESE0E'
000370 C5D2E8C5E2C5F1C5             957                =CL8'EKYESE1E'
000378 C5D2E8C5E2C5F2C5             958                =CL8'EKYESE2E'
000380 C5D2E8C5E2C5F3C5             959                =CL8'EKYESE3E'
000388 C5D2E8C5E2C5F4C5             960                =CL8'EKYESE4E'
000390 C5D2E8C5E2C5F5C5             961                =CL8'EKYESE5E'
```

Figure 8 (Part 15 of 23). First Sample Segment Exit Routine (Assembler)

```
000398 C5D2E8C5E2C5F6C5          962              =CL8'EKYESE6E'
0003A0 C5D2E8C5E2C5F7C5          963              =CL8'EKYESE7E'
0003A8 C5D2E8C5E2C5F8C5          964              =CL8'EKYESE8E'
0003B0 C5D2E8C5E2C5F9C5          965              =CL8'EKYESE9E'
0003B8 00000065                  966              =A(DPR_SEG1L)
0003BC 0000001E                  967              =A(30)
0003C0 00000014                  968              =A(20)
0003C4 00000023                  969              =A(35)
0003C8 00000035                  970              =A(DL1_CITY+L'DL1_CITY-DL1_SEG1)
0003CC 0000000A                  971              =A(DPR_SEG1KEY+L'DPR_SEG1KEY-DPR_SEG1)
0003D0 0000000C                  972              =A(DPR_FAMILY-DPR_SEG1)
0003D4 0000001E                  973              =A(L'DPR_FAMILY)
0003D8 0000002C                  974              =A(DPR_FIRST-DPR_SEG1)
0003DC 00000014                  975              =A(L'DPR_FIRST)
0003E0 00000042                  976              =A(DPR_CITY-DPR_SEG1)
0003E4 00000023                  977              =A(L'DPR_CITY)
0003E8 0000000C                  978              =F'12'
0003EC 00000010                  979              =F'16'
0003F0 D5D6                      980              =C'NO'
0003F2 D9E5                      981              =C'RV'
0003F4 000A                      982              =AL2(DL1_FIXEDL)
0003F6 0065                      983              =AL2(DPR_SEG1L)
0003F8 40                        984              =C' '
0003F9 C3C1D3D340C6E4D5          985              =CL55'CALL FUNCTION NOT SUPPORTED'
000430 E4D5E2E4D7D7D6D9          986              =CL55'UNSUPPORTED DBD OR SEGNAME'
000467 F3D9C440D7C1D9C1          987              =CL55'3RD PARAMETER HAS INCORRECT LENGTH'
00049E C9D5E5C1D3C9C440          988              =CL55'INVALID SEGMENT LENGTH'
0004D5 C6C1D4C9D3E840C6          989              =CL55'FAMILY FIELD DOES NOT FIT WITHIN SEGMENT'
00050C D3C5D5C7E3C840D6          990              =CL55'LENGTH OF FAMILY FIELD IS INVALID'
000543 C6C9D9E2E360D5C1          991              =CL55'FIRST-NAME FIELD DOES NOT FIT WITHIN SEGMENT'
00057A D3C5D5C7E3C840D6          992              =CL55'LENGTH OF FIRST-NAME FIELD IS INVALID'
0005B1 C3C9E3E840C6C9C5          993              =CL55'CITY FIELD DOES NOT FIT WITHIN SEGMENT'
0005E8 D3C5D5C7E3C840D6          994              =CL55'LENGTH OF CITY FIELD IS INVALID'
                                 996 ***********************************************************************
                                 997 *       DESCRIPTION OF GETMAINED AREA CONTAINING:                     *
                                 998 *            - SAVE-AREA                                              *
                                 999 *            - EXIT WORKSPACE                                         *
                                1000 ***********************************************************************

000000                          1002 GETM     DSECT
                                1003 *---------------------------------------------------------------------*
                                1004 *       REGISTER SAVE-AREA                                            *
                                1005 *---------------------------------------------------------------------*
000000                          1006 SAVE     DS    18F                REGISTER SAVE-AREA

                                1008 *---------------------------------------------------------------------*
                                1009 *       WORK SPACE FOR EXIT                                           *
                                1010 *---------------------------------------------------------------------*

000048                          1012 EXITWORK DS    CL22               NOT USED BY THIS SAMPLE EXIT

                          0005E 1014 GETML    EQU   *-GETM             LENGTH OF GETMAINED AREA
                                1016 **********************************************************************
                                1017 *       DESCRIPTIONS OF SEGMENT SEG1 IN                              *
                                1018 *         - ITS DL/I DB FORMAT                                       *
                                1019 *         - ITS DPROP FORMAT                                         *
                                1020 **********************************************************************

                                1022 *---------------------------------------------------------------------*
                                1023 *           DESCRIPTION OF SEGMENT 'SEG1' IN ITS                      *
                                1024 *           VARIABLE-LENGTH DL/I DB FORMAT                            *
```

Figure 8 (Part 16 of 23). First Sample Segment Exit Routine (Assembler)

```
                                   *-------------------------------------------------------------------*

000000                                  DL1_SEG1     DSECT ,
                                        *
                                        ***           SEGMENT PORTION WITH FIXED-LENGTH FIELDS HAVING
                                        ***           A FIXED START POSITION.
                                        *
000000                                  DL1_SEG1LL   DS    H                   LENGTH OF SEGMENT
000002                                  DL1_SEG1KEY  DS    0CL8                KEY FIELD
000002                                  DL1_KEYFLD1  DS    CL2                 SUB-FIELD OF KEY OF SEG1
000004                                  DL1_KEYFLD2  DS    CL6                 SUB-FIELD OF KEY OF SEG1
                            0000A       DL1_FIXEDL   EQU   *-DL1_SEG1          LENGTH OF FIXED PORTION
                                        *
                                        ***           START OF VARIABLE SEGMENT PORTION WITH CONTIGUOUS
                                        ***           PAIRS OF:
                                        ***             (LENGTH FIELD,VARIABLE-LENGTH FIELD).
                                        ***           WITH THE EXCEPTION OF THE FIRST PAIR: ALL
                                        ***           PAIRS HAVE A VARIABLE START POSITION (SINCE THE
                                        ***           PAIRS ARE STORED ADJACENTLY IN ORDER TO CONSERVE
                                        ***           DASD STORAGE IN THE DL/I DB).
                                        *
                            0000A       DL1_SEG1VAR  EQU   *
                                        *
00000A                                  DL1_FAMILY_L DS    HL2                 LENGTH OF FAMILY NAME
00000C                                  DL1_FAMILY   DS    CL30                FAMILY NAME
                                        *
00002A                      0000D                    ORG   DL1_FAMILY+1
00000D                                  DL1_FIRST_L  DS    HL2                 LENGTH OF FIRST NAME
00000F                                  DL1_FIRST    DS    CL20                FIRST NAME
                                        *
000023                      00010                    ORG   DL1_FIRST+1
000010                                  DL1_CITY_L   DS    HL2                 LENGTH OF CITY-NAME
000012                                  DL1_CITY     DS    CL35                CITY-NAME
000035                      00035                    ORG

                                        *-------------------------------------------------------------------*
                                        *            DESCRIPTION OF SEGMENT 'SEG1' IN ITS DPROP FIXED-FORMAT.*
                                        *                                                                   *
                                        *            IN THIS FIXED FORMAT, THE PAIRS OF                     *
                                        *                (LENGTH FIELD,VARIABLE-LENGTH FIELD)               *
                                        *            ARE NOT IMMEDIATELY ADJACENT. INSTEAD THIS FORMAT      *
                                        *            RESERVES FOR EACH VARIABLE LENGTH FIELD ENOUGH STORAGE *
                                        *            FOR ITS MAXIMUM FIELD LENGTH.                          *
                                        *            THEREFORE EACH PAIR STARTS AT A FIXED LOCATION         *
                                        *            WITHIN THE SEGMENT.                                    *
                                        *-------------------------------------------------------------------*

000000                                  DPR_SEG1     DSECT ,
000000                                  DPR_SEG1LL   DS    H                   SEGMENT LENGTH
000002                                  DPR_SEG1KEY  DS    0CL8                KEY FIELD
000002                                  DPR_KEYFLD1  DS    CL2                 SUB-FIELD OF KEY OF SEG1
000004                                  DPR_KEYFLD2  DS    CL6                 SUB-FIELD OF KEY OF SEG1
                                        *
00000A                                  DPR_FAMILY_L DS    HL2                 LENGTH OF FAMILY NAME
00000C                                  DPR_FAMILY   DS    CL30                FAMILY NAME
                                        *
00002A                                  DPR_FIRST_L  DS    HL2                 LENGTH OF FIRST NAME
00002C                                  DPR_FIRST    DS    CL20                FIRST NAME
                                        *
000040                                  DPR_CITY_L   DS    HL2                 LENGTH OF CITY NAME
000042                                  DPR_CITY     DS    CL35                CITY NAME
                                        *
                            00065       DPR_SEG1L    EQU   *-DPR_SEG1          LENGTH OF SEGMENT
```

Figure 8 (Part 17 of 23). First Sample Segment Exit Routine (Assembler)

```
        ***********************************************************************
        *            DESCRIPTION OF ANCHOR AREA                               *
        ***********************************************************************

000000  ANCHOR       DSECT ,
000000  ANCHOR_PTR   DS    F'0'                PTR TO GETMAINED AREA
000004               DS    CL60' '             NOT USED
                 EKYRCDAX ,                EXIT INTERFACE CONTROL BLOCK
        +***************** START OF CONTROL BLOCK SPECIFICATION **************/
        +*                                                                    */
        +*       CONTROL BLOCK NAME:                                          */
        +*         EKYRCDAX (DAX)                                             */
        +*                                                                    */
        +*       DESCRIPTIVE NAME:                                            */
        +*         DPROP SEGMENT EXIT INTERFACE BLOCK                         */
        +*                                                                    */
        +*                                                                    */
        +**********************************************************************
        +*                                                                    *
        +*         THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
        +*                                                                    *
        +*         5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
        +*         ALL RIGHTS RESERVED.                                       *
        +*                                                                    *
        +*         U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
        +*         USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
        +*         GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
        +*                                                                    *
        +*         LICENSED MATERIALS - PROPERTY OF IBM.                      *
        +*                                                                    *
        +**********************************************************************
        +*                                                                    */
        +*       STATUS: V1 R2 M0                                             */
        +*                                                                    */
        +*       FUNCTION:                                                    */
        +*         THIS IS THE CONTROL BLOCK USED TO INTERFACE BETWEEN        */
        +*              - DPROP OR DXT                                        */
        +*              AND                                                   */
        +*              - A USER'S SEGMENT EXIT ROUTINE (THESE USER           */
        +*                EXIT ROUTINES ARE CALLED BY DXT 'USER DATA          */
        +*                EXIT ROUTINES')                                     */
        +*                                                                    */
        +*         THERE IS ONE DAX CONTROL BLOCK FOR EACH SEGMENT            */
        +*         EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT         */
        +*         IN VIRTUAL STORAGE.                                        */
        +*         FOR SYNCH PROPAGATION IN MPP REGIONS:                      */
        +*         - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER       */
        +*           SUBTASK.                                                 */
        +*         FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR            */
        +*         CCU AND DLU PROCESSING, AND FOR ASYNCH PROPAGATION         */
        +*         (DEPENDING ON HOW AYSNCH PROPAGATION IS IMPLEMENTED):      */
        +*         - THIS IS THE DURATION OF THE JOBSTEP.                     */
        +*                                                                    */
        +*--------------------------------------------------------------------*/
        +*       IMPORTANT NOTES:                                             */
        +*       ================                                             */
        +*       - SINCE THE SAME USER EXIT ROUTINE CAN BE INVOKED BOTH       */
        +*         BY DPROP AND BY DXT: CHANGES TO THIS CONTROL BLOCK MUST    */
        +*         BE COORDINATED BETWEEN DPROP DEVELOPMENT AND DXT           */
        +*         DEVELOPMENT.                                               */
        +*                                                                    */
        +*       - FIELDS MARKED IN THE COMMENT WITH '***DXT ONLY***'         */
        +*         HAVE NO MEANING, WHEN THE SEGMENT USER EXIT                */
        +*         ROUTINE IS INVOKED BY DPROP.                               */
        +*--------------------------------------------------------------------*/
```

Figure 8 (Part 18 of 23). First Sample Segment Exit Routine (Assembler)

```
                           1155++*                                                           */
                           1156++*       MODULE TYPE= MACRO                                  */
                           1157++*          PROCESSOR= ASSEMBLER H                           */
                           1158++*                                                           */
                           1159++*       INNER CONTROL BLOCKS: NONE                          */
                           1160++*                                                           */
                           1161++*       MACROS USED FROM MACRO LIBRARY: NONE                */
                           1162++*                                                           */
                           1163++*       CHANGE ACTIVITY:                                    */
                           1164++*                     KMP0057  12/13/90                     */
                           1165++*                     KMP0060  02/08/91 COPYRIGHT INFORMATION */
                           1166++*                     KMPREL2  03/20/91                     */
                           1167++*                                                           */
                           1168+******************** END OF CONTROL BLOCK SPECIFICATION **************/

000000                     1170+DAX      DSECT
                    00000  1171+DVRDAX   EQU   *                     LABEL FOR DXT COMPATIBILITY
                           1172+*-------------------------------------------------------------------*
                           1173+*   THIS SECTION OF THE CB MAY NOT BE MODIFIED BY EXIT             *
                           1174+*-------------------------------------------------------------------*
000000                     1175+DAXPFX   DS    0CL32                 PREFIX OF CONTROL BLOCK
000000                     1176+DAXTNAME DS    CL8                   EYE CATCHER: "DVRXCDAX"
000008                     1177+DAXRSVD  DS    CL24                  RESERVED FOR DXT INTERNAL USE
                           1178+*
000020                     1179+DAXPFXE  DS    0CL448                PREFIX EXTENSION
000020                     1180+DAXCALL  DS    CL2                   TYPE OF CALL TO EXIT:
                           1181+*                                      =C'NO' - NORMAL CALL,
                           1182+*                                         ISSUED TO CONVERT DATA FROM
                           1183+*                                         'IMS DATABASE FORMAT' TO
                           1184+*                                         'DPROP/DXT' FORMAT
                           1185+*                                      =C'RV' - REVERSE CALL
                           1186+*                                         ISSUED TO CONVERT
                           1187+*                                         DATA FROM:
                           1188+*                                         'DPROP/DXT' FORMAT
                           1189+*                                         TO
                           1190+*                                         'IMS DATABASE FORMAT'
                           1191+*                  ***DXT ONLY***      =C'RE' - RETURN CALL, ISSUED
                           1192+*                                         INSTEAD OF NEXT REQUEST FOR
                           1193+*                                         NEW DATA AT REQUEST OF EXIT
                           1194+*                                         (SEE DAXRETC VALUE 4)
                           1195+*                  ***DXT ONLY***      =C'ED' - END-OF-DATA CALL
                           1196+*                                         ISSUED BY DXT.
                           1197+*
000022                     1198+DAXDATYP DS    CL2                   TYPE OF DATA BEING PASSED--
                           1199+*                                      =C'DL' - DL/I DATA
                           1200+*                  ***DXT ONLY***      =C'PS' - PHYSICAL SEQUENTIAL
                           1201+*                  ***DXT ONLY***      =C'VK' - VSAM KSDS DATA
                           1202+*                  ***DXT ONLY***      =C'VE' - VSAM ESDS DATA
                           1203+*                  ***DXT ONLY***      =C'GD' - GDI RECRD DATA
                           1204+*
000024                     1205+DAXFIL   DS    CL32                  NAME OF FILE OR PCB FROM WHICH
                           1206+*                                    DATA IS BEING PASSED
                           1207+*
000044                     1208+DAXPSB   DS    CL8                   NAME OF PSB IF TYPE IS "DL"
                           1209+*
00004C                     1210+DAXSEGM  DS    CL32                  NAME OF SEGMENT IF TYPE IS "DL"
                           1211+*                                    IF CALLER IS DPROP:
                           1212+*                                       - NAME OF PHYSICAL SEGM.
                           1213+*                                    IF CALLER IS DXT:
                           1214+*                                       - NAME OF SEGM. SPECIFIED
                           1215+*                                         IN THE USED DBD (DBD CAN
                           1216+*                                         BE A PHYSICAL OR LOGICAL
                           1217+*                                         DBD)
                           1218+*
```

Figure 8 (Part 19 of 23). First Sample Segment Exit Routine (Assembler)

```
00006C                      1219+DAXPCBAD DS    AL4    ***DXT ONLY***  PTR TO PCB IF TYPE IS "DL"
                            1220+*
000070                      1221+DAXPCBLS DS    AL4    ***DXT ONLY***  PTR TO LIST OF DEM'S PCBS,
                            1222+*                                     IF DEM IS A DL/I DEM
                            1223+*
000074                      1224+DAXKFBAD DS    AL4                    PTR TO SEGMENT'S FULLY
                            1225+*                                     CONCAT KEY (IF DL/I).
                            1226+*                                     IF CALLER IS DPROP:
                            1227+*                                        - 0, IF 'NOKEY' HAS BEEN
                            1228+*                                          SPECIFIED ON EXIT=
                            1229+*                                          OF DBDGEN.
                            1230+*
000078                      1231+DAXKFBLN DS    F                      LENGTH OF SEGM'S FULLY
                            1232+*                                     CONCAT KEY (IF DL/I)
                            1233+*                                     IF DPROP: 0, IF 'NOKEY' HAS BEEN
                            1234+*                                               SPECIFIED ON EXIT=
                            1235+*                                               OF DBDGEN.
                            1236+*
00007C                      1237+DAXINLN  DS    0F
00007C                      1238+DAXDLEN  DS    F                      LENGTH OF IMS DB SEGMENT BUFFER
                            1239+*
000080                      1240+DAXOUTLN DS    0F
000080                      1241+DAXFLEN  DS    F                      LENGTH OF DPROP SEGMENT BUFFER
                            1242+*
000084                      1243+DAXSYSPR DS    AL4    ***DXT ONLY***  POINTER TO SYSPRINT DCB (EXIT
                            1244+*                                     MAY WISH TO RECORD INFORMATION
                            1245+*                                     IN SYSPRINT VIA "PUT"--
                            1246+*                                     DCB FACTS: LRECL=121,
                            1247+*                                        NO CARRIAGE CONTROL CHAR
                            1248+*
000088                      1249+DAXENVT  DS    0CL12                  ENVIRONMENT SUBFIELDS
000088                      1250+DAXOPSYS DS    CL4                    OPERATING SYSTEM:
                            1251+*                                      =C'ESA ' IF MVS/ESA
                            1252+*                 ***DXT ONLY***       =C'XA  ' IF MVS/XA
                            1253+*                 ***DXT ONLY***       =C'MVS ' IF MVS
                            1254+*
00008C                      1255+DAXTRANS DS    CL4                    DB/DC ENVIRONMENT:
                            1256+*                                      =C'BAT ' IF IMS BATCH/BMP
                            1257+*                                      =C'MPP ' IF IMS MPP
                            1258+*                                      =C'IFP ' IF FAST PATH
                            1259+*                                      =C'CICS' IF CICS
                            1260+*                                      =C'    ' IF NONE OF ABOV.
                            1261+*
000090                      1262+DAXPROGM DS    CL4                    CALLING PROGRAM:
                            1263+*                                      =C'DXT ' IF DataRefresher
                            1264+*                                      =C'DPRS' IF DPROP SYNCH PROP
                            1265+*                                      =C'DPRA' IF DPROP ASYNCH PROP
                            1266+*                                      =C'DPRC' IF DPROP CCU  PROP
                            1267+*                                      =C'DPRL' IF DPROP DLU
                            1268+*
000094                      1269+DAXEXIT  DS    CL8                    NAME OF THIS EXIT ROUTINE
                            1270+*
00009C                      1271+DAXDBNM  DS    CL8                    NAME OF IMS DATABASE
                            1272+*                                     IF CALLER IS DPROP:
                            1273+*                                      - NAME OF PHYSICAL DBD.
                            1274+*                                     IF CALLER IS DXT:
                            1275+*                                      - NAME OF USED DBD (CAN BE
                            1276+*                                        NAME OF A PHYSICAL OR
                            1277+*                                        LOGICAL DBD)
                            1278+*
0000A4                      1279+DAXDPRPN DS    CL24                   RESERVED
                            1280+*
0000BC                      1281+DAXASGNO DS    F      ***DXT ONLY***  NUMBER OF DAXASEGS ARRAY
                            1282+*                                     ELEMENTS CONTAINING
                            1283+*                                     ANCESTOR SEGM INFORMATION
```

Figure 8 (Part 20 of 23). First Sample Segment Exit Routine (Assembler)

```
                                   1284++*
0000C0                             1285+DAXASEGS DS    15CL12 ***DXT ONLY***  ARRAY OF ANCESTOR SEGMS,
                                   1286++*                                    ONLY FOR DL/I SEGM EXIT,
                                   1287++*                                    IN ORDER FROM ROOT TO
                                   1288++*                                    PARENT SEGMENT (EACH
                                   1289++*                                    ARRAY ELEMENT IS MAPPED
                                   1290++*                                    BY DAXANCTR DSECT, BELOW)
                                   1291++*
000174                             1292+DAXRSVD1 DS    CL46                   RESERVED FOR DXT USE
0001A2                      00174  1293+         ORG   DAXRSVD1               REDEFINE THIS AREA
                                   1294++*
000174                             1295+DAXDPRCT DS    CL4' ' --DPROP ONLY-- IF CALLER IS DPROP:
                                   1296++*                                    - EXIT IS CALLED TO PROCESS:
                                   1297++*                                      'ISRT': A DL/I OR DB2 INSERT
                                   1298++*                                      'DLET': A DL/I OR DB2 DELETE
                                   1299++*                                      'REPL': A DL/I OR DB2 REPLACE
                                   1300++*                                              (AFTER-REPLACE IMAGE)
                                   1301++*                                   IF CALLER IS DXT:
                                   1302++*                                    - NOT USED
000178                             1303+DAXREPL  DS    C' '   --DPROP ONLY-- IF CALLER IS DPROP AND IF
                                   1304++*                                   DAXDPRCT IS 'REPL':
                            000C1  1305+DAXREPLA EQU   C'A'                     'A': AFTER-REPLACE  IMAGE
                            000C2  1306+DAXREPLB EQU   C'B'                     'B': BEFORE-REPLACE IMAGE

000179                             1308+DAXSEGT  DS    C' '   --DPROP ONLY-- IF CALLER IS DPROP:
                                   1309++*                                    - TYPE OF SEGMENT PROCESSED:
                            000E4  1310+DAXSEGTU EQU   C'U'                     'U': UPDATED IMS SEGMENT
                            000C1  1311+DAXSEGTA EQU   C'A'                     'A': ANCESTOR OF UPDATED SEGM
                            000C9  1312+DAXSEGTI EQU   C'I'                     'I': INTERNAL SEGMENT

00017A                             1314+DAXPSUP  DS    C' '   --DPROP ONLY-- IF CALLER IS DPROP, DESCRIPTION
                                   1315++*                                   WHETHER PROPAGATION-SUPPRESSION
                                   1316++*                                   IS ALLOWED:
                            000D5  1317+DAXPSUPN EQU   C'N'                     'N': SUPPRESSION NOT ALLOWED
                            000E8  1318+DAXPSUPY EQU   C'Y'                     'Y': SUPPRESSION ALLOWED

00017B                             1320+         DS    C' '                  RESERVED
                                   1321++*
00017C                             1322+DAXISEGM DS    CL8' ' --DPROP ONLY-- IF CALLER IS DPROP:
                                   1323++*                                    - FOR RH PROPAGATION
                                   1324++*                                      NAME OF SEGMENT TO
                                   1325++*                                      PROCESS. SAME AS PHYS.
                                   1326++*                                      IMS SEGNAME IN DAXSEGM
                                   1327++*                                      IF NOT MAPPING CASE 3
                                   1328++*                                      ENTITY (INTERNAL)
                                   1329++*                                      SEGMENT IN PROCESS.
                                   1330++*                                   IF CALLER IS DXT:
                                   1331++*                                    - NOT USED
000184                             1332+DAXIDDSB DS    A      --DPROP ONLY-- IF CALLER IS DPROP:
                                   1333++*                                    - FOR RH PROPAGATION
                                   1334++*                                      POINTER TO THE BUFFER
                                   1335++*                                      CONTAINING THE 'BEFORE-CHANGE'
                                   1336++*                                      IMS DATABASE SEGMENT.
                                   1337++*                                      BUFFER CONTAINS THE
                                   1338++*                                      BEFORE IMAGE OF THE
                                   1339++*                                      IMS SEGMENT IF:
                                   1340++*                                      - DAXDPRCT EQ REPL, OR
                                   1341++*                                      - DAXDPRCT EQ DLET, OR
                                   1342++*                                      - DAXSEGT EQ DAXSEGTI
                                   1343++*                                        (INTERNAL SEGMENT OF
                                   1344++*                                        MAPPING CASE 3)
                                   1345++*                                      OR CONTAINS ALL BINARY
                                   1346++*                                      ZEROES IN OTHER CASES.
                                   1347++*                                      BUFFER IS READ ONLY
                                   1348++*                                      FOR THE EXIT ROUTINE.
```

Figure 8 (Part 21 of 23). First Sample Segment Exit Routine (Assembler)

```
000188                          1349+DAXIDDSL DS    A      --DPROP ONLY-- IF CALLER IS DPROP:
                                1350++                                     - FOR RH PROPAGATION
                                1351++                                       LENGTH OF THE 'BEFORE-CHANGE'
                                1352++                                       IMS DB SEGMENT POINTED-TO
                                1353++                                       BY DAXIDDSB.
00018C                   001A2  1354+         ORG
                                1355++                                    POINT TO THE END OF DAXRSVD1
                                1356++*-------------------------------------------------------------------*
                                1357++*   THE NEXT GROUP OF FIELDS MAY BE MODIFIED BY THE EXIT ROUTINE     *
                                1358++*-------------------------------------------------------------------*
0001A2                          1359+DAXENTRD DS    CL1                    SET BY EXIT ROUTINE TO
                                1360++*                                    C'X', INDICATES
                                1361++*                                    THAT EXIT HAS BEEN ENTERED
                                1362++*
0001A3                          1363+DAXINCTL DS    CL1                    SET BY EXIT ROUTINE TO
                                1364++*                                    C'X', INDICATES
                                1365++*                                    THAT EXIT IS IN CONTROL
                                1366++*
0001A4                          1367+DAXRETC  DS    F                      RETURN CODE--
                                1368++*                                      VALUE SET HERE BY EXIT,
                                1369++*
                                1370++*                                    RETURN CODE VALUES...
                         00000  1371+DAXRCOK  EQU   0                        = 0 - NORMAL, OUTPUT
                                1372++*                                            DATA RETURNED
                                1373++*
                         00004  1374+DAXRCOKR EQU   4      ***DXT ONLY***    = 4 - NORMAL, OUTPUT
                                1375++*                                            DATA RETURNED,
                                1376++*                                            DXT SHOULD
                                1377++*                                            RETURN TO EXIT FOR NEXT
                                1378++*                                            OCCURRENCE OF THIS RECORD
                                1379++*                                            OR SEGMENT
                                1380++*
                         00008  1381+DAXRCNQ  EQU   8                        = 8 - IF CALLER IS DPROP:
                                1382++*                                            DPROP WILL SUPPRESS
                                1383++*                                            THE PROPAGATION OF
                                1384++*                                            THE CHANGED DL/I DATA
                                1385++*                                          - IF CALLER IS DXT:
                                1386++*                                            DXT SHOULD NOT
                                1387++*                                            CONSIDER DATA TO
                                1388++*                                            BE ELIGIBLE FOR
                                1389++*                                            EXTRACT
                                1390++*
                         0000C  1391+DAXRCERB EQU   12                       =12   ERROR
                                1392++*                                          - IF CALLER IS DPROP:
                                1393++*                                            PROPAGATION FAILURE.
                                1394++*                                            DPROP/RUP WILL
                                1395++*                                            GO THROUGH ITS USUAL
                                1396++*                                            ERROR HANDLING LOGIC.
                                1397++*                                          - IF CALLER IS DXT:
                                1398++*                                            DXT SHOULD
                                1399++*                                            TERMINATE BATCH
                                1400++*
                         00010  1401+DAXRCERD EQU   16                       =16   ERROR
                                1402++*                                          - IF CALLER IS DPROP:
                                1403++*                                            RUP WILL ABEND
                                1404++*                                          - IF CALLER IS DXT:
                                1405++*                                            DXT SHOULD
                                1406++*                                            TERMINATE DEM EXECUTION
                                1407++*
0001A8                          1408+DAXSMESG DS    CL64                   TEXT OF MESSAGE PASSED
                                1409++*                                    FROM EXIT ROUTINE TO DPROP/DXT.
                                1410++*                                    ALL BLANKS MEANS NO MESSAGE.
```

Figure 8 (Part 22 of 23). First Sample Segment Exit Routine (Assembler)

```
                                    1411+*                                   - IF CALLER IS DPROP:
                                    1412+*                                     MSG WILL BE WRITTEN TO
                                    1413+*                                     VARIOUS DESTINATIONS ACCORDING
                                    1414+*                                     TO USUAL DPROP/RUP ERROR HANDLING
                                    1415+*                                     LOGIC IN MESSAGE EKYR980I OR
                                    1416+*                                     EKYR981E.
                                    1417+*                                   - IF CALLER IS DXT:
                                    1418+*                                     TEXT OF MESSAGE WILL BE
                                    1419+*                                     WRITTEN TO
                                    1420+*                                     SYSPRINT DATA SET IN MESSAGE
                                    1421+*                                     DVRA0_50.
                                    1422+*                                     (UNDERSCORE IS REPLACED
                                    1423+*                                     BY ONE OF SEVERAL DIGITS)
                                    1424+*                                     HAS EFFECT FOR ALL CALLS.
                                    1425+*
0001E8                              1426+DAXDPRPM DS    CL24                   STORAGE RESERVED FOR DATA EXIT
                                    1427+*
000200                              1428+DAXRSVD2 DS    CL32                   RESERVED FOR DXT USE
000220                              1429+DAXSCRT1 DS    CL128                  WORK SPACE (SCRATCHPAD)
                                    1430+*                                     MAY BE USED BY EXIT
                                    1431+*                                     ROUTINE AS DESIRED
                                    1432+*
                             002A0  1433+DAXEND   EQU   *                      END OF DAX DSECT
                             002A0  1434+DAXLEN   EQU   *-DAX                  LENGTH OF DAX DSECT
                                    1435+*****************************************************************
                                    1436+*
                                    1437+*    DAXANCTR DSECT   ***DXT ONLY***
                                    1438+*                     MAPS THE ARRAY ELEMENTS OF DAXASEGS
                                    1439+*
                                    1440+*****************************************************************
000000                              1441+DAXANCTR DSECT ,      ***DXT ONLY***
000000                              1442+DAXASGNM DS    CL8    ***DXT ONLY***   ANCESTOR SEGM NAME
                                    1443+*
000008                              1444+DAXASGAD DS    AL4    ***DXT ONLY***   ANCESTOR SEGM ADDRESS
                                    1445+*


                                    1447 *********************************************************************
                                    1448 *       REDEFINITION OF THE MESSAGE AREA LOCATED IN THE DAX        *
                                    1449 *********************************************************************

0002A0                              1451 DAX      DSECT
0002A0                       001A8  1452          ORG   DAXSMESG
0001A8 4040404040404040             1453 MSGID    DC    CL8' '             MESSAGE ID
0001B0 40                           1454 MSGBL1   DC    C' '               ONE BLANK
0001B1 4040404040404040             1455 MSGTXT   DC    CL55' '            TEXT


000000                              1457          END   EKYESE1A
```

Figure 8 (Part 23 of 23). First Sample Segment Exit Routine (Assembler)

Definitions for the First Sample Segment Exit Routine

This section contains definitions associated with the first sample Segment exit routine. The following types of definitions are provided:

- IMS DBDGEN and PSBGEN definitions
- DB2 CREATE TABLE definitions
- DataRefresher definitions required to define the PR with DataRefresher and to extract the IMS data with DataRefresher
- SQL statements defining the PR without DataRefresher in the MVG input tables

DBDGEN Definitions

Figure 9 shows a DBDGEN definition for the Segment exit routine in Figure 8 on page 46.

```
         DBD NAME=DB1,VERSION=V123456789,                              C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),               C
               EXIT=(EKYRUP00)
         DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=SEG1,PARENT=0,BYTES=(101,10)
         FIELD  NAME=(KEY,SEQ,U),BYTES=8,START=3
*
         DBDGEN
         FINISH
         END
```

Figure 9. DBDGEN Definition

Note: The EXIT= keyword of the DBD macro specifies that EKYRUP00 (the RUP) be called when a segment of this DBD is changed. This is required for synchronous data propagation.

PSBGEN Definitions

Figure 10 shows a PSBGEN definition for the Segment exit routine in Figure 8 on page 46.

```
         PCB   TYPE=DB,...                                             C
               ...
         SENSEG ...
         PCB   TYPE=DB,...                                             C
               ...
         SENSEG ...
PCBDPR1  PCB   TYPE=DB,DBDNAME=DB1,LIST=NO                             C
               KEYLEN=101,PROCOPT=A
         SENSEG NAME=SEG1
*
         PSBGEN PSBNAME=PSBDPR1
         END
```

Figure 10. PSBGEN Definition

Note: The first two PCBs represent PCBs used by the application programs. The third PCB, PCBDPR1, is the PCB reserved for HUP usage.

CREATE TABLE Statement

Figure 11 on page 70 shows a CREATE TABLE statement for the segment exit routine in Figure 8 on page 46.

```
    CREATE TABLE T096606.TABLE01
      (KEY1        CHAR(2)     NOT NULL,
       KEY2        CHAR(6)     NOT NULL,
       FAMILY      VARCHAR(30)         ,
       FIRST       VARCHAR(20)         ,
       CITY        VARCHAR(35)         ,
     PRIMARY KEY (KEY1, KEY2))
     DATA CAPTURE CHANGES
     IN DU096606.PROPTS;


CREATE UNIQUE INDEX XN01 ON TABLE01 (KEY1, KEY2)
       USING VCAT KOE ;
```

Figure 11. CREATE TABLE Statement

Note: The DATA CAPTURE CHANGES option of the create table command specifies that the DB2 Changed Data Capture exit (the HUP) be called when a row of this table is changed under IMS attach.

Using DataRefresher to Define the PR

This section shows how to define the PR in Figure 8 on page 46 using DataRefresher.

CREATE DXTPSB

Figure 12 on page 71 shows a CREATE DXTPSB statement for the segment exit routine in Figure 8 on page 46.

```
CREATE DXTPSB   NAME=KOEPSB2

  DXTPCB   NAME=DB1, DBNAME=DB1, DBACCESS=HDAM

  SEGMENT  NAME=SEG1, PARENT=0, BYTES=101,
           DATAEXIT=EKYESE1A, XBYTES=101, FORMAT=V

      FIELD    NAME     = KEY ,
               START    = 3,
               BYTES    = 8,
               SEQFLD   = R
      FIELD    NAME     = KEY1,
               TYPE     = C,
               START    = 3,
               BYTES    = 2
      FIELD    NAME     = KEY2,
               TYPE     = C,
               START    = 5,
               BYTES    = 6
      FIELD    NAME     = LFAMILY,
               TYPE     = H,
               START    = 11,
               BYTES    = 2
      FIELD    NAME     = FAMILY,
               TYPE     = VC,
               LFIELD   = LFAMILY,
               START    = 13,
               BYTES    = 30
      FIELD    NAME     = LFIRST,
               TYPE     = H,
               START    = 43,
               BYTES    = 2
      FIELD    NAME     = FIRST,
               LFIELD   = LFIRST,
               TYPE     = VC,
               START    = 45,
               BYTES    = 20
      FIELD    NAME     = LCITY,
               TYPE     = H,
               START    = 65,
               BYTES    = 2
      FIELD    NAME     = CITY,
               TYPE     = VC,
               LFIELD   = LCITY,
               START    = 67,
               BYTES    = 35;
```

Figure 12. CREATE DXTPSB Statement

Notes:

1. Segment exit routine EKYESE1A is specified on the DATAEXIT= keyword of the SEGMENT statement of CREATE DXTPSB.

   The SEGMENT statement also provides the following specifications:

   - BYTES=101 specifies the maximum length of the segment in its IMS DB format.
   - XBYTES=101 specifies the maximum length of the segment in its DPROP format.
   - FORMAT=V specifies the segment has a variable length in its DPROP format.

2. The FIELD statements describe the fields as they appear in the DPROP format of the segment (as opposed to the segment in its IMS DB format).

   All propagated fields need to be described in a FIELD statement.

3. The fields FAMILY, FIRST, and CITY are defined by TYPE=VC as variable-length character fields.

   DataRefresher requires that each variable-length field have an associated length field. The length fields are described with their own FIELD statements. The LFIELD= keyword of a variable-length field must identify the name of the length field.

   For example, this is illustrated in the FAMILY field. The LFIELD= keyword of the FAMILY field identifies LFAMILY as the length field of FAMILY.

   The EXTRACT statement (see below) propagates the variable-length fields, but does not propagate the length fields.

CREATE DXTVIEW

Figure 13 shows a CREATE DXTVIEW statement for the Segment exit routine in Figure 8 on page 46.

```
CREATE
   DXTVIEW NAME     = VIEW011,
           DXTPSB   = KOEPSB2,
           DXTPCB   = DB1,
           SEGMENT  = SEG1,
           MINSEGM  = SEG1,
           FIELDS   = *     ;
```

Figure 13. CREATE DXTVIEW Statement

DataRefresher UIM SUBMIT Command and EXTRACT Statement

Figure 14 shows a DataRefresher UIM SUBMIT command and EXTRACT statement for the Segment exit routine in Figure 8 on page 46.

```
SUBMIT   EXTID=PR001,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPDIR=TW,
                         MAPCASE=1,
                         ACTION=REPL,
                         ERROPT=BACKOUT,
                         PCBLABEL=PCBDPR1'

EXTRACT
    INTO T096606.TABLE01 (KEY1  NOT NULL,
                          KEY2  NOT NULL,
                          FAMILY,
                          FIRST,
                          CITY)
                  SELECT  KEY1,
                          KEY2,
                          FAMILY,
                          FIRST,
                          CITY
                FROM VIEW011 ;
```

Figure 14. DataRefresher UIM SUBMIT Command and EXTRACT Statement

Notes:

1. The MAPEXIT= keyword of the SUBMIT control statement specifies EKYMCE00. This results in DataRefresher UIM calling the DPROP-provided Map Capture Exit EKYMCE00 during processing of the SUBMIT or EXTRACT. This is needed to allow DPROP to create the PR.
2. MAPUPARM= is used to provide the DPROP propagation keywords.
3. The EXTRACT statement describes to DataRefresher and DPROP which fields must be mapped to which columns.

   The EXTRACT statement propagates the variable-length fields FAMILY, FIRST, and CITY; it does not propagate the length fields LFAMILY, LFIRST, and LCITY.

Using DataRefresher for the Extract

This section covers INITDEM and USE DXTPSB Control Statements. Figure 15 shows INITDEM and USE DXTPSB control statements for the Segment exit routine in Figure 8 on page 46.

```
INITDEM  NAME=DEMPROD;
USE DXTPSB=KOEPSB2;
```

Figure 15. Using DataRefresher for the Extract: INITDEM and USE DXTPSB Control Statements

Defining the PR in the MVG Input Tables

This section shows how to define the PR without using DataRefresher. Figure 16 on page 74 describes the DSNTEP2 SQL statements required to define the PR in the MVG input tables.

The following rows are inserted into the MVG input tables:

- One row is inserted into the DPRIPR table (the PR table).

  This row identifies the PR ID. By inserting an F into the PRTYPE column and a 1 into the MAPCASE column, you can set up the SQL statement so that the PR belongs to mapping case 1 of an extended-function PR.

- One row for the Entity segment Type SEG1 is inserted into the DPRISEG table (the SEG table).

  Because SEG1 is the root segment, no rows are inserted into DPRISEG for physical ancestors.

  The row describing SEG1 provides the following column values:

  – The nonblank value EKYESE1A in the SEGEXIT column. This specifies that the segment must be processed by the Segment exit routine EKYESE1A.

  – The value 101 in the SEGEXITL column specifies the maximum length of the segment in its DPROP format.

  – The value V in the SEGEXITF column specifies that the segment in its DPROP format has a variable length.

- One row is inserted into the DPRITAB table (the TAB table).

This row indicates that the target table is T096606.TABLE01.

- One row is inserted into the DPRIFLD table (the FLD table) for each propagated field.

  The DPRIFLD rows describe the fields as they appear in the DPROP format of the segment (as opposed to the segment in its IMS DB format).

  The fields FAMILY, FIRST, and CITY are defined by the VC value in the DATATYPE column as variable-length character fields.

  DPROP requires two DPRIFLD rows for each variable length field:

  – One row describes the variable-length field.
  – The other row describes the length field.

  The FAMILY field illustrates this. The row describing the variable-length field FAMILY identifies in the LENFIELD column the name of the length field, LFAMILY.

  The row describing the length field LFAMILY has a blank value in the COLNAME column, because the length field is not propagated (only the variable-length field FAMILY is propagated).

```
DELETE FROM T096606.DPRIPR WHERE PRID = 'PR001'           ;


INSERT INTO T096606.DPRIPR
             ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR,
               ERROPT,   ACTION)
      VALUES ('PR001',  'T096606','F',    '1',      'TW',
              'BACKOUT','REPL')                           ;

INSERT INTO T096606.DPRISEG
             ( PRID,     DBNAME,   SEGNAME, ROLE,   PCBLABEL,
               SEGEXIT,  SEGEXITL, SEGEXITF )
      VALUES ('PR001',  'DB1',    'SEG1',  'E',     'PCBDPR1',
              'EKYESE1A',101 ,    'V'                  ;

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL,   TABNAME )
      VALUES ('PR001','T096606', 'TABLE01')               ;

INSERT INTO T096606.DPRIFLD
             ( PRID,     DBNAME,   SEGNAME, FLDNAME,
               TABQUAL,  TABNAME,           COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'KEY1',
              'T096606','TABLE01',         'KEY1',
              'C ',      3,        2)                     ;

INSERT INTO T096606.DPRIFLD
             ( PRID,     DBNAME,   SEGNAME, FLDNAME,
               TABQUAL,  TABNAME,           COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'KEY2',
              'T096606','TABLE01',         'KEY2',
              'C ',      5,        6)                     ;
```

Figure 16 (Part 1 of 2). Defining the PR in the MVG Input Tables

```
INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,           COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'LFAMILY',
             'T096606','TABLE01',          '    ',
             'H ',      11,       2 )                         ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,           COLNAME,
              DATATYPE, POSITION, BYTES,   LENFIELD )
      VALUES ('PR001',  'DB1',    'SEG1',  'FAMILY',
             'T096606','TABLE01',          'FAMILY' ,
             'VC',      13,       30,      'LFAMILY')  ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,           COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'LFIRST ',
             'T096606','TABLE01',          '    ',
             'H ',      43,       2 )                         ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,           COLNAME,
              DATATYPE, POSITION, BYTES,   LENFIELD )
      VALUES ('PR001',  'DB1',    'SEG1',  'FIRST',
             'T096606','TABLE01',          'FIRST'  ,
             'VC',      45,       20,      'LFIRST')          ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,           COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'LCITY  ',
             'T096606','TABLE01',          '    ',
             'H ',      65,       2 )                         ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,           COLNAME,
              DATATYPE, POSITION, BYTES,   LENFIELD )
      VALUES ('PR001',  'DB1',    'SEG1',   'CITY  ',
             'T096606','TABLE01',           'CITY'  ,
             'VC',      67,       35,       'LCITY' )          ;


COMMIT;
```

Figure 16 (Part 2 of 2). Defining the PR in the MVG Input Tables

Second Sample Segment Exit Routine

Figure 17 on page 77 contains another example of a Segment exit routine. This example supports the propagation of an IMS segment containing internal segments propagated by a mapping case 3 PR.

When it receives a changed IMS data segment and is called for IMS-to-DPROP mapping, the exit routine transforms the segment into a DPROP-supported format. During this transformation process, the exit routine creates in each occurrence of the internal segment type an ID field. The ID field is required by DPROP and allows identification of each occurrence of the internal segment within its containing

IMS segment. The exit routine builds a counter field. The counter field describes how many internal segments are contained within a particular occurrence of the containing segment.

When it is called for DPROP-to-IMS mapping, the exit routine must build the IMS format of the segment. The Segment exit routine receives the following input:

- Either a changed occurrence of an internal segment (in its DPROP format), or the changed containing segment (in its DPROP format)
- The existing before-change image of the IMS segment in its IMS format

By combining information from this input, the segment exit routine builds the new after-change image of the IMS segment in its IMS format.

The source code in Figure 17 on page 77 is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYESE2C. Following the source code are definitions related to the sample Segment exit routine.

```
*----------------- START OF SPECIFICATIONS ---------------------*
*                                                              *
* MODULE NAME: EKYESE2C                                        *
* -----------                                                  *
*                                                              *
* DESCRIPTIVE NAME: SAMPLE SEGMENT EXIT COBOL ROUTINE          *
* ----------------                                             *
*                                                              *
* FUNCTION: EKYESE2C IS A SAMPLE DPROP SEGMENT EXIT ROUTINE    *
* --------  WRITTEN IN COBOL AND USED FOR THE TRANSFORMATION   *
*           OF A SEGMENT LAYOUT BETWEEN ITS:                   *
*             - IMS FORMAT AND                                 *
*             - DPROP FORMAT.                                  *
*                                                              *
*        EKYESE2C ILLUSTRATES ONE OF THE MOST TYPICAL USAGE    *
*        OF DPROP SEGMENT USER EXITS: THE SUPPORT OF THE       *
*        PROPAGATION OF AN IMS SEGMENT CONTAINING AN           *
*        INTERNAL SEGMENT / REPEATING GROUP OF FIELDS.         *
*                                                              *
*        THIS SAMPLE SEGMENT EXIT ROUTINE SUPPORTS TYPE=E      *
*        PR'S AND IS THEREFORE CALLED BOTH FOR:                *
*         - IMS-TO-DPROP MAPPING (E.G. DURING IMS-TO-DB2       *
*             PROPAGATION; ALSO DURING DXT-EXTRACTS,           *
*             CCU-PROCESSING AND DLU PROCESSING).              *
*         - DPROP-TO-IMS MAPPING (E.G. DURING DB2-TO-IMS       *
*             PROPAGATION; ALSO DURING CCU PROCESSING AND      *
*             DLU PROCESSING).                                 *
*                                                              *
*        IN THIS EXAMPLE THE PROPAGATED IMS SEGMENT IS A       *
*        BANK ACCOUNT SEGMENT. THE IMS SEGMENT CONSISTS OF     *
*        THE FOLLOWING FIELDS:                                 *
*         - THE ACCOUNT-NBR  (THIS IS THE KEY OF THE SEG)      *
*         - THE CUSTOMER-NAME                                  *
*         - A REPEATING GROUP OF FIELDS WITH THREE OCCURRENCES. *
*           EACH OCCURRENCE OF THE REPEATING GROUP CONTAINS    *
*           INFORMATION ABOUT ONE TYPE OF CREDIT THAT THE      *
*           BANK IS GRANTING. THIS INFORMATION IS:             *
*             - THE CURRENT AMOUNT OF CREDIT GRANTED TO THE    *
*               CUSTOMER/ACCOUNT                               *
*             - THE CREDIT LIMIT FOR THE CUSTOMER/ACCOUNT.     *
*                                                              *
*        THE DATABASE ADMINISTRATOR WANTS TO HAVE A NORMALIZED  *
*        DB2 TABLE DESIGN AND THEREFORE WANTS TO:              *
*                                                              *
*        1) PROPAGATE THE ACCOUNT-NBR AND CUSTOMER-NAME        *
*           TO/FROM THE TABLE CALLED "ACCOUNT":                *
*                                                              *
*           - THIS IS DONE WITH A MAPPING-CASE-1 PR.           *
*                                                              *
*        2) PROPAGATE THE INFORMATION RELATED TO THE           *
*           DIFFERENT TYPES OF CREDITS (TOGETHER WITH THE      *
*           ACCOUNT-NBR) TO/FROM ANOTHER TABLE CALLED "CREDIT": *
*                                                              *
*           - THIS IS DONE WITH A MAPPING-CASE-3 PR.           *
*                                                              *
*           EACH OCCURRENCE OF THE CREDIT INFORMATION (THERE   *
*           ARE 3 OF THEM) IS CONSIDERED TO BE AN OCCURRENCE   *
*           OF AN INTERNAL SEGMENT AND IS PROPAGATED TO/FROM   *
*           ONE ROW OF THE TABLE "CREDIT".                     *
*                                                              *
*           IN ORDER TO DISTINGUISH WITHIN THE CREDIT TABLE    *
*           THE 3 TYPE OF CREDIT INFORMATION (AND IN ORDER     *
*           TO HAVE A DB2 PRIMARY KEY), THE CREDIT TABLE DOES  *
*           NOT ONLY CONTAIN AN ACCOUNT-NBR COLUMN AND THE     *
*           CURRENT CREDIT AMOUNT AND LIMIT.                   *
*           THE CREDIT TABLE CONTAINS ALSO A "TYPE" COLUMN     *
*           WHICH IDENTIFIES THE TYPE OF CREDIT.               *
```

Figure 17 (Part 1 of 11). Second Sample Segment Exit Routine (COBOL)

```
*                                                               *
*       THE SAMPLE SEGMENT EXIT ROUTINE "EKYESE2C" PROVIDES     *
*       LOGIC TO SUPPORT THE PROPAGATION OF THE IMS SEGMENT     *
*       TO/FROM THE TABLES ACCOUNT AND CREDIT.                  *
*                                                               *
*       1) FOR IMS-TO-DPROP MAPPING, THE SAMPLE EXIT PROVIDES   *
*          THE FOLLOWING FUNCTIONS WHEN BUILDING THE DPROP      *
*          FORMAT OF THE SEGMENT:                               *
*                                                               *
*          - THE EXIT ROUTINE CREATES IN THE DPROP FORMAT       *
*            AN ID-FIELD FOR EACH OCCURRENCE OF THE INTERNAL    *
*            SEGMENT. THIS IS THE FIELD CALLED "TYPE".          *
*                                                               *
*            THIS ADDRESS THE DPROP REQUIREMENT THAT INTERNAL   *
*            SEGMENTS HAVE AN "ID" FIELD IDENTIFYING UNIQUELY   *
*            THE OCCURRENCES OF THE INTERNAL SEGMENTS WITHIN    *
*            THE CONTAINING SEGMENT.                            *
*                                                               *
*            IN THE DPROP FORMAT, EACH OCCURRENCE OF THE IN-    *
*            TERNAL SEGMENT WILL CONSIST OF FOLLOWING FIELDS:   *
*            - THE FIELD "TYPE" (THIS IS THE ID-FIELD           *
*                CREATED BY THE EXIT)                           *
*            - THE FIELD "AMOUNT" (COPIED FROM THE IMS          *
*                FORMAT OF THE SEGMENT)                         *
*            - THE FIELD "LIMIT"  (COPIED FROM THE IMS          *
*                FORMAT OF THE SEGMENT).                        *
*                                                               *
*          - THE EXIT ROUTINE CREATES IN THE DPROP FORMAT       *
*            A COUNT FIELD. ITS VALUE IS THE NUMBER OF          *
*            OCCURRENCES OF THE INTERNAL SEGMENT-TYPE WITHIN    *
*            THE CONTAINING SEGMENT.                            *
*            NOTE THAT A COUNT FIELD IS REQUIRED BY DPROP       *
*            FOR THE PROPAGATION OF INTERNAL SEGMENTS WITH      *
*            TYPE=E PR'S.                                       *
*                                                               *
*       2) FOR DPROP-TO-IMS MAPPING, THE SAMPLE EXIT            *
*          DISTINGUISHES THE TWO FOLLOWING CASES:               *
*                                                               *
*          A) IT IS CALLED DURING A REPLACE, DELETE, OR         *
*             INSERT OF A ROW OF THE "CREDIT" TABLE.            *
*                                                               *
*             IN THIS CASE, THE EXIT ROUTINE GETS FOLLOWING     *
*             TWO INPUTS FROM DPROP:                            *
*                                                               *
*             - THE CHANGED OCCURRENCE OF THE INTERNAL SEGMENT *
*               IN ITS DPROP FORMAT.                            *
*               THIS INPUT HAS BEEN BUILT BY DPROP BY MAPPING   *
*               THE CHANGED CREDIT ROW TO THE DPROP FORMAT      *
*               OF THE INTERNAL SEGMENT.                        *
*                                                               *
*             - THE EXISTING "BEFORE-CHANGE" IMS SEGMENT IN     *
*               ITS IMS FORMAT.                                 *
*                                                               *
*             THE SEGMENT EXIT ROUTINE IS RESPONSIBLE BY        *
*             COMBINING INFORMATION IN THESE TWO INPUTS TO      *
*             BUILD THE NEW "AFTER-CHANGE" IMS SEGMENT IN       *
*             ITS IMS FORMAT.                                   *
*                                                               *
*          B) IT IS CALLED DURING A REPLACE, DELETE, OR         *
*             INSERT OF A ROW OF THE "ACCOUNT TABLE"            *
*                                                               *
*             IN THIS CASE, THE EXIT ROUTINE GETS FOLLOWING     *
*             TWO INPUTS FROM DPROP:                            *
*                                                               *
```

Figure 17 (Part 2 of 11). Second Sample Segment Exit Routine (COBOL)

```
*            - THE CHANGED OCCURRENCE OF THE CONTAINING      *
*              SEGMENT IN ITS DPROP FORMAT.                  *
*              THIS INPUT HAS BEEN BUILT BY DPROP BY MAPPING *
*              THE CHANGED ACCOUNT ROW TO THE DPROP FORMAT   *
*              OF THE CONTAINING SEGMENT.                    *
*                                                            *
*            - THE EXISTING "BEFORE-CHANGE" IMS SEGMENT IN   *
*              ITS IMS FORMAT (ONLY FOR REPLACES AND         *
*              DELETES OF ROWS OF THE ACCOUNT TABLE).        *
*                                                            *
*            THE SEGMENT EXIT ROUTINE IS RESPONSIBLE BY      *
*            COMBINING INFORMATION IN THESE TWO INPUTS TO    *
*            BUILD THE NEW "AFTER-CHANGE" IMS SEGMENT IN     *
*            ITS IMS FORMAT.                                 *
*                                                            *
/*                                                           *
*                                                            *
*      THE FIGURE BELOW DESCRIBES ON THE LEFT-HAND SIDE      *
*      THE SEGMENT IN ITS IMS FORMAT AND ON THE RIGHT-HAND   *
*      SIDE THE SEGMENT IN ITS DPROP FORMAT.                 *
*                                                            *
*                                                            *
*       *--------------------*    *---------------------*    *
*       | IMS SEGMENT IN ITS |    |  IMS SEGMENT IN ITS |    *
*       | IMS FORMAT         |    |  DPROP FORMAT       |    *
*       *--------------------*    *---------------------*    *
*                                                            *
*       *--------*-----*-----*    *--------*-----*------*    *
*       |FLD NAME| FLD | FLD |    |FLD NAME| FLD | FLD  |    *
*       |        | FMT |START|    |        | FMT |START |    *
*       *--------*-----*-----*    *--------*-----*------*    *
*       |ACNT_NBR|  C  |  1  |<-->|ACNT_NBR|  C  |  1   |    *
*       |NAME    |  C  |  10 |<-->|NAME    |  C  |  10  |    *
*       |        |     |     |    |COUNT   |  H  |  31  |    *
*       |        |     |     |    |TYPE_1  |  P  |  33  |    *
*       |AMOUNT_A|  P  |  31 |    |AMOUNT_1|  P  |  34  |    *
*       |LIMIT_A |  P  |  38 |    |LIMIT_1 |  P  |  41  |    *
*       |        |     |     |    |TYPE_2  |  P  |  48  |    *
*       |AMOUNT_B|  P  |  45 |    |AMOUNT_2|  P  |  49  |    *
*       |LIMIT_B |  P  |  52 |    |LIMIT_2 |  P  |  56  |    *
*       |        |     |     |    |TYPE_3  |  P  |  63  |    *
*       |AMOUNT_C|  P  |  59 |    |AMOUNT_3|  P  |  64  |    *
*       |LIMIT_C |  P  |  66 |    |LIMIT_3 |  P  |  71  |    *
*       *--------*-----*-----*    *--------*-----*------*    *
*                                                            *
*      BOTH THE IMS FORMAT AND THE DPROP FORMAT OF THE       *
*      IMS SEGMENT ARE DEFINED AS FIXED-LENGTH.              *
*                                                            *
*      THE INTERNAL SEGMENT IS DEFINED TO DPROP AS FOLLOWS:  *
*                                                            *
*       - IT HAS A VARIABLE NUMBER OF OCCURRENCES.           *
*         (THE NUMBER OF OCCURRENCES IS IN THE COUNT FELD    *
*         "COUNT" OF THE CONTAINING SEGMENT)                 *
*       - THE FIRST OCCURRENCE STARTS AT A FIXED LOCATION    *
*         WITHIN THE CONTAINING SEGMENT (START=33)           *
*       - IT HAS A FIXED LENGTH (15 BYTES)                   *
*       - IT CONSISTS OF THE FOLLOWING FIELDS:               *
*         THE 1-BYTE TYPE, 7-BYTES AMOUNT, AND 7-BYTES LIMIT *
*                                                            *
*      PLEASE REFER TO THE DSECTS TOWARDS THE BOTTOM OF THIS *
*      MODULE IN ORDER TO FIND ALL THE DETAILS ABOUT THE     *
*      "IMS FORMAT" AND THE "DPROP FORMAT" OF THE SEGMENTS   *
*                                                            *
```

Figure 17 (Part 3 of 11). Second Sample Segment Exit Routine (COBOL)

```
*       FOLLOWING CONVENTIONS ARE USED TO DESCRIBE CREDIT-INFO *
*       WHICH DO NOT EXIST:                                    *
*        - IN THE IMS FORMAT, A NON-EXISTING CREDIT-INFO       *
*          HAS A ZERO VALUE IN THE FIELD "LIMIT".              *
*        - IN THE DPROP FORMAT, THE COUNT REFLECTS THE NUMBER  *
*          OF EXISTING CREDIT INTERNAL SEGMENTS. EXISTING      *
*          CREDIT INTERNAL SEGMENTS FOLLOW EACH-OTHER IN THE   *
*          DPROP FORMAT OF THE IMS SEGMENT (NON-EXISTING       *
*          INTERNAL SEGMENTS ARE ELIMINATED.                   *
*          THIS MUST BE SO IN ORDER TO CONFORM TO THE WAY THAT *
*          INTERNAL SEGMENTS ARE DEFINED TO DPROP AND DXT.     *
*                                                              *
*                                                              *
* INPUT:   1ST PARAMETER: ADDRESS OF DAX (DAX IS THE EXIT      *
* -----                   INTERFACE CONTROL BLOCK)             *
*          2ND PARAMETER: ADDRESS OF SEGMENT IN IMS FORMAT     *
*          3RD PARAMETER: ADDRESS OF SEGMENT IN DPROP FORMAT   *
*          4TH PARAMETER: ADDRESS OF ANCHOR AREA PRESERVED     *
*                         ACROSS CALLS TO THIS EXIT.           *
*                                                              *
* OUTPUT:  THE SEGMENT FORMAT TRANSFORMATION HAS BEEN DONE     *
* ------                                                       *
*                                                              *
* EXIT-ERROR=                                                  *
*    RETURN CODE  = 12: MAPPING PROBLEM / INVALID DATA         *
*                 = 16: SHOULD-NOT-OCCUR ERRORS                *
*                       (INVALID CALL FUNCTION,                *
*                        PARAMETER AREA TOO SMALL,             *
*                        INVALID SEGMENT NAME).                *
*                                                              *
*                                                              *
* ERROR MESSAGES ISSUED BY EKYESE2C                            *
*         EKYESE1E: CALL FUNCTION NOT SUPPORTED                *
*         EKYESE2E: UNSUPPORTED DBD OR SEGNAME                 *
*         EKYESE3E: UNEXPECTED LENGTH OF IMS SEGMENT           *
*         EKYESE4E: DPROP SEGMENT IS TOO SHORT                 *
*         EKYESE5E: IMS    SEGMENT IS TOO SHORT                *
*         EKYESE6E: UNEXPECTED VALUE IN TYPE COLUMN OF         *
*                      CREDIT TABLE                            *
*                                                              *
* CHANGE ACTIVITY= NONE                                        *
*                                                              *
*----------------- END OF SPECIFICATIONS -----------------------*
/*
*----------------- LOGIC OF EKYESE2C ---------------------------*
*                                                              *
*  MAIN-LINE LOGIC:                                            *
*  ================                                            *
*                                                              *
*  1) MODULE ENTRY LOGIC:                                      *
*  ----------------------                                      *
*                                                              *
*     - SET "MODULE ENTERED" AND "MODULE IN CONTROL" FLAGS     *
*       INTO DAX.                                              *
*                                                              *
*     - VERIFY THAT THE EXIT IS INVOKED TO PROPAGATE THE       *
*       CORRECT DATABASE AND SEGMENT                           *
*                                                              *
*     - BRANCH ACCORDING TO CALL FUNCTION EITHER FOR:          *
*       - THE PROCESSING OF IMS-TO-DPROP, OR                   *
*       - THE PROCESSING OF DPROP-TO-IMS                       *
*                                                              *
```

Figure 17 (Part 4 of 11). Second Sample Segment Exit Routine (COBOL)

```
*  2) IMS-TO-DPROP FORMATTING                                     *
*  --------------------------                                     *
*                                                                 *
*     - CHECK LENGTH OF SEGMENT IN ITS IMS FORMAT AND             *
*       CHECK THAT DPROP SEGMENT BUFFER IS LARGE ENOUGH           *
*                                                                 *
*     - MOVE TO DPROP FORMAT THE ACCOUNT-NBR AND THE              *
*       CUSTOMER NAME.                                            *
*                                                                 *
*     - INITIALIZE THE NUMBER OF INTERNAL SEGMENT OCCURRENCES     *
*       TO ZERO.                                                  *
*                                                                 *
*     - FOR EACH NON-ZERO LIMIT IN THE IMS FORMAT:                *
*                                                                 *
*       - INCREASE THE OCCURRENCE COUNTERS BY 1                   *
*       - CREATE IN THE DPROP BUFFER THE ID OF THE INTERNAL       *
*         SEGMENT.                                                *
*       - MOVE TO THE DPROP BUFFER THE DATA OF THE INTERNAL       *
*         SEGMENT.                                                *
*                                                                 *
*     NOTE: A LIMIT WITH A ZERO VALUE IN THE IMS FORMAT IS        *
*     ----  CONSIDERED TO IDENTIFY A "NON-EXISTING" CREDIT        *
*           INFORMATION.                                          *
*                                                                 *
*           IN THE DPROP FORMAT THERE WILL BE NO OCCURRENCE OF    *
*           INTERNAL SEGMENTS FOR THESE NON-EXISTING CREDITS.     *
*           AS REQUIRED BY DPROP, THE OCCURRENCES FOR THE         *
*           EXISTING INTERNAL SEGMENTS WILL FOLLOW EACH OTHER.    *
*                                                                 *
*                                                                 *
*  3) DPROP-TO-IMS-FORMATTING                                     *
*  --------------------------                                     *
*                                                                 *
*     - CHECK THAT IMS SEGMENT BUFFER IS LARGE ENOUGH             *
*                                                                 *
*     - INITIALIZE IMS SEGMENT BUFFER AS FOLLOWS:                 *
*                                                                 *
*       - IF BEFORE-CHANGE IMAGE IS PROVIDED BY THE CALLER,       *
*             COPY THE BEFORE-CHANGE IMAGE TO IMS BUFFER          *
*       - ELSE INITIALIZE IMS BUFFER WITH PROPER INITIAL          *
*             VALUES (ZEROES AND BLANKS).                         *
*                                                                 *
*     - IF PROCESSING THE CHANGE TO THE TARGET OF THE             *
*          CONTAINING SEGMENT:                                    *
*                                                                 *
*       - COPY INFORMATION OF CHANGED CONTAINING SEGMENT FROM     *
*         DPROP BUFFER TO IMS BUFFER.                             *
*                                                                 *
*     - IF PROCESSING THE CHANGE TO THE TARGET OF AN INTERNAL     *
*          SEGMENT:                                               *
*                                                                 *
*       - IF PROCESSING A DELETE,                                 *
*            SET APPROPRIATE CREDIT INFO TO 0 IN THE IMS BUFFER   *
*       - IF PROCESSING A REPLACE OR INSERT,                      *
*            COPY INFORMATION OF CHANGED INTERNAL SEGMENT         *
*            FROM DPROP BUFFER TO IMS BUFFER.                     *
*                                                                 *
*                                                                 *
*  ERROR LOGIC                                                    *
*  ===========                                                    *
*                                                                 *
*     - FORMAT AN ERROR MESSAGE INTO DAX                          *
*     - SET RETURN CODE INTO DAX                                  *
*     - RETURN TO THE CALLER.                                     *
*                                                                 *
*---------------- END OF LOGIC ----------------------------------*
```

Figure 17 (Part 5 of 11). Second Sample Segment Exit Routine (COBOL)

```
/*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. EKYESE2C.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 77      X1            PIC S9(8) COMP.
*                INDEX FOR INTERNAL SEGMENTS IN IMS FORMAT
 77      X2            PIC S9(8) COMP.
*                INDEX FOR INTERNAL SEGMENTS IN DPROP FORMAT
 77      IMSSEGL       PIC S9(8) COMP  VALUE +72.
*                IMS SEGMENT LENGTH
 77      DPRSEGL       PIC S9(8) COMP  VALUE +77.
*                DPR SEGMENT LENGTH
*
*-----------------------------------------------------------------*
*  REDEFINITION OF THE MESSAGE AREA LOCATED IN THE DAX            *
*-----------------------------------------------------------------*
*
 01      MSGLINE.
  02     MSGID         PIC X(11).
  02     MSGBL1        PIC X.
  02     MSGTXT        PIC X(52).
*
*-----------------------------------------------------------------*
*  WORK AREA FOR THE IMS SEGMENT IN ITS DPROP-FORMAT              *
*-----------------------------------------------------------------*
*
 01      DPRSEG.
*
  02     DPRACNBR      PIC X(9).
*                              ACCOUNT NUMBER (KEY)
  02     DPRNAME       PIC X(21).
*                              NAME
  02     DPRCOUNT      PIC 9(4)   COMP.
*                              COUNT INTERNAL SEGM OCCURENCES
  02     DPRINSEG          OCCURS 3.
*                              3 OCCURRENCES OF INTERNAL SEG
   03    DPRTYPE       PIC 9      COMP-3.
*                              ID
   03    DPRAMOUN      PIC 9(11)V99  COMP-3.
*                              CURRENT AMOUNT
   03    DPRLIMIT      PIC 9(11)V99  COMP-3.
*
*-----------------------------------------------------------------*
*  LINKAGE SECTION                                                *
*-----------------------------------------------------------------*
*
 LINKAGE SECTION.
*
*-----------------------------------------------------------------*
*  DESCRIPTION OF THE SEGMENT EXIT INTERFACE "DAX"                *
*-----------------------------------------------------------------*
*
    COPY EKYRCDXC.
*
*-----------------------------------------------------------------*
*  DESCRIPTION OF IMS SEGMENT IN ITS IMS FORMAT                   *
*-----------------------------------------------------------------*
*
 01      IMSSEG.
*
```

Figure 17 (Part 6 of 11). Second Sample Segment Exit Routine (COBOL)

```
  02     IMSACNBR     PIC X(9).
*                                 ACCOUNT NUMBER (KEY)
  02     IMSNAME      PIC X(21).
*                                 NAME OF CUSTOMER
  02     IMSINSEG         OCCURS 3.
*                                 3 OCCURRENCES OF INTERNAL SEG
   03    IMSAMOUN     PIC 9(11)V99 COMP-3.
*                                 CURRENT AMOUNT TYPE-A CREDIT
   03    IMSLIMIT     PIC 9(11)V99 COMP-3.
*
*---------------------------------------------------------------*
*  THE THIRD PARAMETER CAN POINT TO DPRSEG OR TO DPRISEG       *
*---------------------------------------------------------------*
*
 01      THIRDPARM    PIC X(77).
 01      CONTAINING   REDEFINES  THIRDPARM   PIC X(32).
 01      INTERNAL     REDEFINES  THIRDPARM   PIC X(45).
*
*---------------------------------------------------------------*
*  DSECT FOR THE BEFORE_CHANGE IMS IMAGE                       *
*---------------------------------------------------------------*
*
 01      IMSBEFIM     PIC X(72).
*
*---------------------------------------------------------------*
* PROCEDURE DIVISION                                            *
*---------------------------------------------------------------*
*
  PROCEDURE DIVISION USING DAX,
                           IMSSEG,
                           THIRDPARM.
*
*---------------------------------------------------------------*
*  SET THE "EXIT ENTERED" AND "EXIT IN CONTROL" FLAGS.         *
*---------------------------------------------------------------*
*
     MOVE "X"  TO DAXENTRD.
     MOVE "X"  TO DAXINCTL.
     MOVE ZERO TO DAXRETC.
*
*---------------------------------------------------------------*
*  VERIFY THAT THE EXIT IS CALLED TO FORMAT THE EXPECTED       *
*  IMS DATABASE AND SEGMENT TYPE                               *
*---------------------------------------------------------------*
*
     IF  DAXDBNM NOT = "DB123"
         GO TO INVDBSEG.
     IF  DAXSEGM NOT = "ACCOUNT"
         GO TO INVDBSEG.
*
*---------------------------------------------------------------*
*  BRANCH ACCORDING TO CALL-FUNCTION                           *
*---------------------------------------------------------------*
*
     IF  DAXCALL = "NO"
         GO TO IMSTDPR.
*                            "NORMAL CALL" (IMS TO DPROP)
     IF  DAXCALL = "RV"
         GO TO DPRTIMS.
*                            "REVERSE CALL" (DPROP TO IMS)
     GO TO INVCALL.
*                            UNSUPPORTED CALL FUNCTION
*
```

Figure 17 (Part 7 of 11). Second Sample Segment Exit Routine (COBOL)

```
*##############################################################*
*#  NORMAL CALL TO TRANSFORM THE SEGMENT FROM ITS             #*
*#    IMS FORMAT INTO ITS DPROP FORMAT                        #*
*##############################################################*
*
 IMSTDPR.
*
*--------------------------------------------------------------*
*  CHECK THE LENGTH OF SEGMENT IN ITS IMS FORMAT AND CHECK      *
*  THAT THE DPROP BUFFER IS LARGE ENOUGH TO CONTAIN THE         *
*  SEGMENT IN ITS DPROP FORMAT.                                 *
*--------------------------------------------------------------*
*
     IF  DAXDLEN NOT = IMSSEGL
         GO TO INVLENN1.
     IF  DAXFLEN < DPRSEGL
         GO TO INVLENN2.
*
*--------------------------------------------------------------*
*  MOVE THE ACCOUNT NUMBER AND CUSTOMER NAME TO DPROP FORMAT    *
*--------------------------------------------------------------*
*
     MOVE IMSACNBR TO DPRACNBR.
     MOVE IMSNAME  TO DPRNAME.
*
*--------------------------------------------------------------*
*  INITIALIZE PROCESSING FOR THE THREE CREDITS:                 *
*    ---> INITIALIZE COUNTER FIELD TO ZERO                      *
*--------------------------------------------------------------*
*
     MOVE ZERO TO DPRCOUNT.
     MOVE ZERO TO X1, X2.
     MOVE ZERO TO DPRTYPE  (1), DPRTYPE  (2), DPRTYPE  (3).
     MOVE ZERO TO DPRAMOUN (1), DPRAMOUN (2), DPRAMOUN (3).
     MOVE ZERO TO DPRLIMIT (1), DPRLIMIT (2), DPRLIMIT (3).
*                            INIT INDEXES FOR INTERNAL SEGMENTS
     PERFORM MOVECRED 3 TIMES.
*                            MOVE 1, 2 OR 3 CREDITS.
     MOVE DPRSEG TO THIRDPARM.
*                            RETURN IMS SEGMENT IN DPROP FORMAT
     GO TO ENDPGM.
*
*--------------------------------------------------------------*
*  MOVE THE OCCURRENCE OF THE INTERN SEG FOR TYPE_A CREDITS.    *
*--------------------------------------------------------------*
*
 MOVECRED.
     ADD +1 TO X1.
*                            INCREMENT INDEX FOR NEXT INT SEG
     IF  IMSLIMIT (X1) = ZERO
         NEXT SENTENCE
*                            SKIP IF THIS FIELD IS ZERO
     ELSE
         ADD +1  TO X2
         MOVE X2 TO DPRCOUNT
*                            INCREMENT COUNTER OF INTERNAL SEGS
         MOVE X1           TO DPRTYPE  (X2)
         MOVE IMSAMOUN (X1) TO DPRAMOUN (X2)
         MOVE IMSLIMIT (X1) TO DPRLIMIT (X2).
*                            SET ID, MOVE AMOUNT AND LIMIT
 ENDMOVEC.
*
*##############################################################*
*#  REVERSE CALL TO TRANSFORM THE SEGMENT FROM ITS            #*
*#    DPROP FORMAT INTO ITS IMS FORMAT                        #*
*##############################################################*
```

Figure 17 (Part 8 of 11). Second Sample Segment Exit Routine (COBOL)

```
*
 DPRTIMS.
*
*---------------------------------------------------------------*
*  CHECK THAT THE IMS BUFFER IS LARGE ENOUGH TO CONTAIN        *
*    THE SEGMENT IN ITS IMS FORMAT.                             *
*---------------------------------------------------------------*
*
     IF  DAXDLEN < IMSSEGL
         GO TO INVLENN3.
*
*---------------------------------------------------------------*
*  INITIALIZE THE AFTER_CHANGE IMS FORMAT AS FOLLOWS:           *
*                                                               *
*  IF  BEFORE-CHANGE IMAGE OF IMS SEGMENT HAS BEEN PROVIDED     *
*      INIT THE AFTER-CHANGE IMAGE WITH BEFORE_CHANGE IMAGE     *
*  ELSE INIT THE AFTER-CHANGE IMAGE WITH PROPER INITIAL VALUES  *
*---------------------------------------------------------------*
*
     IF  DAXIDDSB = NULL
         GO TO CALLR020.
*                            BEFORE-CHANGE IMAGE IS NOT PROVIDED
*
***      INITIALIZE AFTER-CHANGE IMAGE WITH BEFORE-CHANGE VALUES
*
     SET ADDRESS OF IMSBEFIM TO DAXIDDSB.
*                            ADDRESSING OF BEFORE_CHANGE IMAGE
     MOVE IMSBEFIM TO IMSSEG.
*                            MOVE BEFORE_CHANGE TO AFTER-CH.
     GO TO CALLR100.
*
***      INITIALIZE AFTER-CHANGE IMAGE WITH PROPER INITIAL VALUES
*
 CALLR020.
     MOVE "000000000" TO IMSACNBR.
     MOVE SPACES      TO IMSNAME.
     MOVE ZERO        TO IMSAMOUN (1), IMSLIMIT (1).
     MOVE ZERO        TO IMSAMOUN (2), IMSLIMIT (2).
     MOVE ZERO        TO IMSAMOUN (3), IMSLIMIT (3).
*
*---------------------------------------------------------------*
*  DETERMINE WHETHER WE ARE CALLED FOR A CHANGE TO THE         *
*  ACCOUNT TABLE OR TO THE CREDIT TABLE.                        *
*---------------------------------------------------------------*
*
 CALLR100.
     IF  DAXSEGTI
         GO TO CALLR200.
*                            UPDATE OF INTERNAL SEGMENT
*
*---------------------------------------------------------------*
*  EXIT ROUTINE IS CALLED FOR DPROP-TO-IMS MAPPING  BECAUSE     *
*  THE TARGET OF THE CONTAINING SEGMENT HAS CHANGED.            *
*  WE WILL JUST MOVE INFORMATION FROM THE CONTAINING SEGMENT    *
*  IN ITS DPROP FORMAT TO SEGMENT IN ITS IMS FORMAT             *
*---------------------------------------------------------------*
*
     MOVE CONTAINING TO DPRSEG.
*                            GET CONTAINING SEG IN DPROP FORMAT
     MOVE DPRACNBR TO IMSACNBR.
     MOVE DPRNAME  TO IMSNAME.
     GO TO ENDPGM.
*
```

Figure 17 (Part 9 of 11). Second Sample Segment Exit Routine (COBOL)

```
*--------------------------------------------------------------*
*  EXIT ROUTINE IS CALLED FOR DPROP-TO-IMS MAPPING BECAUSE      *
*  THE TARGET OF THE INTERNAL SEGMENT HAS CHANGED.              *
*                                                               *
*  IF  PROCESSING A DELETE                                      *
*      THE EXIT ROUTINE WILL ZERO THE APPROPR. AMOUNT AND LIMIT *
*  IF  PROCESSING AN INSERT OR REPLACE                          *
*      THE EXIT ROUTINE WILL COPY THE AMOUNT AND LIMIT FROM THE *
*      CHANGED INTERNAL SEGMENT TO THE IMS FORMAT OF THE SEGMENT*
*--------------------------------------------------------------*
*
 CALLR200.
*
*--------------------------------------------------------------*
*  DETERMINE WHICH INTERNAL SEGMENT OCCURRENCE HAS CHANGED      *
*--------------------------------------------------------------*
*
     MOVE INTERNAL TO DPRINSEG (1).
*                             GET INTERNAL SEG IN DPROP FORMAT
     IF  DPRTYPE (1) = 0 OR 1 OR 2 OR 3
         MOVE DPRTYPE (1) TO X2
         GO TO CALLR210
*                             CHANGE OF 1ST, 2ND OR 3RD TYPE
     ELSE
         GO TO INVTYPE.
*                             INVALID TYPE
*
*--------------------------------------------------------------*
*  BRANCH DEPENDING ON THE TYPE OF UPDATE                       *
*--------------------------------------------------------------*
*
 CALLR210.
     IF  DAXDPRCT NOT = "DLET"
         GO TO CALLR230.
*
***      A DELETE: ZERO CREDIT INFO IN IMS FORMAT
*
     MOVE 0 TO IMSAMOUN (X2).
     MOVE 0 TO IMSLIMIT (X2).
     GO TO ENDPGM.
*
***  INSERT OR REPLACE: COPY CHANGED CREDIT INFO INTO IMS FORMAT
*
 CALLR230.
     MOVE DPRAMOUN (1) TO IMSAMOUN (X2).
     MOVE DPRLIMIT (1) TO IMSLIMIT (X2).
     GO TO ENDPGM.
*
*--------------------------------------------------------------*
*  ERROR LOGIC:                                                 *
*     - BUILD IN THE INTERFACE CONTROL BLOCK AN ERROR MESSAGE   *
*     - SET A RETURN CODE IN THE INTERFACE CONTROL BLOCK        *
*     - RETURN TO CALLER OF THE EXIT                            *
*--------------------------------------------------------------*
*
 INVCALL.
     MOVE "EKYESE1E" TO MSGID.
     MOVE SPACE TO MSGBL1.
     MOVE "CALL FUNCTION NOT SUPPORTED"
          TO MSGTXT.
     GO TO INVRC16.
*
```

Figure 17 (Part 10 of 11). Second Sample Segment Exit Routine (COBOL)

```
 INVDBSEG.
     MOVE "EKYESE2E" TO MSGID.
     MOVE SPACE TO MSGBL1.
     MOVE "UNSUPPORTED DBD OR SEGNAME"
          TO MSGTXT.
     GO TO INVRC16.
*
 INVLENN1.
     MOVE "EKYESE3E" TO MSGID.
     MOVE SPACE TO MSGBL1.
     MOVE "UNEXPECTED LENGTH OF IMS SEGMENT"
          TO MSGTXT.
     GO TO INVRC16.
*
 INVLENN2.
     MOVE "EKYESE4E" TO MSGID.
     MOVE SPACE TO MSGBL1.
     MOVE "DPROP SEGMENT BUFFER IS TOO SHORT"
          TO MSGTXT.
     GO TO INVRC16.
*
 INVLENN3.
     MOVE "EKYESE5E" TO MSGID.
     MOVE SPACE TO MSGBL1.
     MOVE "IMS SEGMENT BUFFER IS TOO SHORT"
          TO MSGTXT.
     GO TO INVRC16.
*
 INVTYPE.
     MOVE "EKYESE6E" TO MSGID.
     MOVE SPACE TO MSGBL1.
     MOVE "UNEXPECTED VALUE IN TYPE COLUMN OF CREDIT TABLE"
          TO MSGTXT.
     GO TO INVRC12.
*
 INVRC12.
     MOVE MSGLINE TO DAXSMESG.
     MOVE 16 TO DAXRETC.
*                            SET RETURN CODE 12 (ERROR)
     GO TO ENDPGM.
*
 INVRC16.
     MOVE MSGLINE TO DAXSMESG.
     MOVE 12 TO DAXRETC.
*                            SET RETURN CODE 16 (SEVERE ERROR)
     GO TO ENDPGM.
*
*---------------------------------------------------------------*
*  RETURN TO CALLER OF THIS EXIT                                *
*---------------------------------------------------------------*
*
 ENDPGM.
     GOBACK.
*
*---------------------------------------------------------------*
```

Figure 17 (Part 11 of 11). Second Sample Segment Exit Routine (COBOL)

Definitions for the Second Sample Segment Exit Routine

This section contains definitions associated with the second sample Segment exit routine. The following types of definitions are provided:

- IMS DBDGEN and PSBGEN definitions
- DB2 CREATE TABLE definitions
- DataRefresher definitions required to define the PR with DataRefresher and to extract the IMS data with DataRefresher
- SQL statements required to define the PR in the MVG Input Tables without DataRefresher

DBDGEN Definitions

Figure 18 show a DBDGEN definition for the Segment exit routine in Figure 17 on page 77.

```
         DBD NAME=DB123,VERSION=V12,                                 C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),             C
               EXIT=(EKYRUP00)
         DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=ACCOUNT,PARENT=0,BYTES=72
         FIELD NAME=(ACNTNBR,SEQ,U),BYTES=9,START=1
*
         DBDGEN
         FINISH
         END
```

Figure 18. DBDGEN Definition

Note: The EXIT= keyword of the DBD Macro specifies that EKYRUP00 (the RUP) be called when a segment of this DBD is changed. This is required for synchronous data propagation.

PSBGEN Definitions

Figure 19 shows a PSBGEN definition for the Segment exit routine in Figure 17 on page 77.

```
         PCB   TYPE=DB,...                                           C
               ...
         SENSEG ...
         PCB   TYPE=DB,...                                           C
               ...
         SENSEG ...
         PCB   TYPE=DB,DBDNAME=DB123,NAME=HUPPCB,                    C
               KEYLEN=72,PROCOPT=A
         SENSEG NAME=ACCOUNT
*
         PSBGEN PSBNAME=PSBDPR3
         END
```

Figure 19. PSBGEN Definition

Note: The first two PCBs represent PCBs used by the application programs. The third PCB, HUPPCB, is the PCB reserved for HUP usage.

CREATE TABLE Statements

Figure 20 contains the CREATE TABLE statements used to create the ACCOUNT table and the CREDIT table in Figure 17 on page 77.

The figure contains the CREATE UNIQUE INDEX statements required to create the indexes for the DB2 primary keys of the two tables.

```
CREATE TABLE ACCOUNT
          (ACT_NBR   CHAR(9)        NOT NULL,
           NAME      CHAR(21)       NOT NULL WITH DEFAULT,
       PRIMARY KEY  (ACT_NBR))
       DATA CAPTURE CHANGES
       IN DU096606.PROPT1    ;

CREATE UNIQUE INDEX XN01 ON ACCOUNT
          (ACT_NBR)
       USING VCAT KOE ;



CREATE TABLE CREDIT
          (ACT_NBR   CHAR(9)         NOT NULL,
           TYPE      DECIMAL (1,0)   NOT NULL,
           AMOUNT    DECIMAL (13,2)  NOT NULL WITH DEFAULT,
           LIMIT     DECIMAL (13,2)  NOT NULL WITH DEFAULT,
       PRIMARY KEY (ACT_NBR,TYPE),
       FOREIGN KEY (ACT_NBR) REFERENCES ACCOUNT ON DELETE CASCADE)
       DATA CAPTURE CHANGES
       IN DU096606.PROPT2    ;

CREATE UNIQUE INDEX XN02 ON CREDIT
          (ACT_NBR, TYPE)
       USING VCAT KOE ;
```

Figure 20. CREATE TABLE Statements

Note: The DATA CAPTURE CHANGES option of the CREATE TABLE command specifies that the DB2 Changed Data Capture exit (the HUP) be called when a row of this table is changed under IMS attach. The FOREIGN KEY option is used if one-way DB2-to-IMS propagation or two-way propagation is implemented. The containing segment/internal segment relationship should be handled just as a parent/child segment is handled for setting up matching RIRs. In this example, the DELETE CASCADE option is used.

Using DataRefresher To Define the PR: CREATE DXTPSB

Figure 21 on page 90 shows a CREATE DXTPSB definition for the Segment exit routine in Figure 17 on page 77.

```
CREATE DXTPSB    NAME=KOEPSB2

       DXTPCB    NAME=PCB001,DBACCESS=HDAM,DBNAME=DB123

         SEGMENT NAME=ACCOUNT , PARENT=0, BYTES=72 ,
                 EXIT=EKYESE2C, XBYTES=77

             FIELD    NAME=KEY    , START=1,  BYTES=9 , SEQFLD=R
             FIELD    NAME=ACT_NBR, START=1,  BYTES=9 , TYPE=C
             FIELD    NAME=NAME   , START=10, BYTES=21, TYPE=C
             FIELD    NAME=COUNT    START=31, BYTES=2 , TYPE=H

         SEGMENT NAME=CREDIT  , PARENT=ACCOUNT ,
                 FORMAT=FI,
                 OCCURS=COUNT,
                 START =33,
                 BYTES =15

             FIELD    NAME=TYPE    , START=1,  BYTES=1 , TYPE=P, SCALE=0
             FIELD    NAME=AMOUNT ,  START=2,  BYTES=7,  TYPE=P, SCALE=2
             FIELD    NAME=LIMIT  ,  START=9,  BYTES=7,  TYPE=P, SCALE=2;
```

Figure 21. Using DataRefresher to Define the PR: CREATE DXTPSB

Notes:

1. The DXTPCB has two SEGMENT statements, which are followed by FIELD statements.
   - The first SEGMENT statement and its fields describe the containing IMS segment ACCOUNT in its DPROP format.
   - The second SEGMENT statement and its fields describe the internal segment type CREDIT in its DPROP format.
2. The Segment exit routine EKYESE2C is specified on the EXIT= keyword of the SEGMENT statement.

   The EXIT= keyword must be provided on the SEGMENT statement describing the containing segment, never on SEGMENT statements describing internal segments.

   The SEGMENT statement for the ACCOUNT segment also provides the following specifications:
   - BYTES=72 specifies the length of the segment in its IMS format.
   - XBYTES=77 specifies the length of the segment in its DPROP format.
3. The SEGMENT statement for the segment CREDIT describes the internal segment in its DPROP format.
   - FORMAT=FI specifies that the segment has a fixed length and is an internal segment.
   - OCCURS=COUNT specifies that the internal segment has a variable number of occurrences, and that the count field for the internal segment is the field called COUNT. The count field must be defined with a FIELD statement as a field of the containing segment (not as a field in the internal segment).
   - START=33 specifies that the internal segment starts at byte 33 of the containing segment (in its DPROP format).
   - BYTES=15 specifies that the internal segment has a length of 15 bytes.

4. The FIELD statement for the field TYPE describes the one-byte ID field built by the Segment exit routine in the DPROP format during IMS-to-DPROP mapping.
5. The FIELD statements for the fields AMOUNT and LIMIT describe the other fields of the internal segment or repeating group.

Using DataRefresher to Define the PR: CREATE DXTVIEW

Figure 22 shows a CREATE DXTVIEW definition for the Segment exit routine in Figure 17 on page 77.

```
CREATE
   DXTVIEW NAME     = VIEWCRED,
           DXTPSB   = KOEPSB2,
           DXTPCB   = PCB001,
           SEGMENT  = CREDIT,
           MINSEGM  = CREDIT,
           FIELDS   = *         ;
```

Figure 22. Using DataRefresher to Define the PR: CREATE DXTVIEW

Using DataRefresher To Define the PR

This section covers the DataRefresher UIM SUBMIT Command and EXTRACT Statement.

Figure 23 on page 92 contains two pairs of SUBMIT and EXTRACT statements to define the two PRs, PR1 and PR2.

PR1 propagates the fields ACT_NBR and NAME of the containing segment to the ACCOUNT table.

PR2 propagates the fields TYPE, AMOUNT, and LIMIT of the internal segment to the CREDIT table. PR2 propagates the key field ACT_NBR of the containing parent segment ACCOUNT to the CREDIT table.

```
SUBMIT   EXTID=PR1,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO, REPLACE',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,MAPDIR=TW,MAPCASE=1,
                         ACTION=REPL, ERROPT=BACKOUT,
                         PCBLABEL=HUPPCB'

EXTRACT INTO ACCOUNT (ACT_NBR   NOT NULL,
                      NAME      NOT NULL WITH DEFAULT)
        OPTIONS(FLDERR(SUBST(ZERO)))
        SELECT        ACT_NBR,
                      NAME
        FROM VIEWCRED ;




SUBMIT   EXTID=PR2,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO, RESUME(YES)',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,MAPDIR=TW,MAPCASE=3,
                         ACTION=REPL, ERROPT=BACKOUT,
                         PCBLABEL=HUPPCB'

EXTRACT INTO CREDIT (ACT_NBR   NOT NULL,
                     TYPE      NOT NULL,
                     AMOUNT    NOT NULL WITH DEFAULT,
                     LIMIT     NOT NULL WITH DEFAULT)
        OPTIONS(FLDERR(SUBST(ZERO)))
        SELECT       ACT_NBR,
                     TYPE   ,
                     AMOUNT ,
                     LIMIT
        FROM VIEWCRED ;
```

Figure 23. Using DataRefresher to Define the PR: DataRefresher UIM SUBMIT Command and EXTRACT Statement

Notes:

1. The MAPEXIT= keyword of the SUBMIT control statement specifies EKYMCE00. This causes DataRefresher UIM to call the DPROP-provided Map Capture Exit EKYMCE00 during processing of the SUBMIT or EXTRACT. This is needed to allow DPROP to create the PR.

2. MAPUPARM= is used to provide the DPROP propagation keywords.

   MAPCASE=1 defines PR1 as a mapping case 1 PR, and MAPCASE=3 defines PR2 as a mapping case 3 PR.

3. The EXTRACT statement describes to DataRefresher and DPROP which fields should be mapped to which columns.

   The COUNT field is not propagated by either PR1 or PR2.

Using DataRefresher For the Extract

This section covers INITDEM and USE DXTPSB Control Statements. Figure 24 shows INITDEM and USE DXTPSB control statements for the Segment exit routine in Figure 17 on page 77.

```
INITDEM  NAME=DEMPROD;
USE DXTPSB=KOEPSB2;
```

Figure 24. Using DataRefresher for the Extract: INITDEM and USE DXTPSB Control Statements

Defining the PR in the MVG Input Tables

Figure 25 on page 95 describes the DSNTEP2 SQL statements required to define the two PRs, PR1 and PR2, in the MVG input tables.

PR1 propagates to the ACCOUNT table the fields ACT_NBR and NAME of the containing segment.

PR2 propagates to the CREDIT table the fields TYPE, AMOUNT and LIMIT of the internal segment. PR2 propagates to the CREDIT table the key field ACT_NBR of the containing parent segment ACCOUNT.

The following rows are inserted into the MVG input tables to define PR1:

- One row is inserted into the DPRIPR table (the PR table).

  This row identifies the PRID by inserting an **E** into the PRTYPE column and a **1** into the MAPCASE column. The SQL statement specifies that the PR belongs to mapping case 1 of an extended-function PR.

- One row for the entity segment type ACCOUNT is inserted into the DPRISEG table (the SEG table).

  Because ACCOUNT is the root segment, no rows are inserted into DPRISEG for physical ancestors.

  The row describing ACCOUNT provides the following column values:

  – Value **E** in the ROLE column specifies that the segment is the entity segment of the PR.

  – Nonblank value EKYESE2C in the SEGEXIT column specifies that the segment must be processed by the Segment exit routine EKYESE2C.

  – Value **77** in the SEGEXITL column specifies the length of the segment in its DPROP format.

  – Value **F** in the SEGEXITF column specifies that the segment in its DPROP format has a fixed length.

- One row is inserted into the DPRITAB table (the TAB table).

  This row specifies that the target table is T096606.ACCOUNT.

- One row is inserted into the DPRIFLD table (the FLD table) for each propagated field.

  The DPRIFLD rows describe the fields as they appear in the DPROP format of the segment (as opposed to the segment in its IMS DB format).

The following rows are inserted into the MVG input tables to define PR2:

- One row is inserted into the DPRIPR table (the PR table).

  This row identifies the PRID by inserting an **E** into the PRTYPE column and a **3** into the MAPCASE column. The SQL statement specifies that the PR belongs to mapping case 3 of an extended-function PR.

- One row for the Containing segment Type ACCOUNT is inserted into the DPRISEG table (the SEG table).

  Because ACCOUNT is the root segment, no rows are inserted into DPRISEG for physical ancestors.

  The row describing ACCOUNT provides the following column values:

  – Value **C** in the ROLE column specifies that the segment is the containing segment of the mapping case 3 PR.

  – Nonblank value EKYESE2C in the SEGEXIT column specifies that the segment must be processed by the Segment exit routine EKYESE2C.

  – Value 77 in the SEGEXITL column specifies the length of the segment in its DPROP format.

  – Value **F** in the SEGEXITF column specifies that the segment in its DPROP format has a fixed length.

- One row for the internal segment type CREDIT is inserted into the DPRISEG table (the SEG table).

  The row describing CREDIT provides the following column values:

  – Value **E** in the ROLE column. This specifies that the segment is the Entity segment of the PR.

  – Blank value in the SEGEXIT column. This is because DPROP requires that the Segment exit routine be defined in the DPRISEG row of the containing segment (not in the DPRISEG row of the internal segment).

  – Value **FI** in the FORMAT column specifies that the segment has a fixed length and is an internal segment type.

  – Value COUNT in the OCCURS column specifies that the internal segment has a variable number of occurrences and that the number of occurrences is stored in the field COUNT.

  – Value **33** in the START column specifies that the first occurrence of the internal segment starts at location 33 within the containing segment (in its DPROP format).

  – Value **15** in the BYTES column specifies that the length of the internal segment type is 15 bytes.

- One row is inserted into the DPRITAB table (the TAB table).

  This row specifies that the target table is T096606.CREDIT.

- One row is inserted into the DPRIFLD table (the FLD table) for each propagated field. Another row is inserted into the DPRIFLD table for the COUNT field.

  The DPRIFLD rows describe the fields as they appear in the DPROP format of the segment (as opposed to the segment in its IMS DB format).

– The row describing the field ACT_NBR has the value ACCOUNT in the SEGNAME column. This specifies that the ACT_NBR field is located in the containing segment ACCOUNT (not in the internal segment).

– The row describing the field COUNT has a blank value in the COLNAME column, because the COUNT field is not propagated. The value ACCOUNT in the SEGNAME column specifies that the COUNT field is located in the containing segment ACCOUNT (not in the internal segment).

– The row describing the fields TYPE, AMOUNT, and LIMIT COUNT have the value CREDIT in the SEGNAME column. This specifies that these fields are located in the internal segment CREDIT.

```
DELETE FROM T096606.DPRIPR WHERE PRID = 'PR1'              ;

INSERT INTO T096606.DPRIPR
             ( PRID    , USERID  , PRTYPE, MAPCASE, MAPDIR,
               ERROPT  , ACTION)
      VALUES ('PR1'    ,'T096606','E'    ,'1'     ,'TW',
              'BACKOUT','REPL')                              ;

INSERT INTO T096606.DPRISEG
             ( PRID     , DBNAME   , SEGNAME ,   ROLE , PCBLABEL,
               SEGEXIT  , SEGEXITL, SEGEXITF )
      VALUES ('PR1'     ,'DB123'   ,'ACCOUNT',  'E'   ,'HUPPCB' ,
              'EKYESE2C', 77      ,'F')                        ;

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL ,  TABNAME )
      VALUES ('PR1',  'T096606',  'ACCOUNT')                   ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR1'    ,'DB123'  , 'ACCOUNT','ACT_NBR',
              'T096606','ACCOUNT',           'ACT_NBR',
              'C '     , 1       , 9)                        ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR1'    ,'DB123'  ,'ACCOUNT' ,'NAME'  ,
              'T096606','ACCOUNT',           'NAME'  ,
              'C '     , 10      ,  21)                      ;
```

Figure 25 (Part 1 of 2). Defining the PR in the MVG Input Tables

```
DELETE FROM T096606.DPRIPR WHERE PRID = 'PR2'               ;

INSERT INTO T096606.DPRIPR
             ( PRID    , USERID  , PRTYPE, MAPCASE, MAPDIR,
               ERROPT  , ACTION)
      VALUES ('PR2'    ,'T096606','E'    ,'3'     ,'TW'   ,
              'BACKOUT','REPL')                             ;

INSERT INTO T096606.DPRISEG
             ( PRID   ,  DBNAME  , SEGNAME ,   ROLE , PCBLABEL,
               SEGEXIT,  SEGEXITL, SEGEXITF )
      VALUES ('PR2'   , 'DB123'  ,'ACCOUNT',   'C'  , 'HUPPCB',
              'EKYESE2C',77      ,'F')                      ;


INSERT INTO T096606.DPRISEG
             ( PRID   ,  DBNAME  , SEGNAME  ,  ROLE , PCBLABEL,
               SEGEXIT,  SEGEXITL, SEGEXITF ,
               FORMAT ,  OCCURS  , START    ,  BYTES)
      VALUES ('PR2'   , 'DB123'  ,'CREDIT ' , 'E' , '       ',
              '      ',  0       ,' '       ,
              'FI'    , 'COUNT'  , '33'     ,  15)           ;

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL ,   TABNAME )
      VALUES ('PR2',  'T096606',  'CREDIT' )                 ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR2'    ,'DB123'  , 'ACCOUNT','ACT_NBR',
              'T096606','CREDIT ',           'ACT_NBR',
              'C '     , 1       , 9)                       ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR2'    ,'DB123'  , 'ACCOUNT','COUNT' ,
              'T096606','CREDIT ',           '    '  ,
              'H '     , 31      ,  2 )                     ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME  ,
               TABQUAL , TABNAME ,            COLNAME  ,
               DATATYPE, POSITION, BYTES    , SCALE)
      VALUES ('PR2'    ,'DB123'  ,'CREDIT'   ,'TYPE'   ,
              'T096606','CREDIT ',           'TYPE'    ,
              'P '     , 1       , 1        , 0    )     ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME  ,
               TABQUAL , TABNAME ,            COLNAME  ,
               DATATYPE, POSITION, BYTES    , SCALE)
      VALUES ('PR2'    ,'DB123'  , 'CREDIT ','AMOUNT ' ,
              'T096606','CREDIT ',           'AMOUNT'  ,
              'P '     , 2       ,  7       , 2    )      ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME  ,
               TABQUAL , TABNAME ,            COLNAME  ,
               DATATYPE, POSITION, BYTES    , SCALE)
      VALUES ('PR2'    ,'DB123'  , 'CREDIT ','LIMIT  ' ,
              'T096606','CREDIT ',           'LIMIT '  ,
              'P '     , 9       ,  7       , 2    )      ;
```

Figure 25 (Part 2 of 2). Defining the PR in the MVG Input Tables

Third Sample Segment Exit Routine

Figure 26 on page 98 contains an example of a Segment exit routine in PL/I. Its functions are the same as those for the exit routine in “Second Sample Segment Exit Routine” on page 75. For information about this routine, refer to “Second Sample Segment Exit Routine” on page 75.

The source code in Figure 26 on page 98 is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYESE2P. The definitions for this routine are the same as those for EKYESE2C, except that the exit name is different. Specifically, the **EXIT=EKYESE2C** in Figure 21 on page 90, and both occurrences of **EKYESE2C** in Figure 25 on page 95, are changed to **EKYESE2P**. The text that refers to EKYESE2C is also true for EKYESE2P. Refer to “Definitions for the Second Sample Segment Exit Routine” on page 88 for information about the definitions.

```
*PROCESS MAR(2,72,1);
 EKYESE2P: PROCEDURE /* Sample Segment Exit Routine */
                   (DAX_PARM_PTR,
                    IMSSEG_PARM_PTR,
                    DPRSEG_PARM_PTR,
                    USERAREA_PARM_PTR)
          OPTIONS   (FETCHABLE REENTRANT);


 /*********************************************************************
  *                                                                   *
  *       Licensed Materials - Property of IBM                        *
  *                                                                   *
  *       5685-124 (C) Copyright IBM Corp. 1989, 1992.                *
  *                                                                   *
  *       See Copyright Instructions                                  *
  *                                                                   *
  *********************************************************************/
1/*********************************************************************
  * Module name: EKYESE2P                                             *
  *                                                                   *
  * Descriptive name: Sample PL/I Segment Exit Routine                *
  *                                                                   *
  * Function: The intent of this program is to provide a sample of    *
  *           a segment exit routine.  This example is used for the   *
  *           transformation of a segment layout between its:         *
  *             - IMS format                                          *
  *             - DPROP format.                                       *
  *                                                                   *
  * EKYESE2P illustrates the usage of a DPROP segment exit to support *
  * the propagation of an IMS segment containing an internal          *
  * segment / repeating group of fields.                              *
  *                                                                   *
  *                                                                   *
  * This sample segment exit routine supports TYPE=E PR's and is      *
  * therefore called both for:                                        *
  *                                                                   *
  *    - IMS-to-DPROP mapping                                         *
  *      (e.g. during IMS-to-DB2 propagation, also during             *
  *       DXT-extracts, CCU and DLU processing).                      *
  *                                                                   *
  *    - DPROP-to-IMS mapping                                         *
  *      (e.g. during DB2-to-IMS propagation, also during CCU and     *
  *       DLU processing).                                            *
  *                                                                   *
  *                                                                   *
  * In this example the propagated IMS segment is a bank account      *
  * segment.  The IMS segment consists of the following fields:       *
  *                                                                   *
  *    - The account number (this is the key of the segment).         *
  *    - The customer name.                                           *
  *    - A repeating group of fields with three occurrences.          *
  *      Each occurrence of the repeating group contains information  *
  *      about one type of credit that the bank is granting.          *
  *      This data is:                                                *
  *         - the current amount of credit granted to the             *
  *           customer/account.                                       *
```

Figure 26 (Part 1 of 12). Third Sample Segment Exit Routine (PL/I)

```
 *        - the credit limit for the customer/account.              *
 *                                                                   *
 *********************************************************************
1*********************************************************************
 *                                                                   *
 * The database administrator wants to have a normalized DB2 table   *
 * design and therefore wants to:                                    *
 *                                                                   *
 *    1) Propagate the account number and customer name to/from the  *
 *       table called "ACCOUNT".  This is done with a mapping-case-1 *
 *       propagation request.                                        *
 *                                                                   *
 *    2) Propagate the information akin to the different types of    *
 *       credits (together with the account number) to/from another  *
 *       table called "CREDIT".
 *                                                                   *
 *       This is done with a mapping-case-3 propagation request.     *
 *                                                                   *
 *       Each occurrence of the credit information (there are three  *
 *       of them) is considered to be an occurrence of an internal   *
 *       and is propagated to/from one row of the table "CREDIT".    *
 *                                                                   *
 *       To distinguish between the three types of credit            *
 *       information within the CREDIT table (and in order to have   *
 *       a DB2 primary key), the CREDIT table does not only contain  *
 *       an account number column and the current credit amount and  *
 *       limit.  The CREDIT table also contains a "TYPE" column      *
 *       which identifies the type of credit.                        *
 *                                                                   *
 * The sample segment exit routine EKYESE2P provides logic to        *
 * support the propagation of the IMS segment to/from the two tables *
 * "ACCOUNT" and "CREDIT".                                           *
 *                                                                   *
 *********************************************************************
1*********************************************************************
 *                                                                   *
 *                                                                   *
 * 1) For IMS-to-DPROP mapping the sample exit provides the          *
 *    following functions, when building the DPROP format of the     *
 *    segment:                                                       *
 *                                                                   *
 *    - The exit routine creates in the DPROP-format an "ID" field   *
 *      for each occurrence of the internal segment.  This is the    *
 *      field called "TYPE".                                         *
 *                                                                   *
 *      This addresses the DPROP requirement that internal segments  *
 *      have an "ID" field uniquely identifying the occurrences of   *
 *      the internal segments within the containing segment.         *
 *                                                                   *
 *      In the DPROP format, each occurrence of the internal segment *
 *      will consist of the following fields:                        *
 *         - The field "TYPE" (this is the "ID" field created by the *
 *           exit).                                                  *
 *         - The field "AMOUNT" (copied from the IMS format of the   *
 *           segment).                                               *
 *         - The field "LIMIT" (copied from the IMS format of the    *
 *           segment).                                               *
 *                                                                   *
 *    - The exit routine creates in the DPROP-format a count field   *
 *      Its value is the number of occurrences of the internal       *
 *      segment-type within the containing segment.                  *
```

Figure 26 (Part 2 of 12). Third Sample Segment Exit Routine (PL/I)

```
 *      Note that a count field is required by DPROP for the         *
 *      propagation of internal segments with TYPE=E PR's.           *
 *                                                                   *
 *********************************************************************
1*********************************************************************
 *                                                                   *
 * 2) For DPROP-to-IMS mapping the sample exit differentiates        *
 *    between the two following cases:                               *
 *                                                                   *
 *    a) It is called during a REPLACE, DELETE, or INSERT of a row   *
 *       of the "CREDIT" table.                                      *
 *                                                                   *
 *       Here, the exit routine gets the ensuing two inputs from     *
 *       DPROP:                                                      *
 *                                                                   *
 *          - The changed occurrence of the internal segment in its  *
 *            DPROP format.  This input has been built by DPROP by   *
 *            mapping the changed CREDIT row to the DPROP format of  *
 *            the internal segment.                                  *
 *                                                                   *
 *          - The existing "before-change" IMS segment in its IMS    *
 *            format.                                                *
 *                                                                   *
 *       By combining information from these two inputs the segment  *
 *       exit routine is responsible for building the new            *
 *       "after-change" IMS segment in its IMS format.               *
 *                                                                   *
 *    b) It is called during a REPLACE, DELETE, or INSERT of a row   *
 *       of the "ACCOUNT" table.                                     *
 *                                                                   *
 *       Here, this exit routine gets following two inputs from      *
 *       DPROP:                                                      *
 *                                                                   *
 *          - The changed occurrence of the containing segment in    *
 *            its DPROP format.  This input has been built by DPROP  *
 *            by mapping the changed ACCOUNT row to the DPROP format *
 *            of the containing segment.                             *
 *                                                                   *
 *          - The existing "before-change" IMS segment in its IMS    *
 *            format (only for REPLACES and DELETES of rows of the   *
 *            ACCOUNT table).                                        *
 *                                                                   *
 *       By combining information from these two inputs the segment  *
 *       exit routine is responsible for building the new            *
 *       "after-change" IMS segment in its IMS format.               *
 *                                                                   *
 *********************************************************************
1*********************************************************************
 *                                                                   *
 * The figure below describes on the left-hand side the segment in   *
 * its IMS format and on the right-hand side the segment in its      *
 * DPROP format.                                                     *
 *                                                                   *
 *                                                                   *
 *      *-------------------------*   *-------------------------*   *
 *      |    IMS segment in its   |   |    IMS segment in its   |   *
 *      |    IMS format           |   |    DPROP format         |   *
 *      *-------------------------*   *-------------------------*   *
 *                                                                   *
```

Figure 26 (Part 3 of 12). Third Sample Segment Exit Routine (PL/I)

```
 *      *----------*--------*-----*    *----------*--------*-----*   *
 *      | field    | field  |field|    | field    | field  |field|   *
 *      | name     | format |start|    | name     | format |start|   *
 *      *----------*--------*-----*    *----------*--------*-----*   *
 *      | ACNT_NBR |   Z    |  1  |<-->| ACNT_NBR |   Z    |  1  |   *
 *      | NAME     |   C    |  10 |<-->| NAME     |   C    |  10 |   *
 *      |          |        |     |    | COUNT    |   H    |  31 |   *
 *      |          |        |     |    | TYPE_1   |   P    |  33 |   *
 *      | AMOUNT_A |   P    |  31 |    | AMOUNT_1 |   P    |  34 |   *
 *      | LIMIT_A  |   P    |  38 |    | LIMIT_1  |   P    |  41 |   *
 *      |          |        |     |    | TYPE_2   |   P    |  48 |   *
 *      | AMOUNT_B |   P    |  45 |    | AMOUNT_2 |   P    |  49 |   *
 *      | LIMIT_B  |   P    |  52 |    | LIMIT_2  |   P    |  56 |   *
 *      |          |        |     |    | TYPE_3   |   P    |  63 |   *
 *      | AMOUNT_C |   P    |  59 |    | AMOUNT_3 |   P    |  64 |   *
 *      | LIMIT_C  |   P    |  66 |    | LIMIT_3  |   P    |  71 |   *
 *      *----------*--------*-----*    *----------*--------*-----*   *
 *                                                                   *
 * Both the IMS format and the DPROP format of the IMS segment are   *
 * defined as fixed-length.                                          *
 *                                                                   *
 * The internal segment is defined to DPROP as follows:              *
 *                                                                   *
 *    - It has a variable number of occurrences (the number of       *
 *      occurrences is in the count field "COUNT" of the containing  *
 *      segment).                                                    *
 *    - The first occurrence starts at a fixed location within the   *
 *      containing segment (start position = 33).                    *
 *    - It has a fixed length (15 bytes).                            *
 *    - It consists of the following fields:                         *
 *       TYPE   (1 byte).                                            *
 *       AMOUNT (7 bytes).                                           *
 *       LIMIT  (7 bytes).                                           *
 *                                                                   *
 * Please refer to the DSECTS found later in this module to find     *
 * all the details about the "IMS format" and the "DPROP format"     *
 * of the segments.                                                  *
 *                                                                   *
 *********************************************************************
1*********************************************************************
 *                                                                   *
 * The following conventions are used to describe credit information *
 * if it does not exist:                                             *
 *    - In the IMS format, a non-existing credit-info has a zero     *
 *      value in the field "LIMIT".                                  *
 *                                                                   *
 *    - In the DPROP-format, the count reflects the number of        *
 *      existing CREDIT internal segments.  Existing CREDIT internal *
 *      segments follow each other in the DPROP format of the IMS    *
 *      segment (non-existing internal segments are eliminated).     *
 *      This must be so in order to conform to the way that internal *
 *      segments are defined to DPROP and DXT.                       *
 *                                                                   *
 *                                                                   *
 * Input:   1st parameter: Address of DAX (DAX is the exit interface *
 *                         control block).                           *
 *          2nd parameter: Address of segment in IMS format.         *
 *          3rd parameter: Address of segment in DPROP format.       *
 *          4th parameter: Address of anchor area preserved          *
 *                         across calls to this exit.                *
 *                                                                   *
```

Figure 26 (Part 4 of 12). Third Sample Segment Exit Routine (PL/I)

```
 * Output:  the segment format transformation has been completed.    *
 *                                                                   *
 *                                                                   *
 * Exit-error:                                                       *
 *                                                                   *
 *    Return code  = 12 - mapping problem / invalid data             *
 *                 = 16 - should-not-occur errors (invalid call      *
 *                        function, parameter area too small,        *
 *                        invalid segment name).                     *
 *                                                                   *
 *                                                                   *
 *    Error messages issued by EKYESE2P:                             *
 *                                                                   *
 *       EKYESE2P-1E: Call function not supported.                   *
 *       EKYESE2P-2E: Unsupported DBD or segment name.               *
 *       EKYESE2P-3E: Unexpected length of IMS segment.              *
 *       EKYESE2P-4E: DPROP segment is too short.                    *
 *       EKYESE2P-5E: IMS segment is too short.                      *
 *       EKYESE2P-6E: Unexpected value in TYPE column of CREDIT      *
 *                    table.                                         *
 *                                                                   *
 * Change activity= none                                             *
 *                                                                   *
 *                                                                   *
 ****************** End of Specifications ****************************
1 ****************** Logic of EKYESE2P ********************************
 *                                                                   *
 *                                                                   *
 * Main logic:                                                       *
 *                                                                   *
 * 1) Module entry logic:                                            *
 *                                                                   *
 *    - Set "module entered" and "module in control" flags into DAX. *
 *                                                                   *
 *    - Validate that the exit is invoked to propagate the correct   *
 *      database and segment.                                        *
 *                                                                   *
 *    - Process according to call-function either for:               *
 *       - The processing of IMS-to-DPROP, or                        *
 *       - The processing of DPROP-to-IMS                            *
 *                                                                   *
 *                                                                   *
 * 2) IMS-to-DPROP formatting:                                       *
 *                                                                   *
 *    - Check length of segment in its IMS format and check that the *
 *      size of the DPROP segment buffer is sufficient to contain    *
 *      the segment in its DPROP format.                             *
 *                                                                   *
 *    - Assign the account number and the customer name to the DPROP *
 *      format.                                                      *
 *                                                                   *
 *    - Initialize the computation of internal segment occurrences   *
 *      to zero.                                                     *
 *                                                                   *
 *    - For each non-zero limit in the IMS format:                   *
 *       - Increase the occurrence counters by 1.                    *
 *       - Create the ID of the internal segment in the DPROP buffer.*
 *       - Move the data of the internal segment to the DPROP buffer.*
 *                                                                   *
```

Figure 26 (Part 5 of 12). Third Sample Segment Exit Routine (PL/I)

```
 *    Note: A limit with a zero value in the IMS format is deemed    *
 *          as identifying "non-existing" credit information.        *
 *                                                                   *
 *          In the DPROP format there will be no occurrence of       *
 *          internal segments for these  non-existing credits.       *
 *          As required by DPROP, the occurrences for the            *
 *          existing internal segments will follow each other.       *
 *                                                                   *
 *********************************************************************
1*********************************************************************
 *                                                                   *
 * 3) DPROP-to-IMS formatting                                        *
 *                                                                   *
 *    - Check that the IMS segment buffer is large enough.           *
 *                                                                   *
 *    - Initialize the IMS segment buffer as follows:                *
 *       - If before-change image is provided by the caller,         *
 *         copy the before-change image to the IMS buffer,           *
 *         otherwise initialize the IMS buffer with the appropriate  *
 *         initial values (zeroes and blanks).                       *
 *                                                                   *
 *    - If processing a change to the target of a containing         *
 *      segment:                                                     *
 *       - Copy information of the changed containing segment from   *
 *         the DPROP buffer to the IMS buffer.                       *
 *                                                                   *
 *    - If processing a change to the target of an internal          *
 *      segment:                                                     *
 *       - If processing a DELETE, set the appropriate CREDIT        *
 *         information to zero in the IMS buffer.                    *
 *       - If processing a REPLACE or INSERT, copy the information   *
 *         of the changed internal segment from the DPROP buffer to  *
 *         the IMS buffer.                                           *
 *                                                                   *
 * Error Logic:                                                      *
 *                                                                   *
 *    - Format an error message in the "DAX".                        *
 *                                                                   *
 *    - Set a return code in the "DAX".                              *
 *                                                                   *
 *    - Return to the caller.                                        *
 *                                                                   *
 *****************  End of logic summary ***************************/
 %INCLUDE EKYRCDXP; /* DAX control block structure */
1/********************************************************************
 * Description of IMS segment in its IMS format                      *
 ********************************************************************/

 DECLARE IMSSEGBUFF CHAR(72) BASED(IMSSEG_POINTER);

 DECLARE 1 IMSSEG BASED(IMSSEG_POINTER),
    2 IMSACNBR        PIC'(8)9T',    /* Account number (KEY)          */
    2 IMSNAME         CHAR(21),      /* Name of customer              */
    2 IMSINSEG(3),                   /* 3 occurrences of internal seg */
      3 IMSAMOUNT     FIXED DEC(13), /* Current amount type-a credit  */
      3 IMSLIMIT      FIXED DEC(13);
```

Figure 26 (Part 6 of 12). Third Sample Segment Exit Routine (PL/I)

```
 /*********************************************************************
  * Description of IMS segment in its DPROP format                    *
  *********************************************************************/

 DECLARE DPRSEGBUFF CHAR(77);

 DECLARE 1 DPRSEG BASED(DPRSEG_POINTER),

    2 DPRACNBR        PIC '(8)9T',   /* Account number (key)          */
    2 DPRNAME         CHAR(21),      /* Name of customer              */
    2 DPRCOUNT        FIXED BIN(15), /* Internal segment occurrences
                                        count                         */
    2 DPRINSEG(3),                   /* three occurrences of
                                        internal segment              */
      3 DPRTYPE       FIXED DEC(1),  /* ID                            */
      3 DPRAMOUNT     FIXED DEC(13), /* Current amount                */
      3 DPRLIMIT      FIXED DEC(13); /* Limit amount                  */
1/*********************************************************************
  * Description of Internal segment in DPROP format                   *
  *********************************************************************/

 DECLARE DPRI_SEG_POINTER POINTER;

 DECLARE 1 DPRI_SEG(3) BASED (DPRI_SEG_POINTER),
      2 DPRI_TYPE     FIXED DEC(1),  /* ID                            */
      2 DPRI_AMOUNT   FIXED DEC(13), /* Current amount                */
      2 DPRI_LIMIT    FIXED DEC(13); /* Limit amount                  */



 /*********************************************************************
  * Description of Containing segment in DPROP format                 *
  *********************************************************************/

 DECLARE DPRC_SEG_POINTER POINTER;

 DECLARE 1 DPRC_SEG BASED (DPRC_SEG_POINTER),
    2 DPRC_ACNBR      PIC'(8)9T',    /* Account number (KEY)          */
    2 DPRC_NAME       CHAR(21),      /* Name of customer              */
    2 DPRC_COUNT      FIXED BIN(15); /* Internal segment occurrences
                                        count                         */



 /*********************************************************************
  * Area for the before-change IMS image                              *
  *********************************************************************/

 DECLARE IMSBEFIM_POINTER   POINTER;
 DECLARE IMSBEFIM   CHAR(72) BASED(IMSBEFIM_POINTER);
1/*********************************************************************
  * Built-in functions and global variable declarations               *
  *********************************************************************/

 DECLARE DAX_PARM_PTR       POINTER;
 DECLARE IMSSEG_PARM_PTR    POINTER;
 DECLARE DPRSEG_PARM_PTR    POINTER;
 DECLARE USERAREA_PARM_PTR  POINTER;
 DECLARE DAX_POINTER        POINTER;
 DECLARE IMSSEG_POINTER     POINTER;
 DECLARE DPRSEG_POINTER     POINTER;
```

Figure 26 (Part 7 of 12). Third Sample Segment Exit Routine (PL/I)

```
 DECLARE I                  FIXED BIN(31);
 DECLARE J                  FIXED BIN(31);

 DECLARE ADDR               BUILTIN;
 DECLARE NULL               BUILTIN;
1/*********************************************************************
  * Declarations for initialized variables                            *
  *********************************************************************/

 DECLARE EKYESE1E            FIXED BIN(31) INIT(1);
 DECLARE EKYESE2E            FIXED BIN(31) INIT(2);
 DECLARE EKYESE3E            FIXED BIN(31) INIT(3);
 DECLARE EKYESE4E            FIXED BIN(31) INIT(4);
 DECLARE EKYESE5E            FIXED BIN(31) INIT(5);
 DECLARE EKYESE6E            FIXED BIN(31) INIT(6);

 DECLARE RETURN_CODE_12      FIXED BIN(31) INIT(12);
 DECLARE RETURN_CODE_16      FIXED BIN(31) INIT(16);

 DECLARE IMSSEGL             FIXED BIN(31) INIT(72);
 DECLARE DPRSEGL             FIXED BIN(31) INIT(77);

 DECLARE X                   CHAR(1)       INIT('X');
 DECLARE NO                  CHAR(2)       INIT('NO');
 DECLARE RV                  CHAR(2)       INIT('RV');
 DECLARE DB123               CHAR(8)       INIT('DB123   ');
 DECLARE ACCOUNT             CHAR(8)       INIT('ACCOUNT ');
1/*********************************************************************
  * Format error messages and place into DAX control block            *
  *********************************************************************/

 WRITE_ERROR_MESSAGE: PROCEDURE(MESSAGE_NUM,RETURN_CODE);

    DECLARE MESSAGE_NUM    FIXED BIN(31); /* Message identifier input */
    DECLARE RETURN_CODE    FIXED BIN(31); /* Return code input       */

    /******************************************************************
     * Buffer and structure for assignment of DAX message             *
     ******************************************************************/
    DECLARE MSGBUFF CHAR(64);
    DECLARE MSG_PTR POINTER;
    DECLARE 1 MSGLINE BASED(MSG_PTR),
       2 MSGID                 CHAR(11),
       2 MSGBL1                CHAR(1),
       2 MSGTXT                CHAR(52);

    /******************************************************************
     * Error and informative message declarations                     *
     ******************************************************************/
    DECLARE MESSAGE_ID(6)    CHAR(11) INIT
       ('EKYESE2P-1E','EKYESE2P-2E','EKYESE2P-3E',
        'EKYESE2P-4E','EKYESE2P-5E','EKYESE2P-6E');

    DECLARE MESSAGE_TEXT(6)  CHAR(52) INIT
         ('Call function not supported.                        ',
          'Unsupported DBD or segname.                         ',
          'Unexpected length of IMS segment.                   ',
          'DPROP segment buffer is too short.                  ',
          'IMS segment buffer is too short.                    ',
          'Unexpected value in type column of credit table.    ');
```

Figure 26 (Part 8 of 12). Third Sample Segment Exit Routine (PL/I)

```
   /*****************************************************************
    * Format message and assign return code to DAX                  *
    *****************************************************************/

   MSGBUFF  = ' ';                      /* Blank out message buffer */

   MSG_PTR  = ADDR(MSGBUFF);
   MSGID    = MESSAGE_ID(MESSAGE_NUM);
   MSGTXT   = MESSAGE_TEXT(MESSAGE_NUM);

   DAXRETC  = RETURN_CODE;              /* Return code into DAX     */
   DAXSMESG = MSGBUFF;                  /* Formatted message to DAX */

 END WRITE_ERROR_MESSAGE;
1/*******************************************************************
  * Normal call to transform the segment from its IMS format into     *
  * its DPROP format                                                  *
  *******************************************************************/

 IMS_TO_DPROP: PROCEDURE;

   /*****************************************************************
    * Establish addressability to DPROP format segment data area     *
    *****************************************************************/
   DPRSEG_POINTER = ADDR(DPRSEG_PARM_PTR);

   /*****************************************************************
    * Check the length of segment in its IMS format and check that   *
    * the DPROP buffer is large enough to contain the segment in its *
    * DPROP format.                                                  *
    *****************************************************************/
   IF DAXDLEN ¬= IMSSEGL THEN   /* Unexpected length of IMS segment  */
   DO;
      CALL WRITE_ERROR_MESSAGE(EKYESE3E,RETURN_CODE_16);
      GOTO FIN; /* Terminate */
   END;
   IF DAXFLEN < DPRSEGL THEN    /* DPROP segment buffer is too short */
   DO;
      CALL WRITE_ERROR_MESSAGE(EKYESE4E,RETURN_CODE_16);
      GOTO FIN; /* Terminate */
   END;

   /*****************************************************************
    * Assign the account number and customer name to DPROP format    *
    *****************************************************************/
   DPRACNBR = IMSACNBR;
   DPRNAME  = IMSNAME;

   /*****************************************************************
    * Initialize counter field to zero and                           *
    * initialize processing for the three credits                    *
    *****************************************************************/
   DPRCOUNT = 0;
   DPRINSEG = 0;
   J = 0;

   DO I = 1 TO 3;                     /* Move credits              */
      IF IMSLIMIT(I) ¬= 0             /* i.e. non zero limit       */
      THEN DO;
         J = J + 1;
         DPRCOUNT     = J;            /* Increase occurence counter */
```

Figure 26 (Part 9 of 12). Third Sample Segment Exit Routine (PL/I)

```
        DPRTYPE(J)   = I;            /* Assign ID of internal seg  */
        DPRAMOUNT(J) = IMSAMOUNT(I); /* Assign current amount      */
        DPRLIMIT(J)  = IMSLIMIT(I);  /* Assign limit amount        */
     END; /* IMSLIMIT(I) ¬= 0 */
  END; /* I */

 END IMS_TONDPROP;
1/*********************************************************************
  * Reverse call to transform the segment from its DPROP format into  *
  * its IMS format                                                    *
  *********************************************************************/

 DPROP_TO_IMS: PROCEDURE;
  DECLARE INTERNAL           CHAR(1)      INIT('I');
  DECLARE DLET               CHAR(4)      INIT('DLET');
  DECLARE NINE_ZEROS         PIC'(8)9T'   INIT(0);
  DECLARE BLANKS_21          CHAR(21)     INIT(' ');

  /*****************************************************************
   * Check that the IMS buffer is large enough to contain the      *
   * segment in its IMS format.                                    *
   *****************************************************************/
  IF DAXDLEN < IMSSEGL THEN  /* IMS segment buffer is too short    */
  DO;
     CALL WRITE_ERROR_MESSAGE(EKYESE5E,RETURN_CODE_16);
     GOTO FIN; /* Terminate */
  END;

  /*****************************************************************
   * Initialize the after change IMS format as follows:            *
   *                                                               *
   * If before change image of IMS segment has not been provided,  *
   *    initialize the after change image with the appropriate     *
   *    initial values                                             *
   * else                                                          *
   *    initialize the after change image with before change image *
   *****************************************************************/

  IF DAXIDDSB = NULL       /* Before-change image IS NOT provided  */
  THEN DO;
     IMSACNBR = NINE_ZEROS; /* Initialize account number to zeros   */
     IMSNAME  = BLANKS_21;  /* Initialize customer name to blanks   */
     IMSINSEG = 0;          /* Initialize amounts/limits to zero    */
  END; /* DAXIDDSB is NULL */

  ELSE DO;                 /* Before-change image IS provided      */
     IMSBEFIM_POINTER = DAXIDDSB; /* Address of before-change image */
     IMSSEGBUFF = IMSBEFIM; /* Assign before-change to after-change */
  END;
1   IF DAXSEGT = INTERNAL  /* value is "I" (i.e. internal segment)   */
  THEN
  /*****************************************************************
   * Exit routine is called for DPROP-to-IMS mapping because the   *
   * target of the internal segment has changed.                   *
   *                                                               *
   * When processing a DELETE -                                    *
   *    the exit routine will zero the appropriate amount and limit.*
   * When processing an INSERT or REPLACE -                        *
   *    the exit routine will copy the amount and limit from the   *
   *    changed internal segment to the IMS format of the segment. *
   *****************************************************************/
```

Figure 26 (Part 10 of 12). Third Sample Segment Exit Routine (PL/I)

```
    DO;
       DPRSEG_POINTER = ADDR(DPRSEGBUFF);
       /***************************************************************
        * Establish addressability to DPROP format segment data area  *
        ***************************************************************/
       DPRI_SEG_POINTER  = ADDR(DPRSEG_PARM_PTR);
       DPRINSEG = DPRI_SEG;
       SELECT(DPRTYPE(1));
          WHEN(1,2,3)      /* Determine which internal segment       */
          DO;              /* occurrence has changed                 */
             J = DPRTYPE(1);
             SELECT(DAXDPRCT);
                WHEN(DLET) /* Processing a DELETE                    */
                DO;
                   IMSAMOUNT(J) = 0; /* Zero appropriate amount      */
                   IMSLIMIT(J)  = 0; /* Zero appropriate limit       */
                END;
                OTHERWISE  /* Assume processing an INSERT or REPLACE  */
                DO;
                   IMSAMOUNT(J)=DPRAMOUNT(1); /* Copy internal amount */
                   IMSLIMIT(J) =DPRLIMIT(1);  /* Copy internal limit  */
                END;
             END; /* Select DAXDPRCT */
          END; /* When DPRTYPE(1) = 1, 2 or 3 */
          OTHERWISE /* Unexpected value in type column of credit tab  */
          DO;
             CALL WRITE_ERROR_MESSAGE(EKYESE6E,RETURN_CODE_12);
             GOTO FIN; /* Terminate */
          END;
       END; /* Select DPRTYPE */
    END;
1   ELSE /* DAXSEGT not = "I" (i.e. not an internal segment)         */
    /******************************************************************
     * Exit routine is called for DPROP-to-IMS mapping  because      *
     * the target of the containing segment has changed.             *
     * We will just move information from the containing segment     *
     * in its DPROP format to segment in its IMS format.             *
     ******************************************************************/
    DO;
       /***************************************************************
        * Establish addressability to DPROP format segment data area  *
        ***************************************************************/
       DPRC_SEG_POINTER = ADDR(DPRSEG_PARM_PTR);
       IMSACNBR = DPRC_ACNBR;
       IMSNAME  = DPRC_NAME;
    END;

 END DPROP_TO_IMS;
1/*********************************************************************
  *                       Main Routine                               *
  *********************************************************************/

 /*********************************************************************
  * Establish addressability to DAX control block and IMS segment data*
  *********************************************************************/
 DAX_POINTER    = ADDR(DAX_PARM_PTR);
 IMSSEG_POINTER = ADDR(IMSSEG_PARM_PTR);

 /*********************************************************************
  * Set the "exit entered" and "exit in control" flags.              *
  *********************************************************************/
```

Figure 26 (Part 11 of 12). Third Sample Segment Exit Routine (PL/I)

```
DAXENTRD = X;
DAXINCTL = X;

/*********************************************************************
 * Verify that the exit is called to format the expected IMS        *
 * database and segment type                                         *
 *********************************************************************/
IF DAXDBNM ¬= DB123 THEN              /* Unsupported DBD             */
DO;
   CALL WRITE_ERROR_MESSAGE(EKYESE2E,RETURN_CODE_16);
   GOTO FIN; /* Terminate */
END;

IF DAXSEGM ¬= ACCOUNT THEN            /* Unsupported segname         */
DO;
   CALL WRITE_ERROR_MESSAGE(EKYESE2E,RETURN_CODE_16);
   GOTO FIN; /* Terminate */
END;

/*********************************************************************
 * Process depending on call-function                                *
 *********************************************************************/
SELECT (DAXCALL);

   WHEN (NO) CALL IMS_TO_DPROP;       /* Normal call (IMS to DPROP)  */

   WHEN (RV) CALL DPROP_TO_IMS;       /* Reverse call (DPROP to IMS) */

   OTHERWISE                          /* Unsupported call function   */
   DO;
      CALL WRITE_ERROR_MESSAGE(EKYESE1E,RETURN_CODE_16);
      GOTO FIN; /* Terminate */
   END;

END; /* Select DAXCALL */

FIN: /* End of processing */

END EKYESE2P;
```

Figure 26 (Part 12 of 12). Third Sample Segment Exit Routine (PL/I)

Chapter 3. Field Exit Routines

The RUP and HUP call the Segment and Field exit routines as part of DPROP's generalized mapping logic processing. These exit routines are optional and can be used to reformat or change data during propagation. The generalized mapping logic of the RUP or HUP can take care of most situations, but if your data is stored in an unusual way, or in some form that the RUP or HUP cannot handle, consider writing a Field exit routine to proceed.

A Field exit routine is generally used to convert an individual IMS data field between a *user format* that DPROP does not support, and a *DPROP-supported format* that you have defined in your PR. This is further referred to as:

- ***User-to-DPROP mapping*** when your exit routine is called to convert the field from its user format to the DPROP format. Calls to an exit routine for user-to-DPROP mapping are generated by the RUP as part of IMS-to-DB2 propagation.
- ***DPROP-to-user mapping*** when your exit routine is called to convert the field from its DPROP format to its user format. Calls to exit routine for DPROP-to-user mapping are generated by the HUP as part of DB2-to-IMS propagation.

Typical uses of Field exit routines include:

- Converting IMS fields that have special formats that DPROP does not directly support.
- Performing data conversions that are not supported by the DPROP data conversion routines.
- The following support for DATE and TIME formats in IMS fields:
  - Installation-specific (LOCAL) DATE and TIME formats
  - During RH-propagation, support of DATE and TIME formats other than those identified during DPROP installation
- Converting between some values in an IMS field and a DB2 NULL value.
- Cleaning up or reorganizing IMS data stored in an unusual way.
- If performing DB2-to-IMS propagation to convert the value of a numeric DB2 column into a packed or zoned IMS field, having a sign-code other than the “preferred” sign codes `X'.C'` and `X'.D'`. (The period (.) represents the numeric digit that precedes the sign code.)

Field exit routines have many of the same characteristics as Segment exit routines.

Field exit routines used with TYPE=L or TYPE=F PRs are only called to perform HR-propagation and therefore support only user-to-DPROP mapping.

Field exit routines used with TYPE=E PRs support both user-to-DPROP mapping and DPROP-to-user mapping, even if the TYPE=E PR specifies MAPDIR=HR. This is because your Field exit routine can be called by the CCU and DLU. The conversion performed during DPROP-to-user mapping must be the reverse of the conversion performed during user-to-DPROP mapping.

1. During IMS-to-DB2 mapping, the RUP calls your Field exit routines immediately after your Segment exit routine, if you are using one. If you are not using a Segment exit routine, as soon as the RUP receives the changed data segment, it calls your Field exit routine.

   Your Field exit routine must convert the field from its user format to the DPROP-supported format that you specified during the PR definition. The RUP calls a Field exit routine for each field that requires one according to your field definitions.

   After calling your optional Segment exit routine and your optional Field exit routine, the RUP converts the field formats that you specified in your PR definition to the format of the DB2 columns.

2. During DB2-to-IMS mapping, the HUP converts the format of the DB2 columns into the field format that you specified in your PR definitions. Then the HUP calls your Field exit routine before your Segment exit routine, if you are using one.

   Your Field exit routine must convert the field from the DPROP-supported format specified during PR definition to its user format. The HUP calls a Field exit routine for each field that, according to your PR definitions, can be processed by one.

   Finally, after performing its own data conversion, calling your optional Field exit routines, and calling your optional Segment exit routines, the HUP updates the IMS database segment.

Like the Segment exit routines, your Field exit routines can be written in Assembler, or in COBOL, PL/I, or C. DPROP support for exit routines written in high-level languages requires LE/370 Version 1 Release 2.

For synchronous propagation, the RUP and HUP call your exits in both IMS batch and online dependent regions accessing DB2. For LOG-ASYNC propagation, the RUP calls your exit routines in an MVS batch environment. During user asynchronous propagation, depending on your implementation, the RUP calls your exit routines in IMS batch and dependent regions accessing DB2, or in a non-IMS DB2 TSO or CAF environment. DPROP also calls your exits during execution of the CCU and DLU.

DataRefresher calls Field exits **User Data Type exits**. If you are using DataRefresher to extract IMS data, your exit routines are called directly by DataRefresher during data extraction so that the mapping performed during extraction and data propagation are the same. DataRefresher also generates a definition call to your exit routine when you define the field on the CREATE DXTPSB statement.

How To Write A Field Exit Routine

This section describes some guidelines and requirements to follow when writing your Field exit routine. If your exit routine is used by DataRefresher during data extraction, it must follow these rules. See the appropriate DataRefresher or DXT documentation for information about DataRefresher requirements.

As with Segment exits, when the RUP or HUP calls your Field exit routine, the following four parameters are passed to your exit:

- An interface control block
- A user format buffer for the field in its user format
- A DPROP format buffer for the field in its DPROP format
- A 64-byte anchor area

If your exit routine is written in Assembler, register 1 contains the address of a list. This list is four fullwords long and contains the addresses of the parameters in the order listed above.

Interface Control Block

Figure 28 on page 115 shows the structure of the interface control block, which is called EKYRCUDT and is passed to your Field exit routine. There is one interface control block per exit routine, lasting the duration of the exit in virtual storage. The following table lists:

- The fields most useful to your exit routine
- What the fields are used for
- Their displacement in the control block DSECT

Figure 27 (Page 1 of 2). Interface Control Block Parameters for Field Exits

Field	Used For	Displacement
UDTCALL	Call function, describing whether your exit routine is called either to perform user-to-DPROP mapping, DPROP-to-user mapping, or for a DataRefresher definition call.	X'20'
UDTPROGM	Name of the calling component	X'2C'
UDTSTYPE	User data type (data type of the field in its user format)	X'54'
UDTSBYTV	Number of bytes of field in its user format	X'58'
UDTSSCLV	Value of the scale of field in its user format	X'5E'
UDTTTYPE	DPROP data type (data type of the field in its DPROP format)	X'60'
UDTTBYTV	Number of bytes of field in its DPROP format	X'64'
UDTTSCLV	Value of the scale of field in its DPROP format	X'6A'
UDTXRETC	Return code that your exit routine provides	X'108'
UDTXMESG	exit routine message text	X'10C'
UDTSCRT1	exit routine work space	X'6C'
UDTENTRD	Indicates the exit routine has been entered	X'104'

Figure 27 (Page 2 of 2). Interface Control Block Parameters for Field Exits

Field	Used For	Displacement
UDTINCTL	Indicates the exit routine has control	X'105'
UDTNULLT	Flag indicating conversion into or from a DB2 NULL	X'106'

The interface control block has the same structure as the control block that DataRefresher passes to its User Data Type exits. A more complete description of these fields is included in the copy of the control block in Figure 28 on page 115.

When called for propagation, and for DataRefresher extraction, your exit routine must not change any fields in the control block except:

- UDTXRETC, UDTXMESG, UDTSCRT1, UDTENTRD, UDTINCTL, and UDTNULLT
- UDTTBYTV (when performing user-to-DPROP mapping for a field having a VC or VG DPROP format)
- UDTSBYTV (when performing DPROP-to-user mapping)

Altering any of the other fields in the control block can cause unpredictable results.

When DataRefresher calls it for DEFINITION calls, your exit routine needs to update additional fields. Refer to the appropriate DataRefresher or DXT documentation for more details.

User Format Buffer

The user format buffer contains the field in its User Format:

- When performing user-to-DPROP mapping, DataRefresher or DPROP provides the field to your field exit routine in this buffer. The field is still in its IMS format or in the format your Segment exit routine returns.

  Your field exit routine must not modify this buffer when called to perform user-to-DPROP mapping.

- When performing DPROP-to-user mapping, your field exit routine must provide the field to DPROP in this buffer. The field must be provided in its user format. This is the IMS format or the format your Segment exit routine expects.

  Do not place a field in the user format buffer that is longer than the fixed or maximum field length specified in UDTSBYTV. This causes storage overlays and unpredictable results.

DPROP Format Buffer

The DPROP format buffer contains the field in the DPROP-supported format that you identified during your PR definition:

- When performing user-to-DPROP mapping, your exit routine must convert the field to the format you have defined in your PR, and place the converted field in the DPROP format buffer before returning to the RUP. The RUP reads the field from this buffer and then continues its normal processing as if the converted field were the original.

Do not place a field in the DPROP format buffer that is longer than the fixed or maximum field length specified in UDTTBYTV. This causes storage overlays and unpredictable results.

If the data type of the converted field is VC or VG, then the DPROP format buffer begins with the first data byte of the field (*not* with the field length). When converting to a VC or VG format, the actual field length (expressed in number of bytes) must be set by your exit routine in the UDTTBYTV field of the interface control block.

- When performing DB2-to-IMS mapping, the HUP provides the field for your Field exit routine in this buffer. The provided field is in the DPROP-supported format that you specified during PR definition. DATE and TIME fields are provided by the HUP in ISO format.

  If the DPROP-supported data type is VC or VG, then the DPROP format buffer begins with the first data byte of the field (*not* with the field length). The actual field length (expressed in number of bytes) is provided by DPROP to your exit routine in the UDTTBYTV field of the interface control block.

  Your field exit routine must not modify this buffer when being called to perform DPROP-to-user mapping.

64-Byte Anchor Area

The RUP or HUP gives you 64 bytes as a general storage area. Each exit routine has its own unique anchor area. You can use it for whatever you want. Initially, the area is set to all binary zeros, and is never changed again by DPROP (or DataRefresher if you are using it).

The anchor area exists in virtual storage, and remains yours for the duration of the exit. For IMS batch and BMP regions, the anchor area lasts for the duration of the application program. For MPP regions, the anchor area lasts for the duration of the IMS Program Controller Subtask. This can span multiple MPP executions. For CCU execution, the anchor area lasts for the duration of the job step. For asynchronous propagation, the anchor area lasts for the duration of the MVS task being used by the receiver program to call the RUP.

Interface Control Block DSECT

You can generate the following DSECT in your assembler exit routine by coding the EKYRCUDT macro statement. For high-level language exit routines, you can include or copy one of the following members to map the Field exit routine interface control block:

EKYRCUDC Exit routines written in COBOL
EKYRCUDP Exit routines written in PL/I
EKYRCUDK Exit routines written in C

The interface control block is shown in Figure 28 on page 115 followed by detailed descriptions of its fields.

```
          EKYRCUDT
******************* START OF CONTROL BLOCK SPECIFICATION **************/
*                                                                     */
*       CONTROL BLOCK NAME:                                           */
*         EKYRCUDT (UDT)                                              */
*                                                                     */
*       DESCRIPTIVE NAME:                                             */
*         DPROP FIELD EXIT INTERFACE BLOCK                            */
*                                                                     */
*                                                                     */
**********************************************************************
*                                                                    *
*         THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
*                                                                    *
*         5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
*         ALL RIGHTS RESERVED.                                       *
*                                                                    *
*         U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
*         USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
*         GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
*                                                                    *
*         LICENSED MATERIALS - PROPERTY OF IBM.                      *
*                                                                    *
**********************************************************************
*                                                                     */
*       STATUS: V1 R2 M0                                              */
*                                                                     */
*       FUNCTION:                                                     */
*         THIS IS THE CONTROL BLOCK USED TO INTERFACE BETWEEN         */
*              - DPROP OR DXT                                         */
*              AND                                                    */
*              - A USER'S FIELD EXIT ROUTINE (THESE USER              */
*                EXIT ROUTINES ARE CALLED BY DXT 'USER DATA TYPE      */
*                EXIT ROUTINES')                                      */
*                                                                     */
*         THERE IS ONE UDT CB FOR EACH USER FIELD                     */
*         EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT          */
*         IN VIRTUAL STORAGE.                                         */
*         FOR SYNCH PROPAGATION IN MPP REGIONS:                       */
*         - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER        */
*           SUBTASK.                                                  */
*         FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR             */
*         ASYNCH PROPAGATION, AND FOR CCU PROCESSING:                 */
*         - THIS IS THE DURATION OF THE JOBSTEP.                      */
*                                                                     */
*---------------------------------------------------------------------*/
*       IMPORTANT NOTES:                                              */
*       ================                                              */
*       - SINCE THE SAME USER EXIT ROUTINE CAN BE INVOKED BOTH        */
*         BY DPROP AND BY DXT: CHANGES TO THIS CONTROL BLOCK MUST     */
*         BE COORDINATED BETWEEN DPROP DEVELOPMENT AND DXT            */
*         DEVELOPMENT.                                                */
*                                                                     */
*       - FIELDS MARKED IN THE COMMENT WITH '***DXT ONLY***'          */
*         HAVE NO MEANING, WHEN THE FIELD USER EXIT                   */
*         ROUTINE IS INVOKED BY DPROP.                                */
*---------------------------------------------------------------------*/
*                                                                     */
*       MODULE TYPE= MACRO                                            */
*          PROCESSOR= ASSEMBLER H                                     */
*                                                                     */
*       INNER CONTROL BLOCKS: NONE                                    */
*                                                                     */
*       MACROS USED FROM MACRO LIBRARY: NONE                          */
*                                                                     */
```

Figure 28 (Part 1 of 4). Interface Control Block for a Field Exit Routine

```
                                  66++       CHANGE ACTIVITY:                                          */
                                  67++                     KMP0057  12/13/90                           */
                                  68++                     KMP0060  02/08/91 COPYRIGHT INFORMATION     */
                                  69++                                                                 */
                                  70+******************** END OF CONTROL BLOCK SPECIFICATION **************/

000000                            72+UDT      DSECT
                          00000   73+DVRXCUDT EQU   *                       LABEL FOR DXT COMPATIBILITY
                                  74++*-----------------------------------------------------------------*
                                  75++*   THIS SECTION OF THE CB MAY NOT BE MODIFIED BY EXIT            *
                                  76++*-----------------------------------------------------------------*
000000                            77+UDTPFX   DS    0CL32                   DXT PREFIX
000000                            78+UDTTNAME DS    CL8                     NAME OF BLOCK, "DVRXCUDT"
000008                            79+UDTXADDR DS    AL4                     ADDRESS OF LOADED ROUTINE
00000C                            80+         DS    CL20                    RESERVED FOR DXT USE
                                  81++*
000020                            82+UDTPFXE  DS    0CL300                  PREFIX EXTENSION
000020                            83+UDTPNMOD DS    0CL76
000020                            84+UDTCALL  DS    CL2                     TYPE OF CALL TO EXIT...
                                  85++*                                      'ST' - "SRC->TRG" CALL ISSUED
                                  86++*                                           BY DXT AND BY
                                  87++*                                           DPROP DURING HR MAPPING.
                                  88++*                                           EXIT SHOULD CONVERT THE
                                  89++*                                           DATA FROM THE
                                  90++*                                           - USER FORMAT
                                  91++*                                           TO THE
                                  92++*                                           - DPROP FORMAT
                                  93++*                                      'TS' - "TRG->SRC" CALL
                                  94++*                                           ISSUED BY DPROP DURING RH
                                  95++*                                           MAPPING. EXIT SHOULD
                                  96++*                                           CONVERT DATA FROM THE
                                  97++*                                           - DPROP FORMAT
                                  98++*                                           TO THE
                                  99++*                                           - USER FORMAT
                                 100++*                                           NOT ISSUED BY DXT
                                 101++*                   ***DXT ONLY***     'DF' - "DEFINITION CALL".
                                 102++*                                           **NOT** ISSUED BY DPROP.
                                 103++*                                           DEFINITION CALL ISSUED
                                 104++*                                           BY DXT/UIM FOR EACH DATA
                                 105++*                                           TYPE. EXIT CAN VALIDATE
                                 106++*                                           REQUEST AND RETURN
                                 107++*                                           REQUIRED VALUES.
                                 108++*                                           NOT ISSUED BY DPROP.
000022                           109+         DS    CL2                     RESERVED FOR DXT USE
                                 110++*
000024                           111+UDTENVRN DS    0CL12                   ENVIRONMENTAL INFORMATION
                                 112++*
000024                           113+UDTOPSYS DS    CL4                     OPERATING SYSTEM:
                                 114++*                                      =C'ESA ' IF MVS/ESA
                                 115++*                     ***DXT ONLY***   =C'XA  ' IF MVS/XA
                                 116++*                     ***DXT ONLY***   =C'MVS ' IF MVS
                                 117++*
000028                           118+UDTTRANS DS    CL4                     DB/DC ENVIRONMENT:
                                 119++*                                      =C'BAT ' IF IMS BATCH/BMP
                                 120++*                                      =C'MPP ' IF IMS MP
                                 121++*                                      =C'IFP ' IF FAST PATH
                                 122++*                                      =C'CICS' IF CICS
                                 123++*                                      =C'    ' IF NONE OF ABOVE
                                 124++*
00002C                           125+UDTPROGM DS    CL4                     CALLING PROGRAM:
                                 126++*                                      =C'DXT ' IF DXT
                                 127++*                                      =C'DPRS' IF DPROP SYNCH PROP
                                 128++*                                      =C'DPRA' IF DPROP ASYNCH PROP
                                 129++*                                      =C'DPRC' IF DPROP CCU PROCESSING
                                 130++*                                      =C'DPRL' IF DPROP DLU
```

Figure 28 (Part 2 of 4). Interface Control Block for a Field Exit Routine

```
                                 131+*
000030                           132+UDTEXIT  DS    CL8                  NAME OF THE USER EXIT
                                 133+*
000038                           134+UDTPCBLS DS    AL4  ***DXT ONLY***    ADDRESS LIST OF ALL PCB
                                 135+*                                      ADDRESSES IF DL/I ENVIRONMENT
                                 136+*
00003C                           137+UDTDPRP1 DS    CL24                 ADDITIONAL WORK SPACE
                                 138+*-----------------------------------------------------------------
                                 139+*   THIS SECTION CONTAINS DATA PERTINENT TO FIELD IN ITS USER FORMAT
                                 140+*-----------------------------------------------------------------
000054                           141+UDTSTYPE DS    CL2                  USER DATA TYPE
000056                           142+UDTSBYTI DS    CL1  ***DXT ONLY***    LENGTH INDICATOR FOR USER FORMAT.
                                 143+*                                     NOT USED BY DPROP. USED BY DXT.
                                 144+*                                       'N' - INDICATES LENGTH OF USER
                                 145+*                                             FORMAT RESIDES WITH THE
                                 146+*                                             DEFINITION.
                                 147+*                                       'V' - INDICATES LENGTH OF USER
                                 148+*                                             FORMAT VARIES, AND MUST
                                 149+*                                             BE RETURNED AT
                                 150+*                                             "DEFINITION" TIME.
000057                           151+         DS    CL1                  RESERVED FOR DXT USE
000058                           152+UDTSBYTV DS    H                    LENGTH OF FIELD IN USER FORMAT
                                 153+*
00005A                           154+UDTSSCLI DS    CL1  ***DXT ONLY***    SCALE INDICATOR FOR USER FORMAT
                                 155+*                                     NOT USED BY DPROP. USED BY DXT.
                                 156+*                                       'N' - INDICATES VALUE OF SCALE
                                 157+*                                             OF USER FORMAT
                                 158+*                                             RESIDES WITH THE
                                 159+*                                             DEFINITION.
                                 160+*                                       'V' - INDICATES VALUE OF SCALE
                                 161+*                                             OF USER FORMAT
                                 162+*                                             VARIES, AND MUST
                                 163+*                                             BE RETURNED AT
                                 164+*                                             "DEFINITION" TIME.
00005B                           165+         DS    CL3                  RESERVED FOR DXT USE
00005E                           166+UDTSSCLV DS    H                    VALUE OF SCALE IN USER FORMAT
                                 167+*-----------------------------------------------------------------*
                                 168+*   THIS SECTION CONTAINS DATA PERTINENT TO FIELD IN ITS DPROP FORMAT*
                                 169+*-----------------------------------------------------------------*
000060                           170+UDTTTYPE DS    CL2                  DATA TYPE OF DPROP FORMAT
000062                           171+UDTTBYTI DS    CL1  ***DXT ONLY***    LENGTH INDICATOR FOR DPROP FORMAT
                                 172+*                                     NOT USED BY DPROP. USED BY DXT:
                                 173+*                                       'N' - INDICATES LENGTH OF DPROP
                                 174+*                                             FORMAT RESIDES WITH THE
                                 175+*                                             DEFINITION.
                                 176+*                                       'V' - INDICATES LENGTH OF DPROP
                                 177+*                                             FORMAT VARIES, AND MUST
                                 178+*                                             BE RETURNED AT
                                 179+*                                             "DEFINITION" TIME.
000063                           180+         DS    CL1                  RESERVED FOR DXT USE
000064                           181+UDTTBYTV DS    H                    LENGTH OF FIELD IN DPROP FORMAT
000066                           182+UDTTSCLI DS    CL1  ***DXT ONLY***    SCALE INDICATOR FOR DPROP FORMAT
                                 183+*                                     NOT USED BY DPROP. USED BY DXT.
                                 184+*                                       'N' - INDICATES VALUE OF SCALE
                                 185+*                                             OF DPROP FORMAT
                                 186+*                                             RESIDES WITH THE
                                 187+*                                             DEFINITION.
                                 188+*                                       'V' - INDICATES VALUE OF SCALE
                                 189+*                                             OF DPROP FORMAT
                                 190+*                                             VARIES, AND MUST
                                 191+*                                             BE RETURNED AT
                                 192+*                                             "DEFINITION" TIME.
000067                           193+         DS    CL3                  RESERVED FOR DXT USE
00006A                           194+UDTTSCLV DS    H                    VALUE OF SCALE IN DPROP FORMAT
```

Figure 28 (Part 3 of 4). Interface Control Block for a Field Exit Routine

```
                                     195+*-------------------------------------------------------------------*
                                     196+*   THIS SECTION IS THE COMMUNICATIONS AREA BETWEEN                 *
                                     197+*      - THE EXIT                                                   *
                                     198+*      AND                                                          *
                                     199+*      - DPROP/DXT.                                                 *
                                     200+*-------------------------------------------------------------------*
00006C                               201+UDTXICOM DS    0CL224                DEFINE A COMMUNICATIONS AREA
00006C                               202+UDTDPRP2 DS    CL24                  RESERVED
000084                               203+UDTSCRT1 DS    CL128                 USER EXIT WORK AREA
000104                               204+UDTENTRD DS    CL1                   'ENTERED' FLAG - SET TO 'X' BY
                                     205+*                                     EXIT TO INDICATE THAT DATA
                                     206+*                                     TYPE ROUTINE HAS BEEN ENTERED
000105                               207+UDTINCTL DS    CL1                   'IN-CONTROL' FLAG - SET TO 'X'
                                     208+*                                     BY EXIT TO INDICATE THAT DATA
                                     209+*                                     TYPE ROUTINE IS IN CONTROL
000106                               210+UDTNULLT DS    CL1                   NULL DATA RETURNED FROM EXIT?
                                     211+*                                        'Y' - DATA IS NULL
                                     212+*                                        'N' - DATA IS NOT NULL
000107                               213+         DS    CL1                   RESERVED
000108                               214+UDTXRETC DS    F                     USER EXIT RETURN CODE
                                     215+*                                          0 - SUCCESSFUL COMPLETION
                                     216+*                                      OTHER - ERROR ENCOUNTERED.
                                     217+*                                              IF CALLER IS DPROP:
                                     218+*                                              =4: RUP WILL USE
                                     219+*                                                 ITS USUAL ERROR
                                     220+*                                                 HANDLING LOGIC.
                                     221+*                                              NOT =4: RUP ABENDS
00010C                               222+UDTXMESG DS    CL64                  USER EXIT MESSAGE TEXT
                                     223+*                                      INSERTED INTO DPROP/DXT MESSAGE
                                     224+*                                      IF CALLER IS DPROP:
                                     225+*                                         TEXT WILL BE INSERTED
                                     226+*                                         INTO MSG EKYR970I/EKYR971E.
                             0014C   227+UDTEND   EQU   *                     END OF UDT
                             0014C   228+UDTLEN   EQU   *-UDT                 LENGTH OF UDT
                                     229          END
```

Figure 28 (Part 4 of 4). Interface Control Block for a Field Exit Routine

Interface Control Block Field Descriptions

The following list includes detailed descriptions of the fields in the interface control block. DPROP and DataRefresher descriptions are included. Some of the fields are not useful to your exit routine when DPROP calls it. These fields are described for DataRefresher only.

UDTTNAME Contains the constant **DVRXCUDT**, used to identify the control block in a storage dump.

UDTXADDR The virtual storage entry point address of the exit routine.

UDTCALL The call function that describes what action your exit routine must perform. This field can have the following values:

ST Source to target conversion (user-to-DPROP mapping).

The exit routine is called to convert the field from its user format to its DPROP format. The user format is the format in the IMS segment, or the format that your Segment exit routine provides, if one was called. The DPROP (or DataRefresher) format is defined either on the TRGTYPE= keyword of the DataRefresher CREATE DATATYPE statement, or in the FLDETYPE column of the DPRIFLD MVG input table.

TS Target to source conversion (DPROP-to-user mapping).

The exit routine is called to convert the field from its DPROP format to its user format. The user format is the format in the IMS segment, or the format your Segment exit routine expects, if one is called. The DPROP (or DataRefresher) format is defined either on the TRGTYPE= keyword of the DataRefresher CREATE DATATYPE statement, or in the FLDETYPE column of the DPRIFLD MVG input table.

DF DEFINITION call (*DataRefresher only*).

DataRefresher calls the exit routine to complete or validate a field definition. If you provide all your mapping definitions through the MVG input tables, then your Field exit routine is never called with the DF call function.

The remaining descriptions are only for ST and TS calls. For more information on calls that only DataRefresher uses, refer to the appropriate DataRefresher or DXT documentation.

UDTOPSYS Contains the constant **ESA**, indicating that the program is running in an MVS environment.

UDTTRANS Contains a label describing the environment in which the exit routine is called. This field can have the following values:

BAT	IMS batch or BMP environment
MPP	IMS MPP environment
IFP	IMS Fast Path environment
CICS	CICS environment

If the exit is called in an environment other than those listed above, the value consists of blanks.

UDTPROGM Contains information about the calling program, either DPROP or DataRefresher. This field can have the following values:

DPRS	Called by DPROP during synchronous propagation
DPRA	Called by DPROP during LOG-ASYNC propagation or user asynchronous propagation
DPRC	Called by DPROP during CCU execution
DPRL	Called by DPROP during DLU execution
DataRefresher	Called by DataRefresher

UDTEXIT The load module name of the Field exit routine.

The next five fields describe the user format of the propagated field. The user format is the IMS DB format if you do not use Segment exit routines. If you do use a Segment exit routine, it is the format your Segment exit routine creates (HR-propagation) or expects (RH-propagation).

UDTSTYPE The user data type that was specified either on the SCRTYPE keyword of the DataRefresher CREATE DATATYPE statement, or in the DATATYPE column of the DPRIFLD MVG input table.

UDTSBYTI (*DataRefresher only*) Used for DF calls. Refer to the appropriate DataRefresher or DXT documentation for a complete description.

UDTSBYTV The length (in bytes) of the field in its user format.

For graphic fields, the number of bytes is twice the number of DBCS characters. This is different from the usual DB2 convention that expresses the length of G and VG columns as the number of DBCS characters.

During user-to-DPROP mapping, UDTSBYTV is initialized on entry to your exit routine to the length specified during PR definition.

Observe the following rules about UDTSBYTV for DPROP-to-user mapping:

- On entry to your exit routine, UDTSBYTV is initialized by DPROP to the length specified during PR definition. During processing, your exit routine can decrease (but not increase) the UDTSBYTV value to the actual length of the field in its user format.

UDTSSCLI (*DataRefresher only*) Used for DF calls. Refer to the appropriate DataRefresher or DXT documentation for a complete description.

UDTSSCLV The scale of the field in its user format.

The next five fields describe the field in its DPROP or DataRefresher format.

UDTTTYPE The DPROP or DataRefresher data type that is specified either on the TRGTYPE keyword of the DataRefresher CREATE DATATYPE statement, or in the FLDETYPE column of the DPRIFLD MVG input table.

UDTTBYTI (*DataRefresher only*) used for DF calls. Refer to the appropriate DataRefresher or DXT documentation for a complete description.

UDTTBYTV The length (in bytes) of the field in its DPROP or DataRefresher format.

For graphic fields, the number of bytes is twice the number of DBCS characters. This is different from the usual DB2 convention that expresses the length of G and VG columns as the number of DBCS characters.

Observe the following rules about UDTTBYTV for user-to-DPROP mapping:

- On entry to your exit routine, UDTTBYTV was initialized by DPROP and DataRefresher to the maximum field length (for fields having a VC or VG data type in their DPROP format) or to the fixed length of the field (for fields having data types other than VC and VG in their DPROP format).
- When processing a field having in its DPROP format a VC or VG data type, your exit routine must return the actual length of the field in UDTTBYTV.

During DPROP-to-user mapping, UDTTBYTV is initialized on entry to your exit routine to the actual length of the field.

UDTTSCLI (*DataRefresher only*) Used for DF calls. Refer to the appropriate DataRefresher or DXT documentation for a complete description.

UDTTSCLV The scale of the field in its DPROP or DataRefresher format.

The remaining fields can be changed in your Field exit routine.

UDTSCRT1 An exit routine work space for your own use. Before the first call to your exit routine, DPROP initializes this space to binary zeros, and does not modify it again.

The next two fields are switches that can be useful for problem determination. DPROP and DataRefresher do not require your exit routine to set these fields. However, they can help you determine where a problem occurred if you have an ABEND. DPROP and DataRefresher set these fields to blanks before the first time your exit routine is called.

UDTENTRD Exit-entered flag. As you enter your exit routine, set this field to **X**. DPROP does not change this field again, so if a problem occurs, you can determine if your exit has been entered.

UDTINCTL Exit-in-control flag. You can also set this field to **X**, indicating that your exit routine has control. When DPROP regains control, it resets this field to blank, so you can determine if your exit routine has control when an ABEND occurs.

The next field supports conversion to or from a DB2 null value.

UDTNULLT Null value indicator. This field allows you to map the contents of the IMS field to a DB2 NULL value (user-to-DPROP mapping) or from a DB2 NULL value (DPROP-to-user mapping).

- When your exit routine is called with an ST Call Function (user-to-DPROP mapping), DPROP initializes this field to blanks. If your exit routine sets UDTNULLT to **Y** during an ST call, DPROP or DataRefresher assigns a NULL value to the target DB2 column. You must define the target column as containing a NULL value.
- When your exit routine is called with a TS Call Function (DPROP-to-user mapping), this field indicates whether or not the DB2 column contains a NULL value.

  Y The DB2 column contains a NULL value.
  N The DB2 column does not contain a NULL value.

The next two fields can be used along with the RUP's and HUP's error handling logic. For more information on return codes and error handling techniques, see “Return Codes and Error Handling Techniques” on page 123.

UDTXRETC The return code the exit routine provides when returning to its caller. This field is set to zero when the exit routine is called.

UDTXMESG User-provided error message. It is set to blanks when the exit routine is called. When the exit routine returns, if this field is not blank, DPROP or DataRefresher writes the contents of the field.

DPROP prefaces the message with the number EKYR970I or EKYR971E, and writes the message according to its usual error handling logic. DataRefresher prefaces the message and writes the message to the //SYSPRINT data set.

There is one exception to the above. If the exit routine is called during processing of the optional WHERE clause of the PR, DPROP does not write error messages if the exit returns with a return code 0 or 4 in UDTXRETC.

Performing Data Conversions

The appropriate *Administrators Guide* for your propagation mode lists all the data conversions that are supported directly through the DPROP data conversion routines. If the IMS data field you want to propagate is in a format DPROP does not support, or if you want to perform a conversion that the DPROP data conversion routines do not support, your Field exit routine can perform the conversion. Examples of conversions that are not supported include binary integers to floating-point numbers, and time stamp to date or time formats.

Your exit routine does not need to convert a field directly into or from the format of the DB2 column. The DPROP data conversion routines are still called (if necessary) to complement the conversion done by your field exit routine. The RUP and HUP call the DPROP data conversion routines automatically if the DPROP format of the data field is different from the DB2 column format. Therefore, it is sufficient if your exit routine converts the data field between its user format and a DPROP-supported format.

Exit Routine Processing

When called for user-to-DPROP mapping, your Field exit routine can read the IMS field from the user format buffer and, using the information found in the interface control block, convert the field into the DPROP format you have defined in your PR. For more details on defining user and DPROP formats, see "Telling DPROP About Your Field Exit Routine" on page 125.

When called for DPROP-to-user mapping, your Field exit routine can read the field from the DPROP format buffer and, using the information found in the interface control block, convert the field into the user format.

Meanwhile, there are some restrictions and guidelines to follow when developing your exit routine.

- When DPROP calls it, your exit routine always gets control in AMODE 31, and must return control in AMODE 31. Parameters DPROP passes to your exit are usually located above the 16MB line. The exit routine is loaded above or below the 16MB line depending on the RMODE attribute of the exit load module.

  It is recommended that you code and link-edit your exit routine as reentrant. To simplify programming, DPROP provides work space for your exit routines, both in the interface control block and the 64-byte anchor area.

- If your exit routine is written in Assembler language, DPROP uses standard OS/VS conventions when calling your exit routine.

  – Register 1 points to the parameter list described above.
  – Register 13 contains the address of a register save area.
  – Register 14 contains the return address.
  – Register 15 contains the entry point address of the exit routine

  Upon entry, the exit routine must save the register contents into the save area that the caller provides. If your exit routine calls other routines that use standard MVS linkage conventions, it must also provide a save area of its own. The exit routine must return to its caller using normal OS/VS conventions after restoring the registers. A return code must be provided in the interface control block, not in register 15.

- Your Field exit routine must check that the data returned to DPROP is valid. For example, it must make sure that a packed field contains a number in packed format. Conversions producing invalid formats can cause propagation or application failures. For example, during HR-propagation, SQL statements that the SQL update module generates, or conversions by DPROP data conversion routines, can be rejected. With RH-propagation, invalid conversion can result in application failures, when your IMS applications access the segments with the invalid data.
- When converting a field that is part of the IMS key or mapped to a primary DB2 key, DPROP cannot verify that the key is still unique after it is converted; you must check it.
- Because the exit routine for synchronous propagation runs in the same environment as the propagating application program, it can, if necessary, generate the same type of IMS calls and SQL statements as the application program. For LOG-ASYNC and user asynchronous propagation using either TSO-Attach or CAF-Attach, create only SQL statements, as the exit routines do not execute in an IMS environment, and cannot generate IMS calls.

  If your exit generates IMS calls, use the AIB interface described in *IMS/ESA Application Programming: DL/I Calls*, which allows your exit routine to generate calls without the address of the IMS PCBs.

  During synchronous propagation, any changes you make to propagated data from within your exit routine are not propagated synchronously. However, the date can be propagated asynchronously, if you implement asynchronous propagation.

  Exclude the PCBs your exit routine uses from the list passed to the application program upon entry. You can avoid changing the application program if you need to add PCBs for exclusive use by your exit routine. Refer to *IMS/ESA Utilities Reference: System* for more details.
- A Field exit routine must not perform functions that are not supported by the environment in which it is running. For example, an exit routine running in an MPP region must not write to OS files, and the exit routine must not generate STIMER macros in an IMS environment.

  For performance reasons, your exit routine must generate static rather than dynamic SQL statements. Avoid using functions that have a detrimental impact on the performance of the propagating program, such as performing an OPEN and CLOSE on an OS/VS file each time the exit is called.

Return Codes and Error Handling Techniques

This section discusses how to return from your exit routine to the RUP and HUP, including return codes and error handling techniques.

Return Codes

You set the return code by placing it in the UDTXRETC field of the interface control block. The RUP and HUP read this field when they regain control. The valid return codes are 0 and 4. Returning any other code is an error and DPROP abends.

0 Normal return. DPROP or DataRefresher continues normal processing using the converted field value.

4 A failure occurred. DPROP uses its usual error handling logic. There is one exception to this: if the exit routine is called during processing of the optional WHERE clause of the PR, DPROP does not perform its error logic. The currently processed condition of the WHERE clause is considered to be false (or true if the operand of the condition is ¬=).

For synchronous propagation,

if ERROPT=BACKOUT is in effect, DPROP backs out the propagating application. For LOG-ASYNC propagation, if ERROPT=BACKOUT is in effect, the Receiver terminates with an error message. For user asynchronous propagation, CCU or DLU execution, the RUP and HUP return to their caller with an error. The RUP and HUP use their error reporting logic to write diagnosis information.

If ERROPT=IGNORE is in effect, the RUP and HUP do not perform propagation, and return to the caller without performing a backout and without providing any error indication to the caller. However, if this occurs during CCU or DLU execution, the RUP and HUP return to the CCU or DLU with an error. The RUP and HUP use their error reporting logic to write diagnosis information.

For DataRefresher, further processing is based on the FLDERR keyword of the DataRefresher SUBMIT control statement.

Error Handling Techniques

When you find an error in your exit routine, it is strongly recommended that your exit routine take advantage of the standard error handling logic of the RUP and HUP. In the interface control block, you can supply a return code in UDTXRETC, and an error message in UDTXMESG. You must not return an error message in UDTXMESG without providing an error return code, because this generates excessive console messages.

By supplying DPROP with an error return code and message, you gain many advantages. When an exit returns with an error return code, DPROP traces or snaps both the control blocks involved in the interface, and the data. The exits are included in the standardized error handling scheme of the RUP and HUP, which distinguishes between ERROPT=BACKOUT and ERROPT=IGNORE; this is different for propagation and CCU executions. It protects against excessive console messages. DPROP writes your error message using its standard message writing logic: WTO, trace data set (the IMS log, the //EKYTRACE data set, or the //EKYLOG data set), and Audit trail.

If the exit routine generates its own messages or ABENDs, the RUP and HUP cannot include the exit routine in their standardized error handling, and cannot guard against excessive messages on the MVS consoles. Therefore, it is recommended that your exit routine does not generate its own messages or ABENDs when an error occurs.

Saving Information Across Calls

You can save information across calls to the exit routine. You can save the information either in the 64-byte anchor area, or in the field UDTSCRT1 in the interface control block. If these areas are not large enough, generate a GETMAIN and save the address of the storage in either of these areas.

Updating Your Field Exit Routine

DPROP does not provide any online change logic to replace an existing load module copy of your Field Exit routine with a new version of the load module. If you need to change your exit routine, stop the affected IMS regions, DPROP asynchronous Receiver or user asynchronous receiver programs before performing the change. A change of the exit routine without stopping the IMS regions or receiver programs causes unpredictable results. For example, some MPP regions can use the new version of the exit routine, while other regions use the old version. After the change, you can restart the IMS regions.

Tracing Your Exit Routine

As described in "Tracing Your Exit Routine" on page 41, DPROP provides a trace facility to help you debug your exit routine. For a Field exit routine, DPROP includes in the trace output the user format buffer and DPROP format buffer, rather than the segment buffers.

Telling DPROP About Your Field Exit Routine

This section discusses how to inform DPROP that you want to use a Field exit routine during data propagation. To do this, you must provide DPROP with the name of your exit routine, the two-byte user data type, and the description of the DPROP-supported field format. How you proceed depends on how you enter your PR.

PRs Entered Through DataRefresher UIM

Defining the User Data Type

If you are using both DataRefresher and a Field exit routine, define a *user data type* before calling the exit routine.

Use the DataRefresher CREATE DATATYPE control statement to define the user data type and associate it with a Field exit routine. You define a user data type by assigning it a unique two-byte name using CREATE DATATYPE. You also specify the name of the exit routine on the control statement.

Usually, one CREATE DATATYPE control statement is used for each Field exit routine. But you can use multiple CREATE DATATYPE statements specifying multiple definitions for one exit routine. In this case, your exit routine is responsible for converting the multiple user data types.

You must provide the following keywords on the CREATE DATATYPE statement when calling DataRefresher:

EXIT=exitname
The load module name of the Field exit routine.

SRCTYPE=xx
A two-byte character value used to uniquely identify the user data type and associate it with the exit routine.

The second character of the two-byte value cannot be blank, and the value cannot be VC or VG.

SRCBYTES=nn
The length in bytes of the field in its user format.

SRCSCALE=mm
Optional; the scale of the field in its user format.

TRGTYPE=yy
The data type of the DataRefresher or DPROP format.

It must be a data type that DataRefresher and DPROP support.

TRGBYTES=bb
The length in bytes of the field in its DataRefresher or DPROP format.

For variable-length character and graphic fields, specify the maximum length.

TRGSCALE=ss
The scale of the field in its DataRefresher or DPROP format.

This keyword is used only for packed decimal and zoned decimal data types.

Requesting Exit Routine Use

After defining a user data type, you can request the use of the associated Field exit routine. Specify if the exit routine must process a field by using the FIELD statement of the CREATE DXTPSB control statement.

Identify the user data type on the TYPE= keyword of the FIELD statement. This is the same data type specified earlier in the SRCTYPE= keyword of the CREATE DATATYPE control statement. Each FIELD statement identifying a user data type with the TYPE= keyword is processed by the exit routine.

DataRefresher calls the Field exit routine with a definition call each time it processes a FIELD statement identifying a user data type. During the definition call, the exit routine can validate or change the field definitions provided by the CREATE DATATYPE statement and the FIELD statement. Definition calls are generated when DataRefresher processes your CREATE DXTPSB control statement, not during the extract.

DataRefresher calls your Field exit routine during the extract; DPROP calls it during propagation. During these calls, your exit routine must convert the fields between the user data type and the data type supported by DataRefresher and DPROP.

For more information on DataRefresher calls, see the appropriate DataRefresher or DXT documentation.

PRs Entered into the MVG Input Tables

If you are entering your PR information directly into the MVG input tables, without using DataRefresher, you can use the DPRIFLD (or FLD) table to inform DPROP if your Field exit routine must process a particular field. The FLD table is one of the MVG input tables. Provide the information in the following columns:

FLDEXIT=exitname
The load module name of the Field exit routine.

The name must begin with an alphabetic character. If you insert blanks into the FLDEXIT column or leave the column blank, then the field described in the DPRIFLD row is not processed by your exit routine.

DATATYPE=xx
A two-byte character value used to uniquely identify the user data type, and to associate the exit routine with this data type.

The second character of the two-byte value cannot be blank, and cannot be VC or VG.

BYTES=nn
The length in bytes of the field in its user format.

SCALE=mm
Optional; the scale of the field in its user format.

FLDETYPE=yy
The data type of the DPROP format.

This value must be a data type that DPROP supports.

FLDEBYTE=bb
The length in bytes of the field in its DPROP format.

For variable-length character and graphic fields, specify the maximum length.

FLDESCAL=ss
The scale of the field in its DPROP format.

This keyword is used only for packed decimal and zoned decimal data types.

The MVGU validates all of these columns for a general mapping case. For a PR entered into the MVG input tables, the Field exit routine is not called for a definition call.

First Sample Field Exit Routine

The sample Field exit routine in Figure 29 on page 128 is an example of how to convert the data type of an individual field. In this case, the exit routine converts a bit string field into a character field (during user-to-DPROP mapping) and a character field into a bit string (during DPROP-to-user mapping).

Specifically, during user-to-DPROP mapping, each bit is converted into a character represented by 0 or 1 based on the value of the related bit. This can be useful to convert bit control fields into individual flag character fields.

The source code in Figure 29 on page 128 is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYEFL1A. Following the source code are definitions related to the sample Field exit routine.

```
         PRINT NOGEN
*/******************************************************************/
*/*                                                                */
*/*   MODULE NAME: EKYEFL1A                                        */
*/*                                                                */
*/*   DESCRIPTIVE NAME: SAMPLE DPROP 'FIELD USER EXIT ROUTINE'     */
*/*                                                                */
*/*   STATUS: V1 R2 M0                                             */
*/*                                                                */
*/*   FUNCTION: THE PURPOSE OF THIS PROGRAM IS TO PROVIDE A SAMPLE */
*/*             STRUCTURE FOR A 'FIELD  USER EXIT ROUTINE'.        */
*/*                                                                */
*/*             THIS EXAMPLE CONVERTS A BIT STRING INTO A CHARACTER */
*/*             STRING (OR VICE VERSA) WITH EACH BIT REPRESENTED BY */
*/*             A CHARACTER, TO BE SET O '1' OR '0' BASED ON THE   */
*/*             VALUE OF THE RELATED BIT (ALTERNATE REPRESENTATION */
*/*             MIGHT BE 'T' FOR TRUE AND 'F' FOR FALSE). THIS     */
*/*             FUNCTION COULD BE USEFUL FOR CONVERTING BIT CONTROL */
*/*             FIELDS TO INDIVIDUAL FLAG BYTES.                   */
*/*                                                                */
*/*             IN INSTALLATIONS WHICH COMBINE USAGE OF:           */
*/*                - DXT (FOR THE ORIGINAL EXTRACT OF THE DL/I     */
*/*                       DATA).                                   */
*/*                - DPROP (FOR THE PROPAGATION OF THE DL/I DATA)  */
*/*             THE EXIT WILL BE CALLED BOTH BY DXT AND DPROP.     */
*/*                                                                */
*/*             DXT CALLS THE EXIT:                                */
*/*                - DURING DXT-UIM PROCESSING, WITH A 'DEFINITION */
*/*                  CALL' IN ORDER TO VALIDATE FIELD DEFINITIONS. */
*/*                - DURING DXT-DEM PROCESSING, IN ORDER TO MAP    */
*/*                  DURING THE DL/I DATA EXTRACT BIT-STRINGS INTO */
*/*                  CHARACTER-STRING.                             */
*/*                                                                */
*/*             DPROP CALLS THE EXIT:                              */
*/*                - FOR 'SOURCE-TO-TARGET' (ST) CONVERSION, TO    */
*/*                  MAP THE BIT-STRINGS INTO CHARACTER STRINGS:   */
*/*                  - DURING DATA PROPAGATION                     */
*/*                  - DURING DPROP CCU (CONSISTENCY CHECK UTILITY) */
*/*                - FOR 'TARGET-TO-SOURCE' (TS) CONVERSION, TO    */
*/*                  MAP THE CHARACTER STRINGS INTO BIT-STRINGS:   */
*/*                  - DURING DATA PROPAGATION                     */
*/*                  - DURING DPROP CCU (CONSISTENCY CHECK UTILITY) */
*/*                  - DURING DPROP DLU (DL/I LOAD UTILITIES)      */
*/*                                                                */
*/*                                                                */
*/*   PROCESSING: - FOR 'SOURCE-TO-TARGET' CALLS, THE SOURCE FIELD */
*/*                 IS CONVERTED A BIT A TIME INTO '0' OR '1'      */
*/*                 CHARACTERS IN THE TARGET FIELD. FOR EXAMPLE    */
*/*                 THE 2 BYTE CHARACTER STRING 'A1' IS HEX 'C1F1' */
*/*                 OR '1100000111110001' IN BINARY. IT WOULD      */
*/*                 BE CONVERTED INTO THE 16 BYTE CHARACTER STRING */
*/*                 '1100000111110001'. THE LENGTH OF THE TARGET   */
*/*                 FIELD TERMINATES PROCESSING. IF THE TARGET     */
*/*                 LENGTH IS MORE THAN SOURCE LENGTH TIMES 8 THE  */
*/*                 REMAINING RIGHT HAND BYTES ARE SET TO THE      */
*/*                 CHARACTER '0'.                                 */
```

Figure 29 (Part 1 of 10). Sample Field Exit Routine (Assembler)

```
*/*            - FOR 'TARGET-TO-SOURCE' CALLS, THE TARGET FIELD   */
*/*              (IN THIS CASE INPUT TO THIS ROUTINE) WHICH MUST  */
*/*              BE ALL CHARACTERS '0' OR '1', IS CONVERTED TO    */
*/*              A BIT STRING WITH THOSE BITS ON, WHICH ARE '1'   */
*/*              IN THE TARGET FIELD. WHEN USING THE ABOVE        */
*/*              EXAMPLE, THE 16 BYTE CHARACTER FIELD WITH THE    */
*/*              VALUE '1100000111110001' WOULD BE CONVERTED TO   */
*/*              THE 2 BYTE BIT-STRING '1100000111110001' WHICH   */
*/*              IS HEX 'C1F1' OR CHAR 'A1'. NOTE THAT VALUES     */
*/*              OTHER THAN '0' AND '1' IN THE CHARACTER TARGET   */
*/*              FIELD LEADS TO CONVERSION ERRORS DETECTED BY     */
*/*              THIS FIELD EXIT ROUTINE.                         */
*/*                                                               */
*/*   PROCESSING DURING 'DEFINITION' CALLS ISSUED BY DXT-UIM:     */
*/*          -  THE  SOURCE LENGTH IS CHECKED  AGAINST THE MAXIMUM */
*/*             SOURCE LENGTH(16).                                */
*/*          -  IF THE TARGET LENGTH HAS BEEN DEFINED ON THE      */
*/*             DXT UIM 'CREATE DATATYPE' STATEMENT AS 'VARIES',  */
*/*             THE EXIT SETS ITS VALUE TO 8 TIMES THE SOURCE     */
*/*             LENGTH.                                           */
*/*             IF THE TARGET LENGTH HAS BEEN SPECIFIED ON THE    */
*/*             DXT UIM 'CREATE DATATYPE' STATEMENT: IT IS        */
*/*             CHECKED AGAINST 8 TIMES THE MAXIMUM SOURCE LENGTH. */
*/*          -  TARGET DATA TYPE IS ENSURED                       */
*/*             TO BE  'C' AND  TARGET SCALE ENSURED TO BE 'N'.   */
*/*                                                               */
*/*          NOTE FOR INSTALLATIONS WHICH USE DPROP WITHOUT DXT:  */
*/*             IF DPROP IS USED WITHOUT DXT, THE EXIT WILL NEVER */
*/*             BE INVOKED FOR A DEFINITION CALL (DEFINITION      */
*/*             CALLS ARE NOT NECESSARY, SINCE THE USER PROVIDES  */
*/*             ALL DEFINITIONS (I.E SOURCE LENGTH, TARGET LENGTH) */
*/*             IN THE DPROP 'MVG INPUT TABLES'.                  */
*/*                                                               */
*/*   PROCESSING DURING 'TARGET-TO-SOURCE' CALLS ISSUED BY        */
*/*             DXT-DEM AND DPROP:                                */
*/*          -  THE DATA IN  THE SOURCE BUFFER IS CONVERTED  A BIT */
*/*             AT A TIME  INTO A '1' OR '0' IN THE TARGET BUFFER. */
*/*             PROCESSING STOPS WHEN THE NUMBER OF BITS PROCESSED */
*/*             EQUALS  THE VALUE PASSED  AS THE TARGET LENGTH. IF */
*/*             THE  SOURCE LENGTH IS EXHAUSTED BEFORE THE TARGET, */
*/*             THE  REMAINING  RIGHT HAND TARGET BYES ARE  SET TO */
*/*             CHARACTER '0'.                                    */
*/*                                                               */
*/*   PROCESSING DURING 'SOURCE-TO-TARGET' CALLS ISSUED BY DPROP: */
*/*          -  THE DATA IN  THE TARGET BUFFER IS CONVERTED FROM  */
*/*             ITS CHARACTER FORMAT  TO A BIT-STRING FORMAT IN   */
*/*             THE SOURCE BUFFER. 8 BYTES OF THE TARGET BUFFER   */
*/*             WILL COMPOSE A BYTE (8 BITS) IN THE SOURCE BUFFER. */
*/*             IF THE TARGET LENGTH IS EXHAUSTED BEFORE ALL      */
*/*             SOURCE BITS ARE PROCEED, THEN THE REMAINING RIGHT */
*/*             HAND BITS ARE ALL SET TO '0'.                     */
*/*                                                               */
*/*  INSTALLATIONS USING BOTH DXT AND DPROP WILL NOTICE           */
*/*  THAT THE LOGIC OF EKYEFL1A IS THE SAME AS                    */
*/*  THE LOGIC OF THE DXT-PROVIDED SAMPLE EXIT ROUTINE            */
*/*  DVRXAXUT, BUT IT IS ENHANCED BY THE 'TARGET-TO-SOURCE'       */
*/*  CALL TYPE WHICH IS ISSUED ONLY BY DPROP.                     */
*/*                                                               */
*/*  SPECIFIC EXIT FUNCTIONS DEMONSTRATED BY THIS MODULE.         */
*/*   1. PROCESSING THE INVOCATION PARM LIST.                     */
*/*   2. USING THE USER ANCHOR AREA.                              */
*/*   3. IDENTIFYING THE REQUESTED FUNCTION.                      */
*/*   4. UIM VALIDATION OF 'V' TYPE LENGTH FIELDS.                */
*/*   5. THE USE OF THE MESSAGE AREA.                             */
*/*                                                               */
```

Figure 29 (Part 2 of 10). Sample Field Exit Routine (Assembler)

```
*/*   INPUT: (PASSED AS PARAMETERS).                                  */
*/*    1. UDT - USER DATA TYPE INTERFACE CONTROL BLOCK.               */
*/*    2. SOURCE BUFFER - THE SOURCE USER DATA (N/A FOR DEFINE CALL).*/
*/*    3. TARGET BUFFER - TARGET AFTER CONVERSION (N/A FOR DEFINE).  */
*/*    4. USER ANCHOR AREA - A 64 BYTE AREA FOR USE BY THE EXIT.     */
*/*                                                                   */
*/*  CHANGE ACTIVITY=                                                 */
*/*                                                                   */
*/*******************************************************************/
*/*******************************************************************/
*/*                                                                   */
*/*       RETURN CODE AND MESSAGES ARE SET IN UDT BLOCK.              */
*/*                                                                   */
*/*       RETURN CODE = 0  PROCESSING SUCCESSFUL - NO MESSAGE SET.    */
*/*                                                                   */
*/*       RETURN CODE = 4  DATA TYPE VALIDATION FAILED - MESSAGE SET.*/
*/*        'SOURCE LENGTH NOT SPECIFIED - REQUIRED.     '             */
*/*        'SOURCE LENGTH EXCEEDS MAXIMUM ALLOWED.      '             */
*/*        'TARGET LENGTH NOT SPECIFIED - REQUIRED.     '             */
*/*        'TARGET LENGTH EXCEEDS MAXIMUM ALLOWED.      '             */
*/*        'TARGET DATA TYPE MUST BE CHARACTER          '             */
*/*        'TARGET SCALE MUST NOT BE SPECIFIED          '             */
*/*        'VALUE IN TARGET FIELD OTHER THAN '0' OR '1' '             */
*/*                                                                   */
*/*                                                                   */
*/*       RETURN CODE =16  UNIDENTIFIED FUNCTION - MESSAGE IS SET.    */
*/*        'DATA TYPE CALL FUNCTION CANNOT BE IDENTIFIED'             */
*/*                                                                   */
*/*******************************************************************/
*/*******************************************************************/
*/*  INFORMATION FOR INSTALLATIONS WHICH COMBINE                      */
*/*  THE USAGE OF DXT AND DPROP.                                      */
*/*                                                                   */
*/*       THESE INSTALLATIONS DEFINE THE DL/I-TO-DB2 MAPPING BY       */
*/*       PROVIDING MAPPING DEFINITIONS TO DXT.                       */
*/*       USAGE OF THIS FIELD EXIT ROUTINE REQUIRES                   */
*/*       FOLLOWING SPECIFICATIONS IN THE DXT 'CREATE DATATYPE'       */
*/*       AND 'CREATE DXTPSB' 'FIELD' STATEMENT:                      */
*/*-----------------------------------------------------------------*/
*/*     INVOCATION OF A FIELD USER EXIT ROUTINE                       */
*/*     IS DEFINED BOTH BY SPECIFICATIONS IN THE DXT                  */
*/*     'CREATE DATATYPE' AND 'CREATE DXTPSB'  'FIELD' STATEMENT.     */
*/*                                                                   */
*/*     THE CREATE DATATYPE;                                          */
*/*     EXIT     = EKYEFL1A - THE EXIT LOAD MODULE NAME               */
*/*     SRCTYPE  = XX       - THE TWO CHARACTER USER DATA TYPE ID.    */
*/*     SRCBYTES = VARIES   - THE SOURCE FIELD LENGTH.                */
*/*                OR NNNN                                            */
*/*                           (MAXIMUM SOURCE LENGTH IS 16 FOR        */
*/*                           THIS SAMPLE. THE EXIT PROGRAM COULD     */
*/*                           HAVE THE LIMIT INCREASED TO 4092.)      */
*/*     TRGTYPE  = C        - MUST BE A 'C' FOR CHARACTER TYPE TARGET*/
*/*     TRGBYTES = VARIES   - THE TARGET FIELD/COLUMN LENGTH.         */
*/*              OR  NNNN                                             */
*/*                           THE TARGET LENGTH SHOULD BE             */
*/*                           8 TIMES THE SOURCE LENGTH.              */
*/*                           IF TRGBYTES IS SPECIFIED AS 'VARIES'    */
*/*                           ON THE 'CREATE DATATYPE', THEN THE      */
*/*                           EXIT WILL SET (DURING THE 'DEFINITION   */
*/*                           CALL') THE TARGET LENGTH TO 8 TIMES     */
*/*                           THE SOURCE LENGTH.                      */
*/*                           (MAXIMUM TARGET LENGTH IS 128 IN        */
*/*                           THIS SAMPLE, BUT THE PROGRAM COULD      */
*/*                           HAVE THE LIMIT INCREASED TO 32,736.)    */
*/*     SRCSCALE =, AND TRGSCALE = MUST NOT BE SPECIFIED.             */
*/*                                                                   */
```

Figure 29 (Part 3 of 10). Sample Field Exit Routine (Assembler)

```
*/*     THE FIELD STATEMENT IN CREATE DXTPSB:                          */
*/*     TYPE     = XX       - RELATES THIS FIELD TO A DXT DATATYPE.  */
*/*     BYTES    = NN       - THE SOURCE FIELD LENGTH.               */
*/*                           IF DEFINED AS 'VARIES' IN THE          */
*/*                           DATATYPE STATEMENT, IT MUST NOT        */
*/*                           EXCEED THE MAXIMUM FIELD LENGTH        */
*/*                           ALLOWED BY THE EXIT.                   */
*/*                           IF NOT DEFINED AS 'VARIES',            */
*/*                           IT MUST EQUAL THE 'SRCBYTES'           */
*/*                           OPERAND IN THE DATATYPE STATEMENT.     */
*/*     SCALE    =  MUST NOT BE SPECIFIED.                           */
*/*                                                                  */
*/*                                                                  */
*/******************************************************************/


*/******************************************************************/
*/*  INFORMATION FOR INSTALLATIONS WHICH USE DPROP WITHOUT DXT.     */
*/*                                                                  */
*/*       THESE INSTALLATIONS DEFINE THE DL/I-TO-DB2 MAPPING BY      */
*/*       PROVIDING MAPPING DEFINITIONS IN THE                       */
*/*       DPROP 'MVG INPUT TABLES'.                                  */
*/*       USAGE OF THIS SAMPLE FIELD EXIT ROUTINE REQUIRES           */
*/*       FOLLOWING DEFINITIONS IN THE DPRIFLD TABLE:                */
*/*----------------------------------------------------------------*/
*/*     INVOCATION OF A FIELD USER EXIT ROUTINE                      */
*/*     IS DEFINED BY PROVIDING  SPECIFICATIONS IN                   */
*/*     THAT ROW OF THE 'DPRIFLD' TABLE WHICH DESCRIBES              */
*/*     THE FIELD TO BE MAPPED.                                      */
*/*                                                                  */
*/*     COLUMNS OF THE DPRIFLD ROW SHOULD PROVIDE                    */
*/*     FOLLOWING DEFINITIONS:                                       */
*/*                                                                  */
*/*     COLUMN OF   COLUMN                                           */
*/*     DPRIFLD     VALUE     EXPLANATIONS                           */
*/*     ---------------------------------------------------------*/
*/*     FLDEXIT   = EKYEFL1A: THE EXIT LOAD MODULE NAME              */
*/*     DATATYPE  = XX      : A TWO CHARACTER DATA-TYPE ID.          */
*/*     BYTES     = NNNN    : THE SOURCE FIELD LENGTH                */
*/*     FLDETYPE  = C       : THE TARGET DATA-TYPE. MUST BE 'C '.    */
*/*     FLDEBYTE  = MMMMM   : THE TARGET FIELD LENGTH.               */
*/*                           (MUST BE 8 TIMES THE SOURCE            */
*/*                           FIELD LENGTH).                         */
*/*     SCALE     =         : SHOULD EITHER NOT BE PROVIDED OR       */
*/*                           SHOULD BE SPECIFIED AS ZERO.           */
*/*     FLDESCAL  =         : SHOULD EITHER NOT BE PROVIDED OR       */
*/*                           SHOULD BE SPECIFIED AS ZERO.           */
*/*                                                                  */
*/*                                                                  */
*/******************************************************************/
*/******************************************************************/
*/*       ASSEMBLER LANGUAGE DEPENDENT SECTION                       */
*/******************************************************************/
*/*                                                                  */
*/*       ENTRY REGISTERS                                            */
*/*              1 - A(PARAMETER LIST)                               */
*/*                  PARAMETER LIST = A(UDT)                         */
*/*                                   A(SOURCE BUFFER)               */
*/*                                   A(TARGET BUFFER)               */
*/*                                   A(EXIT ANCHOR AREA)            */
*/*             13 - CALLER'S SAVE AREA                              */
*/*             14 - CALLER'S RETURN ADDRESS                         */
*/*             15 - ENTRY FOR THIS EXIT ROUTINE                     */
*/*                                                                  */
```

Figure 29 (Part 4 of 10). Sample Field Exit Routine (Assembler)

```
                                            254 */*       EXIT  REGISTERS                                            */
                                            255 */*           0-12 - RESTORED                                        */
                                            256 */*             14 - RESTORED                                        */
                                            257 */*             15 - RETURN CODE - 0 = PROCESSING SUCCESSFUL         */
                                            258 */*                                4 = VALIDATION FAILED             */
                                            259 */*                               16 = UNIDENTIFIED FUNCTION         */
                                            260 */*                                                                  */
                                            261 */*     ATTRIBUTES = REENTRANT                                       */
                                            262 */*     RMODE      = ANY                                             */
                                            263 */*     AMODE      = 31                                              */
                                            264 */*                                                                  */
                                            265 */******************************************************************/

000000                                      267 EKYEFL1A CSECT DPROP FIELD USER EXIT ROUTINE
                                            268 EKYEFL1A AMODE 31
                                            269 EKYEFL1A RMODE ANY
                                            270 *********************************************************************
                                    000F0   271 XDBITOFF EQU   C'0'   VALUE TO BE SET IF RELATED BIT IS OFF
                                    000F1   272 XDBITON  EQU   C'1'   VALUE TO BE SET IF RELATED BIT IS ON
                                            273 *********************************************************************
                                    00000   274 R0       EQU   0    REGISTER  0 EQUATE
                                    00001   275 R1       EQU   1    REGISTER  1 EQUATE
                                    00002   276 R2       EQU   2    REGISTER  2 EQUATE
                                    00003   277 R3       EQU   3    REGISTER  3 EQUATE
                                    00004   278 R4       EQU   4    REGISTER  4 EQUATE
                                    00005   279 R5       EQU   5    REGISTER  5 EQUATE
                                    00006   280 R6       EQU   6    REGISTER  6 EQUATE
                                    00007   281 R7       EQU   7    REGISTER  7 EQUATE
                                    00008   282 R8       EQU   8    REGISTER  8 EQUATE
                                    00009   283 R9       EQU   9    REGISTER  9 EQUATE
                                    0000A   284 R10      EQU   10   REGISTER 10 EQUATE
                                    0000B   285 R11      EQU   11   REGISTER 11 EQUATE
                                    0000C   286 R12      EQU   12   REGISTER 12 EQUATE
                                    0000D   287 R13      EQU   13   REGISTER 13 EQUATE
                                    0000E   288 R14      EQU   14   REGISTER 14 EQUATE
                                    0000F   289 R15      EQU   15   REGISTER 15 EQUATE
                                            291          SAVE  (14,12),,EKYEFL1A-&SYSDATE SAVE CALLER'S REGISTERS
                                            298 *
00001A 18CF                                 299          LR    R12,R15             TRANSFER ENTRY REGISTER
                                    00000   300          USING EKYEFL1A,R12        ESTABLISH ADDRESSABILITY
00001C 58B1 0000                    00000   301          L     R11,0(R1)           GET A(INTERFACE CONTROL BLOCK)
                                    00000   302          USING DVRXCUDT,R11        SET INTERFACE CONTROL BLOCK BASE
000020 58A1 000C                    0000C   303          L     R10,12(R1)          GET ADDRESS 64 BYTE ANCHOR AREA
                                    00000   304          USING USERAREA,R10        SET BASE FOR USER AREA
000024 1851                                 305          LR    R5,R1               SAVE ADDRESS OF INPUT PARMS
                                            306 *
000026 92E7 B104      00104                 307          MVI   UDTENTRD,C'X'       SET EXIT-ENTERED-AT-LEASE-ONCE
00002A 92E7 B105      00105                 308          MVI   UDTINCTL,C'X'       SET EXIT-IN-CONTROL FLAG (DPR RESETS
                                            309 *-------------------------------------------------------------------*
                                            310 *        ON FIRST INVOCATION,                                       *
                                            311 *        ISSUE GET MAIN FOR AN EXIT SAVE AREA - NEEDED TO ISSUE CALLS *
                                            312 *        (NOT REQUIRED BY THIS EXIT - SHOWN AS AN EXAMPLE)          *
                                            313 *-------------------------------------------------------------------*
                                            314 *
00002E 5810 A000                    00000   315          L     R1,USERWD1          GET POINTER TO EXIT SAVE AREA
000032 1211                                 316          LTR   R1,R1               HAS ONE BEEN OBTAINED
000034 4770 C046                    00046   317          BNZ   GOTSTOR             YES-THIS IS NOT FIRST CALL
                                            318          GETMAIN R,LV=72           GET STORAGE FOR A SAVE AREA

000042 5010 A000                    00000   323          ST    R1,USERWD1          SAVE ADDRESS OF SAVE IN ANCHOR AREA
000046                                      324 GOTSTOR  DS    0H   STORAGE IS AVAILABLE FOR A SAVE AREA
                                            325 *
```

Figure 29 (Part 5 of 10). Sample Field Exit Routine (Assembler)

```
                                  326 *---------------------------------------------------------------------*
                                  327 *       CHAIN IN SAVE AREA (ADDRESS IN R1)                            *
                                  328 *---------------------------------------------------------------------*
                                  329 *
000046 501D 0008           00008  330          ST    R1,8(R13)          SET EXIT SAVE ADDR IN CALLERS SAVE
00004A 50D1 0004           00004  331          ST    R13,4(R1)          SET A(CALLERS SAVE) IN EXIT AREA
00004E 18D1                       332          LR    R13,R1             SET NEW SAVE AS CURRENT FOR CALLS
                                  333 *
                                  334 *---------------------------------------------------------------------*
                                  335 *       BRANCH DEPENDING ON THE TYPE OF CALL / FUNCTION CODE          *
                                  336 *---------------------------------------------------------------------*
                                  337 *
000050 D501 B020 C23A 00020 0023A 338          CLC   UDTCALL,XDEMSTAR   A DPROP/DEM SOURCE-TO-TARGET CALL?
000056 4780 C072           00072  339          BE    SRCTARG            YES-PROCESS THAT CALL
                                  340 *
00005A D501 B020 C23E 00020 0023E 341          CLC   UDTCALL,XDEMTSRC   A DPROP TARGET-TO-SOURCE CALL?
000060 4780 C100           00100  342          BE    TARGSRC            YES-PROCESS THAT CALL
                                  343 *
000064 D501 B020 C23C 00020 0023C 344          CLC   UDTCALL,XUIMDEFN   IS THIS A UIM DEFINITION CALL?
00006A 4780 C176           00176  345          BE    UIMVAL00           YES-PERFORM VALIDATION- IF ANY
                                  346 *
00006E 47F0 C22A           0022A  347          B     DEMBAD00           NO-UNPREDICTABLE ENVIRONMENT- ABORT
                                  348 *
                                  349          DROP  R10                RELEASE USER ANCHOR AREA BASE
                                  351 ***********************************************************************
                                  352 *                                                                     *
                                  353 *       PROCESSING DPROP OR DXT-DEM 'SOURCE-TO-TARGET' CALL.          *
                                  354 *                                                                     *
                                  355 ***********************************************************************

                                  357 *---------------------------------------------------------------------*
                                  358 *       SET UP TARGET ADDRESSES FOR LOOP                              *
                                  359 *             R8  = 1                                                 *
                                  360 *             R9  = A(LAST TARGET BYTE)                               *
                                  361 *             R10 = A(TARGET (OUTPUT) BUFFER)                         *
                                  362 *---------------------------------------------------------------------*
                                  363 *
                                  364 *       RECHECK TARGET LENGTH - SHOULD ALWAYS BE GOOD, BUT.....
000072                            365 SRCTARG  DS    0H
000072 4890 B064           00064  366          LH    R9,UDTTBYTV        GET NUMBER OF TARGET BYTES
000076 4990 C248           00248  367          CH    R9,ZEROLGT         IS TARGET LENGTH NON ZERO
00007A 4780 C1E4           001E4  368          BE    ERRTLGT1           NO-PERFORM ERROR FUNCTION
00007E 4990 C246           00246  369          CH    R9,MAXTARLG        IS MAXIMUM TARGET LENGTH EXCEEDED
000082 4720 C1F2           001F2  370          BH    ERRTLGT2
                                  371 *
000086 58A5 0008           00008  372          L     R10,8(R5)          GET ADDRESS OF TARGET (OUTPUT) BUFFR
00008A 1A9A                       373          AR    R9,R10             GET ADDRESS OF TARGET END +1
00008C 0690                       374          BCTR  R9,0               BACK UP TO TARGET END
00008E 4180 0001           00001  375          LA    R8,1               GET A ONE FOR INCREMENT

                                  377 *---------------------------------------------------------------------*
                                  378 *       SET UP SOURCE ADDRESSES FOR LOOP                              *
                                  379 *             R4  = A(SOURCE (INPUT) BUFFER)                          *
                                  380 *             R6  = 1                                                 *
                                  381 *             R7  = A(LAST SOURCE BYTE)                               *
                                  382 *---------------------------------------------------------------------*
                                  383 *
000092 5845 0004           00004  384          L     R4,4(R5)           GET ADDRESS OF SOURCE (INPUT) BUFFER
                                  385 *
                                  386 *       RECHECK SOURCE LENGTH - SHOULD ALWAYS BE GOOD, BUT.....
000096 4870 B058           00058  387          LH    R7,UDTSBYTV        GET NUMBER OF SOURCE BYTES
00009A 4970 C248           00248  388          CH    R7,ZEROLGT         IS SOURCE LENGTH SPECIFIED
00009E 4780 C1C8           001C8  389          BE    ERRSLGT1           NO-PERFORM ERROR FUNCTION
0000A2 4970 C244           00244  390          CH    R7,MAXSRCLG        IS MAXIMUM SOURCE LENGTH EXCEEDED
0000A6 4720 C1D6           001D6  391          BH    ERRSLGT2
                                  392 *
```

Figure 29 (Part 6 of 10). Sample Field Exit Routine (Assembler)

```
0000AA 1A74                          393          AR    R7,R4              GET ADDRESS OF SOURCE END +1
0000AC 0670                          394          BCTR  R7,0                BACK UP TO SOURCE END
0000AE 4160 0001             00001   395          LA    R6,1                GET A ONE FOR INCREMENT
                                     396 *
0000B2                               397 NEXTBYTE DS    0H   PROCESS NEXT  BYTE (8 BITS) OF SOURCE
0000B2 4334 0000             00000   398          IC    R3,0(R4)           GET BYTE SOURCE BYTE
0000B6 8930 0018             00018   399          SLL   R3,24              PUT SOURCE BYTE IN TOP OF REG
0000BA 4150 0008             00008   400          LA    R5,8               SET NUMBER BITS PER BYTE
                                     401 *
0000BE                               402 NEXTBIT0 DS    0H   PROCESS NEXT  BIT IN SOURCE BYTE (TOP OF R5)
0000BE 1B22                          403          SR    R2,R2              CLEAR WORK REGISTER
0000C0 92F0 A000       00000         404          MVI   0(R10),XDBITOFF    ASSUME THE BIT IS OFF
0000C4 8D20 0001             00001   405          SLDL  R2,1               PUT SOURCE BIT IN WORK REG
0000C8 5420 C240             00240   406          N     R2,LOWBITMK        CLEAR ALL BITS BUT ONE JUST SHIFTED
0000CC 4780 C0D4             000D4   407          BZ    BITSET00           IF IT WAS OFF TARGET VALUE IS SET
0000D0 92F1 A000       00000         408          MVI   0(R10),XDBITON     RESET THE TARGET FOR BIT ON
0000D4                               409 BITSET00 DS    0H   THE TARGET BYTE VALUE HAS BEEN SET
0000D4 86A8 C0E8             000E8   410          BXH   R10,R8,CONVCOMP    POINT TO NEXT TARGET BYTE
0000D8 4650 C0BE             000BE   411          BCT   R5,NEXTBIT0        REPEAT FOR NEXT BIT
0000DC 8746 C0B2             000B2   412          BXLE  R4,R6,NEXTBYTE     POINT TO NEXT SOURCE BYTE AND CONVER
                                     413 *
0000E0                               414 PADSOURC DS    0H   TARGET LONGER THAN SOURCE PAD WITH C'0'
0000E0 92F0 A000       00000         415          MVI   0(R10),XDBITOFF    PAD WITH BIT OFF VALUE
0000E4 87A8 C0E0             000E0   416          BXLE  R10,R8,PADSOURC    POINT TO NEXT TARGET BYTE AND FILL
                                     417 *
0000E8                               418 CONVCOMP DS    0H   CONVERSION COMPLETE EXIT
0000E8 1BFF                          420          SR    R15,R15            RETURN CODE IS ALWAYS ZERO


                                     422 ***********************************************************************
0000EA                               423 EXITRETN DS    0H   RETURN TO CALLER - DXT-UIM, DXT-DEM, OR DPROP
                                     424 ***********************************************************************
0000EA 50F0 B108             00108   425          ST    R15,UDTXRETC       STORE RETURN CODE IN UDT
                                     426 *
0000EE 58DD 0004             00004   427          L     R13,4(R13)         RESTORE CALLER'S SAVE AREA
                                     428          RETURN (14,12),T,RC=(15)  RESTORE CALLER'S REGISTERS
                                     434 ***********************************************************************
                                     435 *                                                                     *
                                     436 *        PROCESSING DPROP 'TARGET-TO-SOURCE' CALL.                    *
                                     437 *                                                                     *
                                     438 ***********************************************************************

                                     440 *---------------------------------------------------------------------*
                                     441 *        SET UP TARGET ADDRESSES FOR LOOP                             *
                                     442 *              R9  = A(TARGET (INPUT) BUFFER)                         *
                                     443 *              R10 = A(LAST TARGET BYTE)                              *
                                     444 *---------------------------------------------------------------------*
                                     445 *
                                     446 *        RECHECK TARGET LENGTH - SHOULD ALWAYS BE GOOD, BUT.....
000100                               447 TARGSRC  DS    0H
000100 48A0 B064             00064   448          LH    R10,UDTTBYTV       GET NUMBER OF TARGET BYTES
000104 49A0 C248             00248   449          CH    R10,ZEROLGT        IS TARGET LENGTH NON ZERO
000108 4780 C1E4             001E4   450          BE    ERRTLGT1           NO-PERFORM ERROR FUNCTION
00010C 49A0 C246             00246   451          CH    R10,MAXTARLG       IS MAXIMUM TARGET LENGTH EXCEEDED
000110 4720 C1F2             001F2   452          BH    ERRTLGT2           YES-PERFORM ERROR FUNCTION
                                     453 *
000114 5895 0008             00008   454          L     R9,8(R5)           GET ADDRESS OF TARGET (INPUT) BUFFER
000118 1AA9                          455          AR    R10,R9             GET ADDRESS OF TARGET END +1
00011A 06A0                          456          BCTR  R10,0              BACK UP TO TARGET END

                                     458 *---------------------------------------------------------------------*
                                     459 *        SET UP SOURCE ADDRESSES FOR LOOP                             *
                                     460 *              R7  = A(SOURCE (OUTPUT) BUFFER)                        *
                                     461 *              R8  = A(LAST SOURCE BYTE)                              *
                                     462 *---------------------------------------------------------------------*
```

Figure 29 (Part 7 of 10). Sample Field Exit Routine (Assembler)

```
                                    463 *
                                    464 *        RECHECK SOURCE LENGTH - SHOULD ALWAYS BE GOOD, BUT.....
00011C 4880 B058              00058 465          LH    R8,UDTSBYTV         GET NUMBER OF SOURCE BYTES
000120 4980 C248              00248 466          CH    R8,ZEROLGT          IS SOURCE LENGTH SPECIFIED
000124 4780 C1C8              001C8 467          BE    ERRSLGT1            NO-PERFORM ERROR FUNCTION
000128 4980 C244              00244 468          CH    R8,MAXSRCLG         IS MAXIMUM SOURCE LENGTH EXCEEDED
00012C 4720 C1F2              001F2 469          BH    ERRTLGT2            YES-PERFORM ERROR FUNCTION
                                    470 *
000130 5875 0004              00004 471          L     R7,4(R5)            GET ADDRESS OF SOURCE (OUTPUT) BUFFR
000134 1A87                         472          AR    R8,R7               GET ADDRESS OF SOURCE END +1
000136 0680                         473          BCTR  R8,0                BACK UP TO SOURCE END

                                    475 *-------------------------------------------------------------------*
                                    476 *        PROCESS NEXT BYTE (8 BITS) OF SOURCE                       *
                                    477 *-------------------------------------------------------------------*
000138                              478 TARGNEXT DS    0H
000138 1722                         479          XR    R2,R2               PRESET ALL BITS TO ZERO
00013A 5810 C450              00450 480          L     R1,=X'00000080'     SETUP 'OR' REGISTER FOR HIGH BIT
                                    481 *
00013E                              482 TARGNX10 DS    0H
00013E 199A                         483          CR    R9,R10              ALL BYTES OF TARGET PROCESSED?
000140 4720 C15A              0015A 484          BH    TARGNX30            YES-SKIP CHECK OF TARGET BYTE
000144 95F0 9000        00000       485          CLI   0(R9),XDBITOFF      IS THIS BYTE OFF/ZERO/FALSE?
000148 4780 C156              00156 486          BE    TARGNX20            YES-VALUE IS OK
00014C 95F1 9000        00000       487          CLI   0(R9),XDBITON       IS THIS BYTE ON/ONE/TRUE?
000150 4770 C21C              0021C 488          BNE   ERRCONV1            NO-PERFORM CONVERSION ERROR FUNCTION
000154 1621                         489          OR    R2,R1               INDICATE THE ONE IN SOURCE BYTE
000156                              490 TARGNX20 DS    0H
000156 4190 9001              00001 491          LA    R9,1(0,R9)          POINT NEXT BYTE IN TARGET BUFFER
00015A                              492 TARGNX30 DS    0H
00015A 8A10 0001              00001 493          SRA   R1,1                SHIFT 'OR' REG TO NEXT BIT POSITION
00015E 4770 C13E              0013E 494          BNZ   TARGNX10            AND PROCESS IF NOT ZERO
                                    495 *
000162 4220 7000              00000 496          STC   R2,0(0,R7)          STORE THE SOURCE BYTE
000166 4170 7001              00001 497          LA    R7,1(0,R7)          POINT NEXT SOURCE BYTE IN BUFFER
00016A 1978                         498          CR    R7,R8               MORE SOURCE BYTES TO PROCESS?
00016C 47D0 C138              00138 499          BNH   TARGNEXT            YES-BRANCH TO PROCESS NEXT BYTE
                                    500 *
000170 1BFF                         501          SR    R15,R15             SET SUCCESSFUL RETURN CODE
000172 47F0 C0EA              000EA 502          B     EXITRETN            GO TO COMMON RETURN POINT
                                    504 *********************************************************************
                                    505 *        PROCESSING A DXT-UIM 'DEFINITION CALL'.                    *
                                    506 *                                                                   *
                                    507 *        LETS PERFORM UIM VALIDATION                                *
                                    508 *********************************************************************
                                    509 *
000176                              510 UIMVAL00 DS    0H   PERFORM UIM VALIDATION
000176 4870 B058              00058 511          LH    R7,UDTSBYTV         GET NUMBER OF MAX SOURCE BYTES
00017A 4970 C248              00248 512          CH    R7,ZEROLGT          IS SOURCE LENGTH SPECIFIED
00017E 4780 C1C8              001C8 513          BE    ERRSLGT1            NO-PERFORM ERROR FUNCTION
000182 4970 C244              00244 514          CH    R7,MAXSRCLG         IS MAXIMUM SOURCE LENGTH EXCEEDED
000186 4720 C1D6              001D6 515          BH    ERRSLGT2
                                    516 *        SOURCE IS VALIDATED - PROCESS TARGET
00018A 4860 B064              00064 517          LH    R6,UDTTBYTV         GET NUMBER OF MAX TARGET BYTES
00018E 95E5 B062        00062       518          CLI   UDTTBYTI,C'V'       IS TARGET LENGTH VARIES TYPE
000192 4770 C1A0              001A0 519          BNE   UIMVAL60            NOT JUST DO VALIDATION
000196 1867                         520          LR    R6,R7               DUPLICATE SOURCE BYTES
000198 8960 0003              00003 521          SLL   R6,3                MULTIPLE BY 8 FOR NUMBER BITS
00019C 4060 B064              00064 522          STH   R6,UDTTBYTV         SET NUMBER OF MAX TARGET BYTES
0001A0                              523 UIMVAL60 DS    0H   VALIDATE TARGET LENGTH
0001A0 4960 C248              00248 524          CH    R6,ZEROLGT          IS TARGET LENGTH NON ZERO
0001A4 4780 C1E4              001E4 525          BE    ERRTLGT1            NO-PERFORM ERROR FUNCTION
0001A8 4960 C246              00246 526          CH    R6,MAXTARLG         IS MAXIMUM TARGET LENGTH EXCEEDED
0001AC 4720 C1F2              001F2 527          BH    ERRTLGT2
                                    528 *
```

Figure 29 (Part 8 of 10). Sample Field Exit Routine (Assembler)

```
                                       529 *       VALIDATE ADDITION CONTROL VALUES
0001B0 D501 B060 C238 00060 00238      530         CLC   UDTTTYPE,XDTYPEC    IS TARGET DATA TYPE CHARACTER
0001B6 4770 C200             00200     531         BNE   UIMBAD40            NO-PERFORM ERROR FUNCTION
0001BA 95D5 B066      00066            532         CLI   UDTTSCLI,C'N'       IS TARGET SCALE SET FOR VARIES
0001BE 4770 C20E             0020E     533         BNE   UIMBAD50            NO-PERFORM ERROR FUNCTION
                                       534 *
0001C2 1BFF                            535         SR    R15,R15             SET SUCCESSFUL RETURN CODE
0001C4 47F0 C0EA             000EA     536         B     EXITRETN            GO TO COMMON RETURN POINT
                                       538 ***********************************************************************
                                       539 *       ERROR LOGIC.                                                  *
                                       540 ***********************************************************************

0001C8                                 542 ERRSLGT1 DS   0H   SOURCE LENGTH IS OMITTED (ZERO)
0001C8 D23F B10C C24A 0010C 0024A      543         MVC   UDTXMESG,EMSG0000   SET SOURCE LENGTH ZERO MESSAGE
0001CE 41F0 0004             00004     544         LA    R15,4               SET ERROR DETECTED INDICATION
0001D2 47F0 C0EA             000EA     545         B     EXITRETN            GO TO COMMON RETURN POINT
0001D6                                 546 ERRSLGT2 DS   0H   SOURCE LENGTH IS EXCEEDED
0001D6 D23F B10C C28A 0010C 0028A      547         MVC   UDTXMESG,EMSG0010   SET SOURCE LENGTH EXCEEDED
0001DC 41F0 0004             00004     548         LA    R15,4               SET ERROR DETECTED INDICATION
0001E0 47F0 C0EA             000EA     549         B     EXITRETN            GO TO COMMON RETURN POINT
0001E4                                 550 ERRTLGT1 DS   0H   TARGET LENGTH IS ZERO
0001E4 D23F B10C C2CA 0010C 002CA      551         MVC   UDTXMESG,EMSG0020   SET TARGET LENGTH IS ZERO MESSAGE
0001EA 41F0 0004             00004     552         LA    R15,4               SET ERROR DETECTED INDICATION
0001EE 47F0 C0EA             000EA     553         B     EXITRETN            GO TO COMMON RETURN POINT
0001F2                                 554 ERRTLGT2 DS   0H   TARGET LENGTH IS EXCEEDED
0001F2 D23F B10C C30A 0010C 0030A      555         MVC   UDTXMESG,EMSG0030   SET LENGTH EXCEEDED MESSAGE
0001F8 41F0 0004             00004     556         LA    R15,4               SET ERROR DETECTED INDICATION
0001FC 47F0 C0EA             000EA     557         B     EXITRETN            GO TO COMMON RETURN POINT
000200                                 558 UIMBAD40 DS   0H   TARGET DATA TYPE IS NOT CHARACTER
000200 D23F B10C C34A 0010C 0034A      559         MVC   UDTXMESG,EMSG0040   SET INVALID DATA TYPE MESSAGE
000206 41F0 0004             00004     560         LA    R15,4               SET ERROR DETECTED INDICATION
00020A 47F0 C0EA             000EA     561         B     EXITRETN            GO TO COMMON RETURN POINT
00020E                                 562 UIMBAD50 DS   0H   TARGET SCALE IS INVALID ('N' NOT SPECIFIED)
00020E D23F B10C C38A 0010C 0038A      563         MVC   UDTXMESG,EMSG0050   SET INVALID TARGET SCALE MESSAGE
000214 41F0 0004             00004     564         LA    R15,4               SET ERROR DETECTED INDICATION
000218 47F0 C0EA             000EA     565         B     EXITRETN            GO TO COMMON RETURN POINT
00021C                                 566 ERRCONV1 DS   0H   TARGET FIELD CONTAINS BYTE OTHER THAN '0' OR '1'
00021C D23F B10C C3CA 0010C 003CA      567         MVC   UDTXMESG,EMSG0060   SET TARGET FIELD VALUE IS INVALID
000222 41F0 0004             00004     568         LA    R15,4               SET ERROR DETECTED INDICATION
000226 47F0 C0EA             000EA     569         B     EXITRETN            GO TO COMMON RETURN POINT
                                       570 *
00022A                                 571 DEMBAD00 DS   0H   INVALID USER DATA TYPE CALL FUNCTION
00022A D23F B10C C40A 0010C 0040A      572         MVC   UDTXMESG,EMSGD000 SET NOT SUPPORTED MESSAGE
000230 41F0 0010             00010     573         LA    R15,16              SET ERROR DETECTED INDICATION
000234 47F0 C0EA             000EA     574         B     EXITRETN            GO TO COMMON RETURN POINT
                                       576 ***********************************************************************
                                       577 *       DATA DEFINITIONS                                              *
                                       578 ***********************************************************************

000238 C340                            580 XDTYPEC  DC   CL2'C '             IS TARGET DATA TYPE CHARACTER
00023A E2E3                            581 XDEMSTAR DC   CL2'ST'             CODE FOR DEM/DPROP SOURCE TO TARGET
00023C C4C6                            582 XUIMDEFN DC   CL2'DF'             CODE FOR UIM VALIDATION CALL
00023E E3E2                            583 XDEMTSRC DC   CL2'TS'             CODE FOR DPROP TARGET TO SOURCE
                                       584 *
000240                                 585 LOWBITMK DS   0F                  ALIGN MASK ON FULL WORD
000240 00000001                        586          DC   XL4'00000001'       MASK CLEAR ALL BUT FIRST BIT
                                       587 *
000244 0010                            588 MAXSRCLG DC   AL2(016) MAXIMUM SOURCE LENGTH
000246 0080                            589 MAXTARLG DC   AL2(128) MAXIMUM TARGET LENGTH
000248 0000                            590 ZEROLGT  DC   AL2(0)   NO LENGTH VALUE
                                       591 *
```

Figure 29 (Part 9 of 10). Sample Field Exit Routine (Assembler)

```
                                  592 *
00024A                            593 EMSG0000 DS    0CL64
00024A C5E7C9E37EC5D2E8           594          DC    CL16'EXIT=EKYEFL1A - '
00025A E2D6E4D9C3C540D3           595          DC    CL44'SOURCE LENGTH NOT SPECIFIED - REQUIRED.     '
000286 40404040                   596          DC    CL4'    '
00028A                            597 EMSG0010 DS    0CL64
00028A C5E7C9E37EC5D2E8           598          DC    CL16'EXIT=EKYEFL1A - '
00029A E2D6E4D9C3C540D3           599          DC    CL44'SOURCE LENGTH EXCEEDS MAXIMUM ALLOWED.      '
0002C6 40404040                   600          DC    CL4'    '
0002CA                            601 EMSG0020 DS    0CL64
0002CA C5E7C9E37EC5D2E8           602          DC    CL16'EXIT=EKYEFL1A - '
0002DA E3C1D9C7C5E340D3           603          DC    CL44'TARGET LENGTH NOT SPECIFIED - REQUIRED.     '
000306 40404040                   604          DC    CL4'    '
00030A                            605 EMSG0030 DS    0CL64
00030A C5E7C9E37EC5D2E8           606          DC    CL16'EXIT=EKYEFL1A - '
00031A E3C1D9C7C5E340D3           607          DC    CL44'TARGET LENGTH EXCEEDS MAXIMUM ALLOWED.      '
000346 40404040                   608          DC    CL4'    '
00034A                            609 EMSG0040 DS    0CL64
00034A C5E7C9E37EC5D2E8           610          DC    CL16'EXIT=EKYEFL1A - '
00035A E3C1D9C7C5E340C4           611          DC    CL44'TARGET DATA TYPE MUST BE CHARACTER.         '
000386 40404040                   612          DC    CL4'    '
00038A                            613 EMSG0050 DS    0CL64
00038A C5E7C9E37EC5D2E8           614          DC    CL16'EXIT=EKYEFL1A - '
00039A E3C1D9C7C5E340E2           615          DC    CL44'TARGET SCALE MUST NOT BE SPECIFIED.         '
0003C6 40404040                   616          DC    CL4'    '
0003CA                            617 EMSG0060 DS    0CL64
0003CA C5E7C9E37EC5D2E8           618          DC    CL16'EXIT=EKYEFL1A - '
0003DA E5C1D3E4C540C9D5           619          DC    CL44'VALUE IN TARGET FIELD OTHER THAN ''0'' OR ''1'' '
000406 40404040                   620          DC    CL4'    '
00040A                            621 EMSGD000 DS    0CL64
00040A C5E7C9E37EC5D2E8           622          DC    CL16'EXIT=EKYEFL1A - '
00041A C4C1E3C140E3E8D7           623          DC    CL44'DATA TYPE CALL FUNCTION CANNOT BE IDENTIFIED'
000446 4B404040                   624          DC    CL4'.   '
                                  625 *
000000                            626 PRMLIST  DSECT MAPPING OF PARMS PASSED ON ENTRY
000000                            627 PRMUTDB@ DS    A    ADDRESS OF USER DATA TYPE INTERFACE BLK(UDT)
000004                            628 PRMSRCB@ DS    A    ADDRESS OF SOURCE BUFFER AREA            *
000008                            629 PRMTARG@ DS    A    ADDRESS OF TARGET BUFFER AREA
00000C                            630 PRMANCH@ DS    A    ADDRESS OF USER ANCHOR AREA FOR THIS EXIT
                                  632 ***********************************************************************
                                  633 *        THE FOLLOWING EKYRCUDT MACRO (WHICH MAPS THE UDT INTERFACE
                                  634 *         CONTROL BLOCK, IS SHIPPED WITH THE DPROP PRODUCT
                                  635 ***********************************************************************
                                  636          EKYRCUDT  ,              DEFINITION OF UDT CONTROL BLOCK


000000                            863 USERAREA DSECT  THIS 64 BYTE FIELD IS FOR USE BY THE EXIT
                                  864 *        THIS AREA IS FOR THE EXCLUSIVE USE OF THIS EXIT
                                  865 *        ITS CONTENTS WILL BE PRESERVED BETWEEN CALLS
                                  866 *        IT IS INITIALIZED TO BINARY ZEROS.
000000                            867 USERWD1  DS    F    USER ANCHOR WORD 1 (IN THIS CASE A SAVE POINTER)
000004                            868          DS    15F  REMAINDER OF USER ANCHORS AREA
                                  869 ***********************************************************************
                                  870          END   ,                  END OF DATA EXIT ROUTINE
000450 00000080                   871                =X'00000080'
```

Figure 29 (Part 10 of 10). Sample Field Exit Routine (Assembler)

Definitions for the First Sample Field Exit Routine

This section contains definitions associated with the sample Field exit routine. The following types of definitions are provided:

- IMS DBDGEN and PSBGEN definitions
- DB2 CREATE TABLE definitions
- DataRefresher definitions required to define the PR with DXT and to extract the IMS data with DataRefresher
- SQL statements required to define the PR in the MVG input tables without DataRefresher

DBDGEN Definitions

Figure 30 shows a DBDGEN definition for the Field exit routine in Figure 29 on page 128.

```
  DBD     NAME=DB1,ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),          C
               EXIT=(EKYRUP00)
  DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
  SEGM    NAME=SEG1,PARENT=0,BYTES=120
  FIELD   NAME=(FLD1,SEQ,U),START=3,BYTES=2
*
  DBDGEN
  FINISH
  END
```

Figure 30. DBDGEN Definition

Note: The EXIT= keyword of the DBD macro specifies that EKYRUP00 (the RUP) be called when a segment of this DBD is changed. This is required for synchronous HR propagation with DPROP.

PSBGEN Definitions

Figure 31 shows a PSBGEN definition for the Field exit routine in Figure 29 on page 128.

```
         PCB   TYPE=DB,DBDNAME=DB1,NAME=HUPPCB,                       C
               KEYLEN=120,PROCOPT=A
         SENSEG NAME=SEG1
*
         PSBGEN PSBNAME=PSBDPR2
         END
```

Figure 31. PSBGEN Definition

CREATE TABLE Statement

Figure 32 on page 139 shows a CREATE TABLE definition for the Field exit routine in Figure 29 on page 128.

```
CREATE TABLE TABLE01
            (COL1     CHAR(6)       NOT NULL,
             COLB     SMALLINT      NOT NULL WITH DEFAULT,
             COLH     SMALLINT      NOT NULL WITH DEFAULT,
             COLC     CHAR(32)      NOT NULL WITH DEFAULT,
          PRIMARY KEY (COL1))
          DATA CAPTURE CHANGES
       IN DU096606.PROPTS    ;

CREATE UNIQUE INDEX XN01 ON TABLE01 (COL1)
       USING VCAT KOE ;
```

Figure 32. CREATE TABLE Statement

Note: The DATA CAPTURE CHANGES clause specifies that the changed DB2 rows are captured and that the DB2CDCEX routine (the HUP) is called when a row of this table is changed. This is required for synchronous RH-propagation with DPROP.

Using DataRefresher to Define the PR

This section shows how to use DataRefresher to define the PR for the Field exit routine in Figure 29 on page 128.

CREATE DATATYPE

Figure 33 shows a CREATE DATATYPE definition for the Field exit routine in Figure 29 on page 128.

```
CREATE DATATYPE SRCTYPE=AA,  EXIT=EKYEFL1A,
       SRCBYTES=VARIES,
       TRGTYPE=C ,
       TRGBYTES=VARIES;
```

Figure 33. CREATE DATATYPE Definition

The CREATE DATATYPE command provides the following information:

- It creates a user data type called AA and associates the Field exit routine EKYEFL1A with this user data type.

  The Field exit routine, EKYEFL1A, is called to reformat each field defined in a DXTPSB with a TYPE=AA keyword.

- SRCBYTES=VARIES means that the length of the fields (in their user format) with a user data type AA can have different BYTES values coded in the FIELD statements of the CREATE DXTPSB control statement.

- TRGTYPE=C means that the Field exit routine reformats the fields between the user data type and a character data type.

- TRGBYTES=VARIES means that the length of the fields (in their DPROP-supported and DXT-supported character format) are established by the definition call generated by DataRefresher UIM when it processes a FIELD statement with this user data type.

CREATE DXTPSB

Figure 34 shows a CREATE DXTPSB definition for the Field exit routine in Figure 29 on page 128.

```
CREATE DXTPSB  NAME=KOEPSB2

     DXTPCB    NAME=PCB001,DBACCESS=HDAM,DBNAME=DB1

       SEGMENT NAME=SEG1, PARENT=0, BYTES=120

           FIELD    NAME=FLD1 , START=3,  BYTES=2 , SEQFLD=R
           FIELD    NAME=FLDB , START=5,  BYTES=1 , TYPE=B
           FIELD    NAME=FLDH , START=6,  BYTES=2 , TYPE=H
           FIELD    NAME=FLDC , START=9,  BYTES=4 , TYPE=AA ;
```

Figure 34. CREATE DXTPSB Definition

Notes:

1. The Field FLDC is defined as having a data type AA. When DataRefresher UIM processes this field statement, it calls the Field exit routine EKYEFL1A associated with the user data type AA for a definition call.

   DataRefresher UIM also calls EKYEFL1A during the extract; DPROP calls it during propagation to reformat the field FLDC between its user data type AA and its character format.

2. The length of the FLDC in its user format is defined on the BYTES= keyword as four bytes.

   The length of the field in its DPROP format is set by the Field exit routine during the definition call that DataRefresher UIM generates.

CREATE DXTVIEW

Figure 35 shows a CREATE DXTVIEW definition for the Field exit routine in Figure 29 on page 128.

```
CREATE DXTVIEW    NAME     = VIEW011,
                  DXTPSB   = KOEPSB2,
                  DXTPCB   = PCB001,
                  SEGMENT  = SEG1,
                  MINSEGM  = SEG1,
                  FIELDS   = *         ;
```

Figure 35. CREATE DXTVIEW Definition

DataRefresher UIM SUBMIT Command and EXTRACT Statement

Figure 36 on page 141 shows a DataRefresher UIM SUBMIT command and EXTRACT statement for the Field exit routine in Figure 29 on page 128.

```
SUBMIT   EXTID=PR001,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPDIR=TW,
                         MAPCASE=1,
                         ACTION=REPL,
                         ERROPT=BACKOUT,
                         PCBLABEL=HUPPCB'

EXTRACT
    INTO TABLE01 (COL1   NOT NULL,
                  COLB   NOT NULL WITH DEFAULT,
                  COLH   NOT NULL WITH DEFAULT,
                  COLC   NOT NULL WITH DEFAULT
                 )
         SELECT   FLD1,
                  FLDB,
                  FLDH,
                  FLDC
         FROM VIEW011 ;
```

Figure 36. DataRefresher UIM SUBMIT Command and EXTRACT Statement

Notes:

1. The MAPEXIT= keyword of the SUBMIT control statement specifies EKYMCE00. This results in DataRefresher UIM calling the DPROP Map Capture Exit EKYMCE00 during the processing of the SUBMIT or EXTRACT. This is needed to allow DPROP to create the PR.
2. The EXTRACT statement informs DataRefresher and DPROP which fields must be mapped to which columns. The EXTRACT statement indicates, for example, that the field FLDC must be mapped to column COLC.

Using DataRefresher for the Extract

This section covers INITDEM and USE DXTPSB Control Statements.

Figure 37 shows INITDEM and USE DXTPSB control statements for the Field exit routine shown in Figure 35 on page 140.

```
INITDEM NAME=DXTPROD;
USE DXTPSB=KOEPSB2;
```

Figure 37. Using DataRefresher for the Extract: INITDEM and USE DXTPSB Control Statements

Defining the PR in the MVG Input Tables

Figure 38 on page 143 describes the SQL statements required to define the PR in the MVG input tables.

The following rows are inserted into the MVG input tables:

- One row is inserted into the DPRIPR table (the PR table).

This row identifies the PRID. By inserting an **E** into the PRTYPE column and a **1** into the MAPCASE column, the SQL statement indicates that the PR belongs to mapping case 1 of an extended-function PR.

- One row for the entity segment type SEG1 is inserted into the DPRISEG table (the SEG table).

  Because SEG1 is the root segment, no rows are inserted into DPRISEG for physical ancestors.

- One row is inserted into the DPRITAB table (the TAB table).

  This row indicates that the target table is T096606.TABLE01.

- One row is inserted into the DPRIFLD table (the FLD table) for each propagated field.

  The DPRIFLD row for the field FLDC has the value EKYEFL1A in the FLDEXIT column. This indicates that the Field is processed by the Field exit routine EKYEFL1A. The value AA in the DATATYPE column is used to identify the user data type.

  For PR definitions entered into the MVG input tables, the Field exit routine is not called for a definition call. Therefore, you must provide in the DPRIFLD row a complete definition of the field in its user and DPROP format. Accordingly, the row describing FLDC contains, in the BYTES column, the length of the field in its user format and, in the FLDEBYTE column, the length of the field in its DPROP format.

```
DELETE FROM T096606.DPRIPR WHERE PRID = 'PR001'           ;


INSERT INTO T096606.DPRIPR
             ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR,
               ERROPT,   ACTION)
      VALUES ('PR001',  'T096606','E',    '1',      'TW',
              'BACKOUT','REPL')                              ;

INSERT INTO T096606.DPRISEG
             ( PRID,   DBNAME, SEGNAME, ROLE, PCBLABEL,
      VALUES ('PR001','DB1',  'SEG1',   'E',  'HUPPCB')     ;

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL,   TABNAME )
      VALUES ('PR001','T096606', 'TABLE01')                  ;

INSERT INTO T096606.DPRIFLD
             ( PRID,     DBNAME,   SEGNAME, FLDNAME,
               TABQUAL,  TABNAME,  COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'FLD1',
              'T096606','TABLE01','COL1',
              'C ',      3,        2)                        ;

INSERT INTO T096606.DPRIFLD
             ( PRID,     DBNAME,   SEGNAME, FLDNAME,
               TABQUAL,  TABNAME,  COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'FLDB',
              'T096606','TABLE01','COLB',
              'B ',      5,        1)                        ;

INSERT INTO T096606.DPRIFLD
             ( PRID,     DBNAME,   SEGNAME, FLDNAME,
               TABQUAL,  TABNAME,  COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR001',  'DB1',    'SEG1',  'FLDH',
              'T096606','TABLE01','COLH',
              'H ',      6,        2)                        ;

INSERT INTO T096606.DPRIFLD
             ( PRID,     DBNAME,   SEGNAME, FLDNAME,
               TABQUAL,  TABNAME,  COLNAME,
               DATATYPE, POSITION, BYTES,
               FLDEXIT,  FLDETYPE, FLDEBYTE)
      VALUES ('PR001',  'DB1',    'SEG1',  'FLDC',
              'T096606','TABLE01','COLC',
              'AA',      9,        4,
              'EKYEFL1A','C',      32)      ;

COMMIT;
```

Figure 38. Defining the PR in the MVG Input Tables

Second Sample Field Exit Routine

Figure 39 on page 144 contains an example of a field exit routine in COBOL. Its functions are the same as those for the exit routine in "First Sample Field Exit Routine" on page 127. For information about this routine, refer to "First Sample Field Exit Routine" on page 127.

The source code in Figure 39 on page 144 is provided in the DPROP Sample Source Library (EKYSRC) under the member name EKYEFL1C. The definitions for this routine are the same as those for EKYEFL1A, except that the exit name is different. Specifically, the **EXIT=EKYEFL1A** in Figure 33 on page 139, and **EKYEfl1a** in Figure 38, are changed to **EKYEFL1C**. The text that refers to EKYEFL1A is also true for EKYEFL1C. Refer to “Definitions for the First Sample Field Exit Routine” on page 138 for information about the definitions.

```
****************** START OF SPECIFICATIONS **********************
*                                                               *
* MODULE NAME: EKYEFL1C                                         *
* -----------                                                   *
*                                                               *
* DESCRIPTIVE NAME: SAMPLE FIELD EXIT ROUTINE                   *
* ----------------                                              *
*                                                               *
* FUNCTION: THE PURPOSE OF THIS PROGRAM IS TO PROVIDE A SAMPLE  *
* --------  STRUCTURE FOR A FIELD EXIT ROUTINE. THIS EXAMPLE    *
*           CONVERTS A BIT STRING INTO A CHARACTER STRING OR    *
*           VICE-VERSA, DEPENDING ON THE FUNCTION CALL, WITH    *
*           EACH BIT REPRESENTED BY A CHARACTER, TO BE SET TO   *
*           '1' OR '0' BASED ON THE VALUE OF THE RELATED BIT    *
*           OR VICE VERSA.                                      *
*           (ALTERNATE REPRESENTATION MIGHT BE 'T' FOR TRUE AND *
*           'F' FOR FALSE.) THIS FUNCTION COULD BE USEFUL FOR   *
*           CONVERTING BIT CONTROL FIELDS TO INDIVIDUAL FLAG    *
*           BYTES.                                              *
*                                                               *
*           IN INSTALLATIONS WHICH COMBINE USEAGE OF:           *
*             - DXT, FOR THE ORIGINAL EXTRACT OF THE DL/I DATA  *
*             - DPROP, FOR THE PROPAGATION OF THE DL/I DATA,    *
*             THE EXIT WILL BE CALLED BOTH BY DataRefresher AND DPROP.    *
*                                                               *
*           DataRefresher CALLS THE EXIT:                                 *
*             - DURING DXT-UIM PROCESSING, WITH A 'DEFINITION   *
*               CALL' IN ORDER TO VALIDATE FIELD DEFINITIONS.   *
*             - DURING DXT-DEM PROCESSING, IN ORDER TO MAP      *
*               DURING THE DL/I DATA EXTRACT BIT-STRINGS INTO   *
*               CHARACTER STRINGS.                              *
*           DPROP CALLS THE EXIT:                               *
*             - DURING DATA PROPAGATION, IN ORDER TO MAP        *
*               THE BIT-STRINGS INTO CHARACTER STRINGS          *
*             - DURING DPROP CCU (CONSISTENCY CHECK UTILITY),   *
*               IN ORDER TO MAP THE BIT-STRINGS INTO            *
*               CHARACTER STRINGS.                              *
*                                                               *
*                                                               *
*                                                               *
* PROCESSING FOR DEFINITION CALL (FUNCTION=DF),                 *
* ISSUED BY DXT-UIM:                                            *
* --------------------------------------------                  *
*                                                               *
*         - THE SOURCE LENGTH IS CHECKED AGAINST THE MAXIMUM    *
*           SOURCE LENGTH (16).                                 *
*         - IF THE TARGET LENGTH HAS BEEN DEFINED ON THE        *
*           DXT-UIM 'CREATE DATATYPE' STATEMENT AS 'VARIES',    *
*           THE EXIT SETS ITS VALUE TO 8 TIMES THE SOURCE       *
*           LENGTH.                                             *
*         - IF THE TARGET LENGTH HAS BEEN SPECIFIED ON THE      *
*           DXT-UIM 'CREATE DATATYPE' STATEMENT, IT IS CHECKED  *
*           AGAINST 8 TIMES THE MAXIMUM SOURCE LENGTH.          *
*         - TARGET DATA TYPE IS ENSURED  TO BE 'C' AND TARGET   *
*           SCALE ENSURED TO BE 'N'.                            *
*                                                               *
*         NOTE FOR INSTALLATIONS WHICH USE DPROP WITHOUT DXT:   *
*           IF DPROP IS USED WITHOUT DXT, THE EXIT WILL NEVER   *
*           BE INVOKED FOR A DEFINITION CALL (DEFINITION CALLS  *
*           ARE NOT NECESSARY, SINCE THE USER PROVIDES ALL      *
```

Figure 39 (Part 1 of 9). Second Sample Field Exit Routine (COBOL)

```
*          DEFINITIONS (I.E. SOURCE LENGTH, TARGET LENGTH)     *
*          IN THE DPROP 'MVG INPUT TABLES'.                    *
*                                                              *
*                                                              *
* PROCESSING FOR CONVERSION SOURCE TO TARGET (FUNCTION=ST),    *
* ISSUED BY DXT-DEM AND DPROP:                                 *
* ---------------------------------------------------------    *
*                                                              *
*          THE SOURCE FIELD IS CONVERTED A BIT AT A TIME INTO  *
*          '0' OR '1' CHARACTERS IN THE TARGET FIELD. FOR      *
*          EXAMPLE THE 2 BYTE CHARACTER STRING 'A1' IS HEX     *
*          'C1F1' OR '1100000111110001' IN BINARY. IT WOULD    *
*          BE CONVERTED INTO THE 16 BYTES CHARACTER STRING     *
*          '1100000111110001'. THE LENGTH OF THE TARGET FIELD  *
*          TERMINATES PROCESSING. IF THE TARGET LENGTH IS      *
*          GREATER THAN SOURCE LENGTH TIMES 8, THE REMAINING   *
*          RIGHT HAND BYTES ARE SET TO THE CHARACTER '0'.      *
*                                                              *
*                                                              *
* PROCESSING FOR CONVERSION TARGET TO SOURCE (FUNCTION=TS),    *
* ISSUED BY DPROP:                                             *
* ---------------------------------------------------------    *
*                                                              *
*          EACH TARGET BYTE IS CONVERTED TO THE CORRESPONDING  *
*          BIT IN THE SOURCE FIELD. THE LENGTH OF THE SOURCE   *
*          FIELD TERMINATES PROCESSING. IF THE SOURCE FIELD    *
*          LENGTH IS GREATER THAN THE TARGET FIELD / 8, THE    *
*          REMAINING RIGHT HAND BITS ARE SET TO 0.             *
*          EACH BYTE MUST BE "0" OR "1".                       *
*                                                              *
/*
*                                                              *
* SPECIFIC EXIT FUNCTIONS DEMONSTRATED BY THIS MODULE.         *
* ----------------------------------------------------         *
*                                                              *
*  1. PROCESSING THE INVOCATION PARM LIST.                     *
*  2. USING THE USER ANCHOR AREA.                              *
*  3. IDENTIFYING THE REQUESTED FUNCTION.                      *
*  4. UIM VALIDATION OF 'V' TYPE LENGTH FIELDS.                *
*  5. THE USE OF THE MESSAGE AREA.                             *
*                                                              *
* INPUT: (PASSED AS PARAMETERS).                               *
* -----                                                        *
*                                                              *
*  1. UDT - USER DATA TYPE INTERFACE CONTROL BLOCK.            *
*  2. SOURCE BUFFER - THE SOURCE USER DATA (N/A FOR DEFINE CALL)*
*  3. TARGET BUFFER - TARGET AFTER CONVERSION (N/A FOR DEFINE). *
*  4. USER ANCHOR AREA - A 64 BYTE AREA FOR USE BY THE EXIT.    *
*                                                              *
*                                                              *
****************************************************************
*                                                              *
*    RETURN CODE AND MESSAGES ARE SET IN UDT BLOCK)            *
*                                                              *
*    RETURN CODE = 0  PROCESSING SUCCESSFUL - NO MESSAGE SET.   *
*                                                              *
*    RETURN CODE = 4  DATA TYPE VALIDATION FAILED - MESSAGE SET.*
*     'SOURCE LENGTH NOT SPECIFIED - REQUIRED.     '           *
*     'SOURCE LENGTH EXCEEDS MAXIMUM ALLOWED.      '           *
*     'TARGET LENGTH NOT SPECIFIED - REQUIRED.     '           *
*     'TARGET LENGTH EXCEEDS MAXIMUM ALLOWED.      '           *
```

Figure 39 (Part 2 of 9). Second Sample Field Exit Routine (COBOL)

```
012100*     'TARGET DATA TYPE MUST BE CHARACTER          '          * 01220000
012200*     'TARGET SCALE MUST NOT BE SPECIFIED          '          * 01230000
012300*                                                             * 01240000
012400*                                                             * 01250000
012500*    RETURN CODE =16  UNIDENTIFIED FUNCTION - MESSAGE IS SET. * 01260000
012600*     'DATA TYPE CALL FUNCTION CANNOT BE IDENTIFIED'          * 01270000
012700*                                                             * 01280000
012800***************************************************************** 01290000
012900/*                                                              01300000
013000***************************************************************** 01310000
013100*  INFORMATION FOR INSTALLATIONS WHICH COMBINE                * 01320000
013200*  THE USAGE OF DataRefresher AND DPROP.                                * 01330000
013300*  -------------------------------------------                * 01340000
013400*                                                             * 01350000
013500*  THESE INSTALLATIONS DEFINE THE DL/I-TO-DB2 AND VICE-VERSA  * 01360000
013600*  MAPPING BY PROVIDING MAPPING DEFINITIONS TO DXT.           * 01370000
013700*  USAGE OF THIS FIELD EXIT ROUTINE REQUIRES FOLLOWING        * 01380000
013800*  SPECIFICATIONS IN THE DataRefresher 'CREATE DATATYPE' AND            * 01390000
013900*  'CREATE DXTPSB' 'FIELD' STATEMENT:                         * 01400000
014000*                                                             * 01410000
014100*  INVOCATION OF A FIELD EXIT ROUTINE IS DEFINED BOTH         * 01420000
014200*  BY SPECIFICATIONS IN THE DXT 'CREATE DATATYPE' AND         * 01430000
014300*  'CREATE DXTPSB' 'FIELD' STATEMENT.                         * 01440000
014400*                                                             * 01450000
014500*  THE CREATE DATATYPE:                                       * 01460000
014600*  -------------------                                        * 01470000
014700*                                                             * 01480000
014800*   EXIT     = EKYEFL1C - THE EXIT LOAD MODULE NAME           * 01490000
014900*   SRCTYPE  = XX       - THE TWO CHARACTER USER DATA TYPE ID.  * 01500000
015000*   SRCBYTES = VARIES   - THE SOURCE FIELD LENGTH.            * 01510000
015100*            OR NNNN      (MAXIMUM SOURCE LENGTH IS 16 FOR    * 01520000
015200*                         THIS SAMPLE. THE EXIT PROGRAM COULD * 01530000
015300*                         HAVE THE LIMIT INCREASED TO 4092.)  * 01540000
015400*   TRGTYPE  = C        - MUST BE A 'C' FOR CHAR TYPE TARGET. * 01550000
015500*   TRGBYTES = VARIES   - THE TARGET FIELD/COLUMN LENGTH      * 01560000
015600*            OR NNNN      THE TARGET LENGTH SHOULD BE 8 TIMES * 01570000
015700*                         THE SOURCE LENGTH.                  * 01580000
015800*                         IF TRGBYTES IS SPECIFIED AS 'VARIES' * 01590000
015900*                         ON THE 'CREATE DATATYPE', THEN THE  * 01600000
016000*                         EXIT WILL SET (DURING THE 'DEFINITION * 01610000
016100*                         CALL') THE TARGET LENGTH TO 8 TIMES * 01620000
016200*                         THE SOURCE LENGTH.                  * 01630000
016300*                         (MAXIMUM TARGET LENGTH IS 128 IN    * 01640000
016400*                         THIS SAMPLE, BUT THE PROGRAM COULD  * 01650000
016500*                         HAVE THE LIMIT INCREASED TO 32,736.) * 01660000
016600*   SRCSCALE =, AND TRGSCALE = MUST NOT BE SPECIFIED.         * 01670000
016700*                                                             * 01680000
016800*  THE FIELD STATEMENT IN CREATE DXTFILE:                     * 01690000
016900*  -------------------------------------                      * 01700000
017000*                                                             * 01710000
017100*   TYPE     = XX       - RELATES THIS FIELD TO A DXT DATATYPE. * 01720000
017200*   BYTES    = NN       - THE SOURCE FIELD LENGTH.            * 01730000
017300*                         IF DEFINED AS 'VARIES' IN THE       * 01740000
017400*                         DATATYPE STATEMENT, IT MUST NOT     * 01750000
017500*                         EXCEED THE MAXIMUM FIELD LENGTH     * 01760000
017600*                         ALLOWED BY THE EXIT.                * 01770000
017700*                         IF NOT DEFINED AS 'VARIES',         * 01780000
017800*                         IT MUST EQUAL THE 'SRCBYTES'        * 01790000
017900*                         OPERAND IN THE DATATYPE STATEMENT.  * 01800000
018000*   SCALE    =   MUST NOT BE SPECIFIED.                       * 01810000
```

Figure 39 (Part 3 of 9). Second Sample Field Exit Routine (COBOL)

```
*                                                                *
/*
******************************************************************
*  INFORMATION FOR INSTALLATIONS WHICH USE DPROP WITHOUT DXT.   *
*  ---------------------------------------------------------     *
*                                                                *
*  THESE INSTALLATIONS DEFINE THE DL/I-TO-DB2 AND VICE-VERSA    *
*  MAPPING BY PROVIDING MAPPING DEFINITIONS IN THE DPROP        *
*  'MVG INPUT TABLES'.                                          *
*  USAGE OF THIS SAMPLE FIELD EXIT ROUTINE REQUIRES FOLLOWING   *
*  DEFINITIONS IN THE DPRIFLD TABLE:                            *
*                                                                *
*  INVOCATION OF A FIELD EXIT ROUTINE IS DEFINED BOTH           *
*  BY SPECIFICATIONS IN THAT ROW OF THE 'DPRIFLD' TABLE         *
*  WHICH DESCRIBES THE FIELD TO BE MAPPED.                      *
*                                                                *
*  COLUMNS OF THE DPRIFLD ROW SHOULD PROVIDE FOLLOWING          *
*  DEFINITIONS:                                                 *
*                                                                *
*  COLUMN OF   COLUMN                                           *
*  DPRIFLD     VALUE     EXPLANATIONS                           *
*  ------------------------------------------------------------- *
*  FLDEXIT   = EKYEFL1C: THE EXIT LOAD MODULE NAME              *
*  DATATYPE  = XX      : A TWO CHARACTER DATA-TYPE ID.          *
*  BYTES     = NNNN    : THE SOURCE FIELD LENGTH                *
*  FLDETYPE  = C       : THE TARGET DATA-TYPE MUST BE 'C '.     *
*  FLDEBYTE  = MMMMM   : THE TARGET FIELD LENGTH                *
*                        (MUST BE 8 TIMES THE SOURCE            *
*                         FIELD LENGTH).                        *
*  SCALE     =         : SHOULD EITHER NOT BE PROVIDED OR       *
*                        SHOULD BE SPECIFIED AS ZERO.           *
*  FLDESCAL  =         : SHOULD EITHER NOT BE PROVIDED OR       *
*                        SHOULD BE SPECIFIED AS ZERO.           *
*                                                                *
********************* END OF SPECIFICATIONS **********************
/*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. EKYEFL1C.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77  XDBITON       PICTURE X      USAGE DISPLAY       VALUE "1".
 77  XDBITOFF      PICTURE X      USAGE DISPLAY       VALUE "0".
 77  MAXSRCLG      PICTURE S9999  USAGE COMPUTATIONAL VALUE 016.
 77  MAXTARLG      PICTURE S9999  USAGE COMPUTATIONAL VALUE 0128.
 77  XTVALUE       PICTURE S9999  USAGE COMPUTATIONAL.
*
 01  EMESSAGE.
   02  EMSG0000.
     03  FILLER    PICTURE X(16)
            VALUE "EXIT=EKYEFL1C - ".
     03  FILLER    PICTURE X(44)
            VALUE "SOURCE LENGTH NOT SPECIFIED - REQUIRED.      ".
     03  FILLER    PICTURE X(04)
            VALUE "    ".
   02  EMSG0010.
     03  FILLER    PICTURE X(16)
            VALUE "EXIT=EKYEFL1C - ".
     03  FILLER    PICTURE X(44)
            VALUE "SOURCE LENGTH EXCEEDS MAXIMUM ALLOWED.       ".
```

Figure 39 (Part 4 of 9). Second Sample Field Exit Routine (COBOL)

```
           03  FILLER    PICTURE X(04)
                   VALUE "    ".
         02  EMSG0020.
           03  FILLER    PICTURE X(16)
                   VALUE "EXIT=EKYEFL1C - ".
           03  FILLER    PICTURE X(44)
                   VALUE "TARGET LENGTH NOT SPECIFIED - REQUIRED.     ".
           03  FILLER    PICTURE X(04)
                   VALUE "    ".
         02  EMSG0030.
           03  FILLER    PICTURE X(16)
                   VALUE "EXIT=EKYEFL1C - ".
           03  FILLER    PICTURE X(44)
                   VALUE "TARGET LENGTH EXCEEDS MAXIMUM ALLOWED.      ".
           03  FILLER    PICTURE X(04)
                   VALUE "    ".
         02  EMSG0040.
           03  FILLER    PICTURE X(16)
                   VALUE "EXIT=EKYEFL1C - ".
           03  FILLER    PICTURE X(44)
                   VALUE "TARGET DATA TYPE MUST BE CHARACTER.         ".
           03  FILLER    PICTURE X(04)
                   VALUE "    ".
         02  EMSG0050.
           03  FILLER    PICTURE X(16)
                   VALUE "EXIT=EKYEFL1C - ".
           03  FILLER    PICTURE X(44)
                   VALUE "TARGET SCALE MUST NOT BE SPECIFIED.         ".
           03  FILLER    PICTURE X(04)
                   VALUE "    ".
         02  EMSGD000.
           03  FILLER PICTURE X(16)
                   VALUE "EXIT=EKYEFL1C - ".
           03  FILLER    PICTURE X(44)
                   VALUE "DATA TYPE CALL FUNCTION CANNOT BE IDENTIFIED".
           03  FILLER    PICTURE X(04)
                   VALUE ".   ".
      /*
      ******************************************************************
      *
       LINKAGE SECTION.
      *
      ******************************************************************
      *  THE FOLLOWING CONTROL BLOCK IS SHIPPED WITH THE DPROP PRODUCT*
      ******************************************************************
      *
          COPY  EKYRCUDC.
      *
      ******************************************************************
      *  THIS DESCRIBES THE SOURCE FIELD TO BE CONVERTED              *
      ******************************************************************
      *
       01  SRCFIELD.
```

Figure 39 (Part 5 of 9). Second Sample Field Exit Routine (COBOL)

```
030100   02  SRCBYTE     PICTURE X      USAGE DISPLAY OCCURS 16 TIMES.  02950000
030200*                                                                 02960000
030300****************************************************************** 02970000
030400*  THIS DESCRIBES THE TARGET FOR THE CONVERTED OUTPUT           * 02980000
030500****************************************************************** 02990000
030600*                                                                 03000000
030700 01  TARFIELD.                                                    03010000
030800   02  TARBYTE     PICTURE X      USAGE DISPLAY OCCURS 128 TIMES. 03020000
030900*                                                                 03030000
031000****************************************************************** 03040000
031100*  THIS 64 BYTE USERAREA IS FOR THE EXCLUSIVE USE OF THIS       * 03050000
031200*  EXIT. ITS CONTENTS WILL BE PRESERVED BETWEEN CALLS.          * 03060000
031300*  IT IS INITIALIZED TO BINARY ZEROS.                           * 03070000
031400****************************************************************** 03080000
031500*                                                                 03090000
031600 01  USERAREA.                                                    03100000
031700   02  SPECAREA    PICTURE S99999 USAGE COMPUTATIONAL.            03110000
031800   02  SPECARE2    PICTURE S99999 USAGE COMPUTATIONAL.            03120000
031900   02  TARNUMBER   PICTURE S99999 USAGE COMPUTATIONAL.            03130000
032000   02  TESTITX  REDEFINES  TARNUMBER.                             03140000
032100     03  TOPPART   PICTURE XXX    USAGE DISPLAY.                  03150000
032200     03  TESTPART  PICTURE X      USAGE DISPLAY.                  03160000
032300   02  SCOUNT      PICTURE S9999  USAGE COMPUTATIONAL.            03170000
032400   02  TCOUNT      PICTURE S9999  USAGE COMPUTATIONAL.            03180000
032500   02  BCOUNT      PICTURE S9999  USAGE COMPUTATIONAL.            03190000
032600   02  FUNCVALD    PICTURE X      USAGE DISPLAY.                  03200000
032700/*                                                                03210000
032800****************************************************************** 03220000
032900*                                                                 03230000
033000 PROCEDURE DIVISION USING EKYRCUDC                                03240000
033100                          SRCFIELD                                03250000
033200                          TARFIELD                                03260000
033300                          USERAREA.                               03270000
033400*                                                                 03280000
033500***      SET CONTROL FLAGS - EXIT ENTERED, EXIT IN CONTROL,       03290000
033600***      FUNCTION NOT IDENTIFIED.                                 03300000
033700*                                                                 03310000
033800     MOVE "X" TO UDTENTRD.                                        03320000
033900     MOVE "X" TO UDTINCTL.                                        03330000
034000     MOVE " " TO FUNCVALD.                                        03340000
034100*                                                                 03350000
034200***      SELECT THE REQUIRE PROCESSING ROUTINE BASED              03360000
034300***      ON CALL FUNCTION                                         03370000
034400*                                                                 03380000
034500***      1. DXT-UIM DEFINE CALL                                   03390000
034600*                                                                 03400000
034700     IF  UDTCDEFN THEN                                            03410000
034800         MOVE "X" TO FUNCVALD                                     03420000
034900         PERFORM UIMVAL00 THROUGH UIMVALX0.                       03430000
035000*                                                                 03440000
035100***      2. DPROP/DataRefresher SOURCE TO TARGET                            03450000
035200*                                                                 03460000
035300     IF  UDTCSRTG THEN                                            03470000
035400         MOVE "X" TO FUNCVALD                                     03480000
035500         PERFORM SRCTART0 THROUGH SRCTARTX.                       03490000
035600*                                                                 03500000
035700***      3. DPROP TARGET TO SOURCE                                03510000
035800*                                                                 03520000
035900     IF  UDTCTGSR THEN                                            03530000
036000         MOVE "X" TO FUNCVALD                                     03540000
```

Figure 39 (Part 6 of 9). Second Sample Field Exit Routine (COBOL)

```
         PERFORM TARTSRC0 THROUGH TARTSRCX.
*
***     4. CALL FUNCTION IS UNIDENTIFIED
*
     IF  FUNCVALD NOT EQUAL "X" THEN
***     SET MESSAGE AND TERMINATE RETURN CODE
         MOVE EMSGD000 TO UDTXMESG
         MOVE 16 TO UDTXRETC.
*
     GOBACK.
/*
******************************************************************
* PROCEDURE: UIMVAL00                                            *
*                                                                *
* FUNCTION: VALIDATE VALUES SET IN THE CONTROL BLOCK FOR THIS   *
*          DATA TYPE. SET CONTROL BLOCKS VALUES WHEN REQUIRED.  *
******************************************************************
*
*    IF  SOURCE LGT IS NOT ZERO
*    .   IF  SOURCE LGT NOT TOO LONG
*    .   .   IF TARGET TYPE VARIABLE
*    .   .       SET TARGET LGT=(SRC LGT*8)
*    .   .   ELSE USE PASSED TARGET LENGTH
*    .   .   END-IF
*    .   .   IF  TARGET IS NOT ZERO
*    .   .   .   IF  TARGET LGT NOT TOO LONG
*    .   .   .   .   IF  TARGET DATA TYPE = CHARACTER
*    .   .   .   .   .   IF  TARGET SCALE = "N"
*    .   .   .   .   .       SET RC=0: VALIDATION SUCCESSFUL
*    .   .   .   .   .   ELSE
*    .   .   .   .   .       MESSAGE = EMSG0050
*    .   .   .   .   ELSE
*    .   .   .   .       MESSAGE = EMSG0040
*    .   .   .   ELSE
*    .   .   .       MESSAGE = EMSG0030
*    .   .   ELSE
*    .   .       MESSAGE = EMSG0020
*    .   ELSE
*    .       MESSAGE = EMSG0010
*    ELSE
*        MESSAGE = EMSG0000.
*
 UIMVAL00.
     MOVE 4 TO UDTXRETC.
     IF  UDTSBYTV GREATER THAN 0 THEN
         IF  UDTSBYTV NOT GREATER THAN MAXSRCLG THEN
             IF  UDTTBYTI = "V" THEN
                 COMPUTE UDTTBYTV = UDTSBYTV * 8
             END-IF
             IF  UDTTBYTV GREATER THAN 0 THEN
                 IF  UDTTBYTV NOT GREATER THAN MAXTARLG THEN
                     IF  UDTTTYPE IS EQUAL TO  "C " THEN
                         IF  UDTTSCLN THEN
                             MOVE 0 TO UDTXRETC
                         ELSE
                             MOVE EMSG0050 TO UDTXMESG
                     ELSE
                         MOVE EMSG0040 TO UDTXMESG
                 ELSE
                     MOVE EMSG0030 TO UDTXMESG
```

Figure 39 (Part 7 of 9). Second Sample Field Exit Routine (COBOL)

```
             ELSE
                 MOVE EMSG0020 TO UDTXMESG
         ELSE
             MOVE EMSG0010 TO UDTXMESG
     ELSE
         MOVE EMSG0000 TO UDTXMESG.
 UIMVALX0. EXIT.
/*
*****************************************************************
* PROCEDURE: SRCTART0                                           *
*                                                               *
* FUNCTION:  CONVERT A EACH SOURCE BYTE TO A  "ON" OR "OFF'     *
*            CHARACTER.                                         *
*****************************************************************
*
 SRCTART0.
*
     MOVE 0 TO UDTXRETC.
*
     MOVE ZERO TO SCOUNT.
     MOVE ZERO TO TCOUNT.
     MOVE ZERO TO BCOUNT.
*
***      MOVE NEXT SOURCE BYTE TO BINARY WORD FOR BIT CHECK.
*
 GETNXSRC.
     MOVE 0 TO BCOUNT.
     IF  SCOUNT IS LESS THAN UDTSBYTV THEN
         ADD  1    TO SCOUNT
         MOVE ZERO TO TARNUMBER
         MOVE SRCBYTE(SCOUNT) TO TESTPART.
*
***      SET NEXT TARGET BYTE TO THE "ON" VALUE OR THE "OFF"
***      VALUE DEPENDING ON THE CORRESPONDING BIT BEING 1 OR 0
*
 TOTARGET.
     ADD 1    TO BCOUNT.
     ADD 1    TO TCOUNT.
*
     IF  TCOUNT IS GREATER THAN UDTTBYTV THEN
         GO TO SRCTARTX.
*
     MOVE XDBITOFF TO TARBYTE(TCOUNT).
*
     IF  SCOUNT IS NOT GREATER THAN UDTSBYTV THEN
         ADD TARNUMBER TO TARNUMBER
         IF  TARNUMBER IS NOT LESS THAN 256 THEN
             SUBTRACT 256 FROM TARNUMBER
             MOVE XDBITON  TO TARBYTE(TCOUNT).
*
     IF  BCOUNT EQUAL TO 8 THEN
         GO TO GETNXSRC
     ELSE
         GO TO TOTARGET.
 SRCTARTX.
/*
*****************************************************************
* PROCEDURE: TARTSRC0                                           *
*                                                               *
* FUNCTION:  CONVERT A EACH 'TARGET' BYTE                       *
```

Figure 39 (Part 8 of 9). Second Sample Field Exit Routine (COBOL)

```
048100*           TO A '0' OR '1' BIT.                              * 04750000
048200*                                                             * 04760000
048300*** IN THE FOLLOWING PROCESS, THE 'TARGET' IS THE SENDING    **** 04770000
048400*** FIELD AND THE 'SOURCE' IS THE RECEIVING FIELD           **** 04780000
048500*                                                             * 04790000
048600***************************************************************** 04800000
048700*                                                               04810000
048800 TARTSRC0.                                                      04820000
048900*                                                               04830000
049000     MOVE ZERO TO UDTXRETC.                                     04840000
049100     MOVE ZERO TO TCOUNT.                                       04850000
049200     MOVE ZERO TO SCOUNT.                                       04860000
049300*                                                               04870000
049400***      PROCESS FIRST OR NEXT 'SOURCE' BYTE.                   04880000
049500*                                                               04890000
049600 TARTSRC1.                                                      04900000
049700     MOVE 256  TO XTVALUE                                       04910000
049800     MOVE ZERO TO TARNUMBER.                                    04920000
049900     MOVE ZERO TO BCOUNT.                                       04930000
050000     ADD  1    TO SCOUNT                                        04940000
050100*                                                               04950000
050200***      WHEN ALL 'SOURCE' BYTES ARE FILLED, THEN STOP          04960000
050300***      ELSE, INITIALIZE THE 'SOURCE' BYTE TO ZERO.            04970000
050400*                                                               04980000
050500     IF  SCOUNT IS GREATER THAN UDTSBYTV THEN                   04990000
050600         GO TO TARTSRCX                                         05000000
050700     ELSE                                                       05010000
050800         MOVE TESTPART TO SRCBYTE(SCOUNT).                      05020000
050900*                                                               05030000
051000***      SET NEXT 'SOURCE' BIT TO 0 OR TO 1 DEPENDING IF        05040000
051100***      THE CORRESPONDING 'TARGET' BYTE IS '0' OR '1'          05050000
051200*                                                               05060000
051300 TARTSRC2.                                                      05070000
051400     COMPUTE XTVALUE = XTVALUE / 2.                             05080000
051500     ADD  1    TO BCOUNT.                                       05090000
051600     ADD  1    TO TCOUNT.                                       05100000
051700*                                                               05110000
051800     IF  TCOUNT IS GREATER THAN UDTTBYTV THEN                   05120000
051900         GO TO TARTSRC1.                                        05130000
052000*                                                               05140000
052100     IF  TARBYTE(TCOUNT) = "1"                                  05150000
052200         ADD XTVALUE TO TARNUMBER.                              05160000
052300*                                                               05170000
052400     IF  BCOUNT EQUAL TO 8 THEN                                 05180000
052500         MOVE TESTPART TO SRCBYTE(SCOUNT)                       05190000
052600         GO TO TARTSRC1                                         05200000
052700     ELSE                                                       05210000
052800         GO TO TARTSRC2.                                        05220000
052900 TARTSRCX.                                                      05230000
053000*                                                               05240000
053300***************************************************************** 05250000
```

Figure 39 (Part 9 of 9). Second Sample Field Exit Routine (COBOL)

Chapter 4. Propagation Exit Routines

If the generalized mapping cases are not flexible enough for your needs, you can use a Propagation exit routine. This type of exit routine supplies all its own mapping logic and propagating SQL or DL/I calls. DPROP calls the exit routine, which retains many of the DPROP support functions. This is the advantage a Propagation exit routine has over an IMS Data Capture exit routine (as described in *IMS/ESA Customization Guide*), or a DB2 Data Capture exit routine. These DPROP-supported functions are discussed below.

If you have specified the use of a Propagation exit routine for a particular PR, DPROP calls your exit routine as soon as it receives the changed data. DPROP does not use any of its own mapping logic; instead, it relies on your exit routine to perform any data transformations you need and to propagate the data to the DB2 table or IMS database.

Your exit routine can be written in Assembler, or in COBOL, PL/I, or C. The DPROP support for exit routines written in HLLs requires LE/370 Version 1 Release 2.

For synchronous propagation, DPROP calls your exits in both IMS batch and online dependent regions accessing DB2. For LOG-ASYNC propagation the RUP calls your exit routines in an MVS batch environment using CAF attach to DB2. For user asynchronous propagation, depending on your implementation, the RUP can call your exit routine in IMS batch and dependent regions accessing DB2, or in a non-IMS DB2 TSO or DB2 CAF environment.

Propagation exit routines differ from Segment and Field exit routines, in that DataRefresher does not call Propagation exit routines during data extraction. In some cases, you can use DataRefresher's more powerful mapping capabilities to extract and load the data. Otherwise, you must write your own programs to extract the IMS data. Loading the DB2 tables can then be done either by creating an input data set for the DB2 Load Utility, or by inserting the DB2 rows with SQL statements; this takes more time.

Propagation exit routines differ from Segment and Field exit routines, in that the DPROP DLU does not call Propagation exit routines. Data propagated by Propagation exit routines can be passed, using sequential files, to the DLU. See *IMS DPROP Reference* for more information.

To avoid propagation failures, the mapping performed during the extract and load must be compatible with the mapping that your Propagation exit routine performs.

Environment Considerations for a Propagation Exit Routine

S In Synchronous propagation mode, your Propagation exit routine can be called by
S the RUP (when the propagation direction is HR) or by the HUP (when the
S propagation direction is RH). Because the RUP and the HUP run as as extensions
S of IMS mixed mode applications your Propagation exit routine runs as an IMS
S mixed mode application. This allows your Propagation exit routine to issue both
S DL/I calls and SQL calls, but you must link edit your Propagation exit routine with
S the DB2 language interface for IMS Attach.

In LOG-ASYNC propagation mode, your Propagation exit routine can only be called by the RUP (propagation direction is always HR). The RUP is called by the Receiver which runs as an MVS application with a CAF Attach to DB2. This means that your Propagation exit routine can **only issue SQL calls**. In this case, you must link edit your Propagation exit routine with the DB2 language interface for CAF Attach.

In User Asynchronous propagation mode, your Propagation exit routine can only be called by the RUP (propagation direction is always HR). The RUP is called by your own user-written receiver programs which can run either as an IMS application, a TSO application or as an MVS application with CAF Attach, depending on how you design it. If you design your own user-written receiver programs to run as an IMS mixed-mode application, then you can issue both DL/I calls and SQL calls from your Propagation exit routine.

It is recommended in all of the above cases that you code and link-edit your Propagation exit routine as reentrant. You must also link-edit your Propagation exit routine with the DPROP Trace Module EKYR410X.

How To Write A Propagation Exit Routine

Because you supply your own mapping logic and SQL or DL/I calls, DPROP is very flexible regarding the structure of your Propagation exit routine. You can even propagate data changes to more than one DB2 table. DPROP does not impose or check rules for the mapping of keys or referential integrity relationships (RIRs). DPROP also does not support the CCU and DLU. Mapping and verifying data propagation is left up to you.

Before discussing the development of your exit routine, the next section briefly lists which functions DPROP supports when using a Propagation exit.

Supported DPROP Functions

As mentioned above, DPROP does not impose or check rules for the mapping of keys or RIRs. Also, Propagation exits do not support the use of the CCU and the DLU.

However, DPROP still supports the following features when you use a Propagation exit routine:

- DPROP-provided tracing support
- DPROP-provided Audit support
- Standardized error handling
- Orderly suspension of propagation
- Activation or deactivation of PRs
- Emergency stops of all propagating activities
- The PROP OFF //EKYIN control statement
- Protection against unintentional updates during IMS extract and DLU processing
- Propagation definitions recorded in the DPROP directory
- Optional DBD version checking (for HR-propagation)

Although you control the propagation of changed data, DPROP still provides some of the valuable functions available to generalized mapping cases.

Creating your own user mapping with a Propagation exit routine, instead of using an IMS Data Capture exit routine, or DB2 Data Capture exit routine, helps establish a common process for managing the data propagation environment for both generalized and user mapping cases.

Propagation Exit Routine Interface

When DPROP receives the changed data, it calls your Propagation exit routine.

1. The RUP calls your Propagation exit routine for IMS-to-DB2 mapping with an interface similar to the IMS Data Capture Exit interface. The following control blocks are passed:
   - The Propagation Interface Control Block (PIC)
   - The Extended Program Communication Block (XPCB)

     The XPCB is a control block that IMS defines; it describes the changed IMS data.
2. The HUP calls your Propagation exit routine for DB2-to-IMS mapping with the following control blocks:
   - The Propagation Interface Control Block (PIC)
   - The HUP Exit Communication Block (HEC)

     The HEC is a control block that DPROP defines; it contains pointers to areas that the DB2 Data Capture exit passes.

Register 1 points to a list that is two fullwords long, containing the addresses of these control blocks.

Propagation Interface Control Block (PIC)

There is one interface control block per exit routine, lasting for the duration of the exit in virtual storage.

You can generate the following DSECT in your assembler exit routine by coding the EKYRCPIC macro statement. For HLL exit routines, you can include or copy one of the following members to map the Propagation exit routine Interface Control Block:

EKYRCPCC Exit routines written in COBOL
EKYRCPCP Eit routines written in PL/I
EKYRCPCK Exit routines written in C

Figure 40 on page 157 shows the structure of the control block, and is followed by a detailed description of its fields.

```
                                   1          EKYRCPIC
                                   2+****************** START OF CONTROL BLOCK SPECIFICATION **************/
                                   3+*                                                                     */
                                   4+*        CONTROL BLOCK NAME:                                          */
                                   5+*          EKYRCPIC (PIC)                                             */
                                   6+*                                                                     */
                                   7+*        DESCRIPTIVE NAME:                                            */
                                   8+*          DPROP PROPAGATION EXIT INTERFACE BLOCK                     */
                                   9+*                                                                     */
                                  10+*                                                                     */
                                  11+**********************************************************************
                                  12+*                                                                     *
                                  13+*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
                                  14+*                                                                     *
                                  15+*          5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
                                  16+*          ALL RIGHTS RESERVED.                                       *
                                  17+*                                                                     *
                                  18+*          U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
                                  19+*          USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
                                  20+*          GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
                                  21+*                                                                     *
                                  22+*          LICENSED MATERIALS - PROPERTY OF IBM.                      *
                                  23+*                                                                     *
                                  24+**********************************************************************
                                  25+*                                                                     */
                                  26+*        STATUS: V1 R2 M0                                             */
                                  27+*                                                                     */
                                  28+*        FUNCTION:                                                    */
                                  29+*          THIS IS THE CONTROL BLOCK USED TO INTERFACE BETWEEN        */
                                  30+*               - DPROP                                               */
                                  31+*               AND                                                   */
                                  32+*               - A USER'S PROPAGATION  EXIT ROUTINE                  */
                                  33+*                                                                     */
                                  34+*          THERE IS ONE PIC CB FOR EACH EXIT PROPAGATION              */
                                  35+*          EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT         */
                                  36+*          IN VIRTUAL STORAGE.                                        */
                                  37+*          FOR SYNCH PROPAGATION IN MPP REGIONS:                      */
                                  38+*          - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER       */
                                  39+*            SUBTASK.                                                 */
                                  40+*          FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR            */
                                  41+*          ASYNCH PROPAGATION, AND FOR CCU PROCESSING:                */
                                  42+*          - THIS IS THE DURATION OF THE JOBSTEP.                     */
                                  43+*                                                                     */
                                  44+*        MODULE TYPE= MACRO                                           */
                                  45+*           PROCESSOR= ASSEMBLER H                                    */
                                  46+*                                                                     */
                                  47+*        INNER CONTROL BLOCKS: NONE                                   */
                                  48+*                                                                     */
                                  49+*        MACROS USED FROM MACRO LIBRARY: NONE                         */
                                  50+*                                                                     */
                                  51+*        CHANGE ACTIVITY:                                             */
                                  52+*                      KMP0057  12/13/90                              */
                                  53+*                      KMP0060  02/08/91 COPYRIGHT INFORMATION        */
                                  54+*                                                                     */
                                  55+******************** END OF CONTROL BLOCK SPECIFICATION **************/

000000                            57+PIC      DSECT
                                  58+*---------------------------------------------------------------*
                                  59+*              THIS SECTION CONTAINS INFORMATION PROVIDED BY    *
                                  60+*              DPROP TO THE INVOKED EXIT AT ENTRY TO CALL. THIS *
                                  61+*              SECTION MUST NOT BE MODIFIED BY THE EXIT.        *
                                  62+*---------------------------------------------------------------*
```

Figure 40 (Part 1 of 4). Interface Control Block for a Propagation Exit Routine

```
000000 C5D2E8D9C3D7C9C3            64+PICEYE   DC    CL8'EKYRCPIC'      EYE CATCHER
000008 4040404040404040            65+PICEXIT  DC    CL8' '             NAME OF THE EXIT ROUTINE
000010 4040                        66+PICCALL  DC    CL2' '             TYPE OF CALL TO EXIT
                                   67++                                 ...'HR': HIERARCH TO RELATIONAL PROP
                                   68++                                 ...'RH': REL. TO HIERARCH
000012 00                          69+PICDBLEV DC    X'00'              DEBUG LEVEL IN EFFECT
                          00002    70+PICDBLV2 EQU   X'02'                 2 : EXTERNAL TRACE OF PROPAGATING
                                   71++                                        SQL STATEMENTS AND DL/I CALLS
000013 00                          72+         DC    X'00'              RESERVED
000014 00000000                    73+PICPTD   DC    A(0)               A(DPROP PTD)
000018 4040404040404040            74+PICPRID  DC    CL8' '             PR-ID
000020 4040404040404040            75+PICPRSET DC    CL8' '             PRSET-ID
000028 4040404040404040            76+PICPRTST DC    CL26' '            PR TIMESTAMP
000042 0000                        77+         DC    XL2'00'            RESERVED
000044 4040404040404040            78+PICPCBLA DC    CL8' '             PCB LABEL AS SPECIFIED ON PR
00004C 0000000000000000            79+         DC    XL56'00'           RESERVED
000084 40404040                    80+PICOPSYS DC    CL4' '             OPERATING SYSTEM
                                   81++                                 ...'ESA ': MVS/ESA
000088 40404040                    82+PICTRANS DC    CL4' '             IMS REGION TYPE
                                   83++                                 ...'MPP ': MPP REGION
                                   84++                                 ...'IFP ': IMS FAST PATH REGION
                                   85++                                 ...'BMP ': IMS BMP REGION
                                   86++                                 ...'BAT ': IMS BATCH REGION
                                   87++                                 ...'    ': IF NONE OF ABOVE
00008C 40404040                    88+PICPROGM DC    CL4' '             CALLING PROGRAM
                                   89++                                 ...'DPRS': DPROP SYNCH PROPAGATION
                                   90++                                 ...'DPRA': DPROP ASYNCH PROPAGATION
000090 0000000000000000            91+         DC    XL12'00'           RESERVED FOR DPROP


                                   93++*-----------------------------------------------------------------*
                                   94++              THIS SECTION IS USED BY THE EXIT TO PROVIDE       *
                                   95++              INFORMATION TO DPROP                              *
                                   96++*-----------------------------------------------------------------*

00009C 40                          98+PICENTRD DC    CL1' '                 SET BY EXIT ROUTINE TO
                                   99++                                     C'X', INDICATES
                                  100++                                     THAT EXIT HAS BEEN ENTERED
                                  101++
00009D 40                         102+PICINCTL DC    CL1' '                 SET BY EXIT ROUTINE TO
                                  103++                                     C'X', INDICATES
                                  104++                                     THAT EXIT IS IN CONTROL

                                  106+*********
                                  107+*********      RETURN CODE AND ERROR MESSAGE
                                  108+*********

00009E 0000                       110+PICXRETC DC    H'0'               RETURN CODE
                                  111++                                 ...4: SQL ERROR
                                  112++                                       SQL ERROR CODE IS IN THE FIELD
                                  113++                                       SQLCODE OF THE SQLCA
                                  114++                                 ...8: DLI ERROR
                                  115++                                       AIBRETRN, AIBREASN AND
                                  116++                                       DLI STATUS CODE IN PCB
                                  117++                                       POINTED BY AIBRSA1
                                  118++                                 ..12: ERROR OTHER THAN SQL ERROR:
                                  119++                                       SOME RESOURCES NOT AVAILABLE
                                  120++                                 ..16: ERROR OTHER THAN SQL ERROR:
                                  121++                                       NOT A RESOURCE AVAILABILITY
                                  122++                                       PROBLEM.
                                  123++                                 ..20: SHOULD NOT OCCUR/SHOULD ABEND
                                  124++
0000A0                            125+PICXMESG DS    0CL280             USER EXIT ERROR/WARNING MESSAGE
                                  126++                                     DPROP WILL WRITE THE MESSAGE
                                  127++                                     TO VARIOUS DESTINATIONS ACCORDING
                                  128++                                     TO USUAL DPROP/RUP ERROR HANDLING
```

Figure 40 (Part 2 of 4). Interface Control Block for a Propagation Exit Routine

```
                                  129++*                                LOGIC.
0000A0                            130+PICXML1  DS    0CL70' '          1ST MESSAGE LINE
0000A0                            131+PICXMSGI DS    CL8' '            ...8 BYTES MESSAGE ID
0000A8                            132+PICXMSGB DS    C' '              ...ONE BLANK
0000A9                            133+PICXMTXT DS    CL61' '           ...61 TEXT BYTES IN 1ST MESSAGE LINE
0000E6                            134+PICXML2  DS    CL70' '           2ND MESSAGE LINE
00012C                            135+PICXML3  DS    CL70' '           3RD MESSAGE LINE
000172                            136+PICXML4  DS    CL70' '           4TH MESSAGE LINE
                                  137++*
0001B8 0000000000000000           138+         DC    XL12'00'          RESERVED FOR DPROP

                                  140+*********
                                  141+*********      NAME OF OBJECTS ASSOCIATED WITH ERROR
                                  142+*********

0001C4 4040404040404040           144+PICDBN   DC    CL8' '            DBDNAME ASSOCIATED WITH THE ERROR
0001CC 4040404040404040           145+PICSEGN  DC    CL8' '            SEG NAME ASSOCIATED WITH THE ERROR
0001D4 4040404040404040           146+PICTABQ  DC    CL8' '            TABLE NAME QUALIFIER ASSOC. W. ERROR
0001DC 4040404040404040           147+PICTABN  DC    CL18' '           TABLE NAME ASSOCIATED WITH THE ERROR
0001EE 0000000000000000           148+         DC    XL14'00'          RESERVED FOR DPROP


                                  150++*-------------------------------------------------------------*
                                  151++*             EXIT WORK AREA                                  *
                                  152++*                                                             *
                                  153++*             THE EXIT WORK AREA CAN BE USED TO SAVE          *
                                  154++*             INFORMATION ACROSS CALLS TO THE EXIT (E.G.      *
                                  155++*             TO SAVE THE ADDRESSES OF GETMAINED AREAS ACROSS *
                                  156++*             CALLS TO THE EXIT.                              *
                                  157++*-------------------------------------------------------------*

000200                            159+         DS    0D
000200 0000000000000000           160+PICSWORK DC    XL256'00'         WORK AREA FOR THE EXIT
000300 0000000000000000           161+         DC    XL16'00'          RESERVED FOR DPROP


                                  163++*-------------------------------------------------------------*
                                  164++*             SQL COMMUNICATION AREA (SQLCA).                 *
                                  165++*                                                             *
                                  166++*             THE EXIT SHOULD USE THIS SQLCA FOR ITS SQL      *
                                  167++*             STATEMENTS.                                     *
                                  168++*-------------------------------------------------------------*

000310                            170+SQLCA            DS 0D
000310                            171+SQLCID           DS CL8          ID
000318                            172+SQLCABC          DS F            BYTE COUNT
00031C                            173+SQLCODE          DS F            RETURN CODE
000320                            174+SQLERRM          DS H,CL70       ERROR MSG PARMS
000368                            175+SQLERRP          DS CL8          IMPL DEPENDENT
000370                            176+SQLERRD          DS 6F
000388                            177+SQLWARN          DS 0C           WARNING FLAGS
000388                            178+SQLWARN0         DS C'W'         IF ANY
000389                            179+SQLWARN1         DS C'W'         = WARNING
00038A                            180+SQLWARN2         DS C'W'         = WARNING
00038B                            181+SQLWARN3         DS C'W'         = WARNING
00038C                            182+SQLWARN4         DS C'W'         = WARNING
00038D                            183+SQLWARN5         DS C'W'         = WARNING
00038E                            184+SQLWARN6         DS C'W'         = WARNING
00038F                            185+SQLWARN7         DS C'W'         = WARNING
000390                            186+SQLEXT           DS CL8
000398                            187+                 DS 4F           RESERVED
```

Figure 40 (Part 3 of 4). Interface Control Block for a Propagation Exit Routine

```
                                  189++*-------------------------------------------------------------------*
                                  190++*             DLI APPLICATION INTERFACE BLOCK (AIB)                 *
                                  191++*                                                                   *
                                  192++*             THE EXIT SHOULD USE THIS AIB FOR ITS DLI              *
                                  193++*             CALL. BEFORE FIRST CALL, DPROP INITS                  *
                                  194++*             AIBID, AIBLEN, AIBRSNM1 AND AIBSFUNC FIELDS.          *
                                  195++*                                                                   *
                                  196++*-------------------------------------------------------------------*

0003A8                            198+PICAIB   DS    0D                 AIB INITIALIZED BY DPROP
0003A8                            199+PIC_AIBID      DS    CL8'DFSAIB'  EYECATCHER
0003B0                            200+PIC_AIBLEN     DS    F            DFSAIB ALLOCATED LENGTH
0003B4                            201+PIC_AIBSFUNC   DS    CL8          SUBFUNCTION CODE
0003BC                            202+PIC_AIBRSNM1   DS    CL8          RESOURCE NAME 1
0003C4                            203+PIC_AIBRSNM2   DS    CL8          RESOURCE NAME 2
0003CC                            204+               DS    2F           RESERVED
0003D4                            205+PIC_AIBOALEN   DS    F            OUTPUT AREA LENGTH (MAX)
0003D8                            206+PIC_AIBOAUSE   DS    F            OUTPUT AREA LENGTH (USED)
0003DC                            207+               DS    2F           RESERVED
0003E4                            208+               DS    H            RESERVED
0003E6                            209+               DS    H            RESERVED
0003E8                            210+PIC_AIBRETRN   DS    F            RETURN CODE
0003EC                            211+PIC_AIBREASN   DS    F            REASON CODE
0003F0                            212+               DS    F            RESERVED
0003F4                            213+PIC_AIBRSA1    DS    A            RESOURCE ADDRESS 1
0003F8                            214+PIC_AIBRSA2    DS    A            RESOURCE ADDRESS 2
0003FC                            215+PIC_AIBRSA3    DS    A            RESOURCE ADDRESS 3
000400                            216+               DS    10F          RESERVED
                          00080   217+PIC_AIBLL      EQU   *-PICAIB     DFSAIB LENGTH
000428                            218+                  DS 4F           RESERVED

                          00438   220+PICEND   EQU   *                  END OF PIC
                          00438   221+PICLEN   EQU   *-PIC              LENGTH OF PIC
                                  222          END
```

Figure 40 (Part 4 of 4). Interface Control Block for a Propagation Exit Routine

Interface Control Block Field Descriptions

The following is a detailed description of the control block fields:

PICEYE Contains the constant **EKYRCPIC**, and is used to identify the control block in a dump.

PICEXIT The load module name of the exit routine.

PICCALL The call function that DPROP sets to **HR** to indicate hierarchical-to-relational or to **RH** to indicate relational-to-hierarchical propagation.

PICDBLEV Contains the DPROP trace debug level in effect. If the PICDBLV2 bit is on, it indicates that you want to trace the propagating SQL statements for HR-propagation, and the propagating IMS calls for RH-propagation. The exit routine can then call the DPROP trace module.

PICPTD Address of an internal DPROP control block that the exit needs for calls to the DPROP trace module.

PICPRID The ID of the PR.

PICPRSET The Set ID of the PR.

PICPRTST The PR time stamp, assigned when MVG processed the PR.

PICOPSYS Set to **ESA** to define the operating system.

PICTRANS Identifies the IMS region type in which the exit routine is called. This field is blank if the exit routine is called from outside an IMS region— for example, during LOG-ASYNC propagation or user asynchronous propagation.

PICPROGM Describes the program calling the exit routine. Set to DPRS for synchronous propagation or DPRA for LOG-ASYNC propagation or user asynchronous propagation.

The next two fields are switches that are useful for problem determination. DPROP does not require your exit routine to set these fields. However, they can help you determine where a problem occurred if you have an ABEND. DPROP sets these fields to blanks before the first time your exit routine is called.

PICENTRD When you enter your exit routine, set this field to **X**. DPROP does not change this field again, so if a problem occurs, you can determine if your exit has been entered.

PICINCTL You must also set this field to **X**, indicating that your exit routine has control. When DPROP regains control, it resets this field to blanks, so you can determine if your exit routine has control when an ABEND occurs.

The next two fields can be used along with the RUP's and HUP's error handling logic. For more information on return codes and error handling techniques, see “Return Codes and Error Handling Techniques” on page 184.

PICXRETC The return code that the exit routine provides when returning to its caller. This field is set to zero when the exit routine is called.

0 Propagation was successful.

4 SQL error. Use return code 4 only if the failing SQL statement used the SQL communication area SQLCA provided in the interface control block.

8 DL/I call error. Use return code 8 only if the failing DL/I call used the DL/I Application Interface Block (AIB) provided in the interface control block.

12 Propagation failure (not caused by SQL or DL/I error); unavailable resource problem.

16 Propagation failure (not caused by SQL or DL/I error); Not an unavailable resource problem.

20 Severe error; DPROP ABENDs.

PICXMESG User-provided error message. It is set to blanks when the exit routine is called. When the exit routine returns, if the first eight bytes are not blank, DPROP writes the contents of the field as an error message with its usual error reporting logic. It is written as a four-line message with 70 bytes in each line. If the trailing lines contain only blanks, they are not written.

The message lines must have the following format:

- The first eight bytes of the first message line must be a message ID, beginning with a letter in the range J-Z (to avoid confusion with IBM-provided messages).

- The ninth character of the first message line must be blank.
- The remaining 61 bytes of the first message line, and the entire second, third, and fourth message lines, can all be used for your message text.

If your exit routine returns an error code to its caller, the following fields can be used to identify which data objects are associated with the error.

For HR-propagation:

PICTABQ Table name qualifier of the table involved in the error.
PICTABN Unqualified table name of the table involved in the error.

For-RH propagation:

PICDBN DBDNAME of the IMS database involved in the error.
PICSEGN Segment name of the segment involved in the error.

The following field is the work area for the exit routine.

PICSWORK The work area can be used to save information across calls to the exit routine. You can also use this field to hold the address of storage that the exit routine obtains the first time it gains control.

DPROP initializes this field to binary zeros before the first call to the exit routine, and never changes this field again. The contents of this field are saved until an application ABENDs in an MPP or an IFP region, when MVS releases the storage. After the ABEND, DPROP again initializes this field to binary zeros.

For these types of asynchronous propagation, the contents of this field are preserved until the end of the MVS task that the receiver program uses to call the RUP.

The PIC, and therefore the work area, is associated with an exit name. When an exit routine is called for multiple segments, tables, or multiple PRs, the work area is the same.

SQLCA This area is the SQL Communication Area, used for the SQL statements your exit routine executes. it is recommended that all SQL statements that your Propagation exit routine generates use this SQL communication area.

If your exit routine encounters an SQL error and returns with a return code of 4, DPROP uses the contents of this area to determine which type of SQL error occurred and to provide detailed error messages.

DFSAIB This area is the DL/I Application Interface Block (AIB) used for the DL/I calls your exit routine executes. it is recommended that all DL/I calls that your Propagation exit routine generates use this AIB.

If your exit routine encounters a DL/I error and returns with a return code of 8, DPROP uses the contents of this area to determine which type of DL/I error occurred and to provide detailed error messages.

Interface for HR Propagation

This section describes the interface used for HR-propagation. If your exit routine must not support HR-propagation, then you can skip this section and continue with the section “Interface for RH-Propagation” on page 171.

Interfaces between the RUP and your Propagation exit routine are the XPCB and the Extended Segment Data Block (XSDB). These are control blocks that the IMS Data Capture function defines; they are used to describe the changed IMS data.

The XPCB is the second parameter passed to your Propagation exit routine when the RUP calls it. It is used to provide information about the changed data and to point to XSDBs. An XSDB points to, and describes, either a changed segment occurrence or a physical ancestor of a changed segment.

Your exit routine must not modify the XPCB, the XSDB, or the data pointed to by these control blocks.

Figure 41 on page 164 provides an overview of the interface defined through the XPCB and XSDBs.

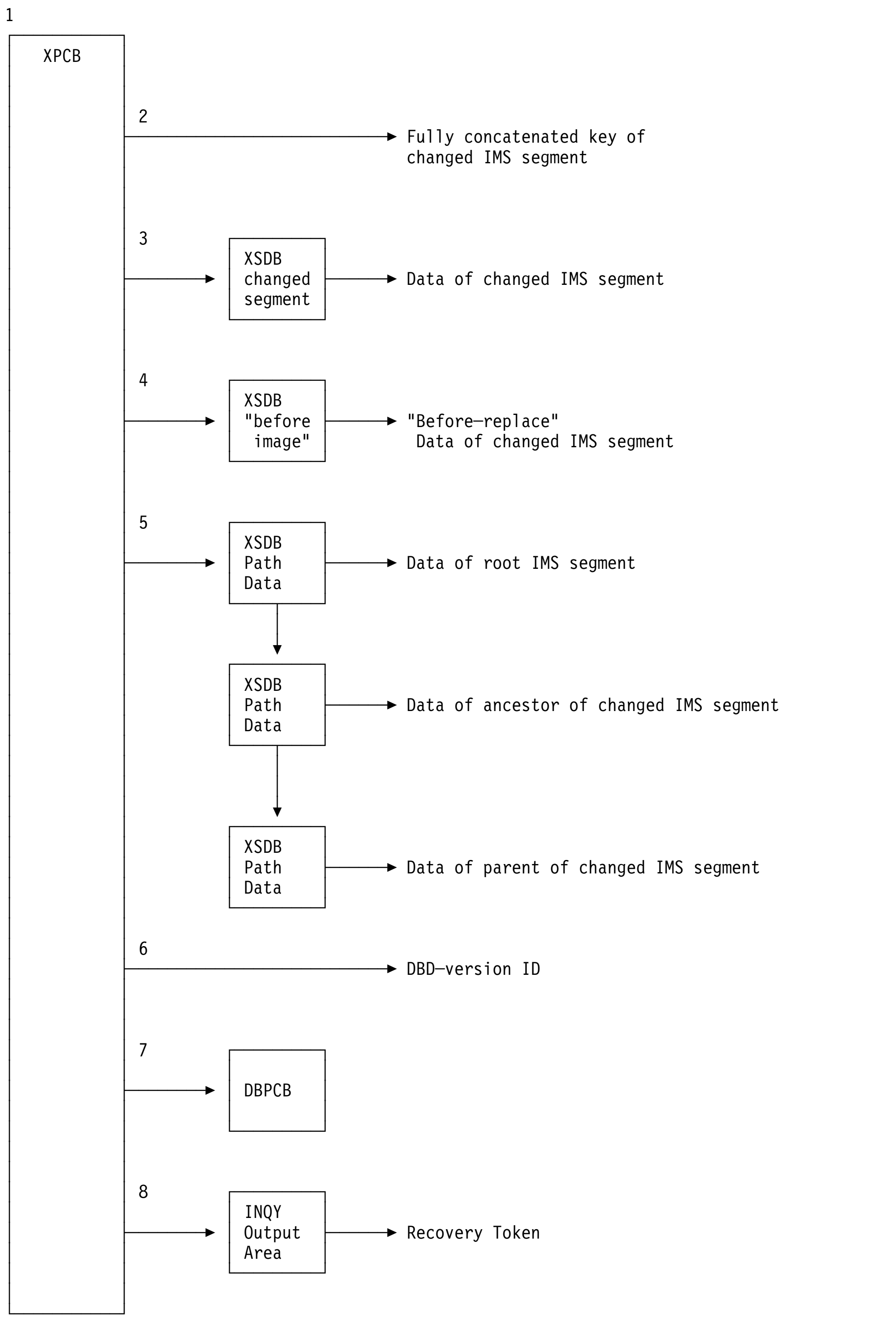

Figure 41. XPCB and XSDB Control Block Structures

As shown in the numbered sections of the figure, the interface consists of:

1. One XPCB control block that provides a description of the changed data and contains various pointers.
2. A pointer to the fully concatenated key of the changed segment.

3. A pointer to the XSDB control block describing the changed segment. This XSDB points to the data of the changed segment.
4. For Replace operations, a pointer to an XSDB describing the segment *before* it was replaced. The XSDB also points to the data of the *before-image* of the segment.
5. A pointer to the first XSDB in a chain of XSDBs for the hierarchical ancestors of the changed segment. The chain is in descending hierarchical order, with each XSDB pointing to the segment data of the segment and the next XSDB in descending order.
6. A pointer to the DBD version ID.
7. A pointer to the DB PCB.
8. A pointer to an area containing the output of an implied IMS INQY ENVIRON call.

The XPCB and XSDB Control Blocks

You can generate the following DSECTs in your assembler exit routine by coding the EKYRCDL1 macro statement. For HLL exit routines, you can include or copy one of the following members to map the XPCB and XSDB Control Blocks:

EKYRCDLC Exit routines written in COBOL
EKYRCDLP Exit routines written in PL/I
EKYRCDLK Exit routines written in C

XPCB DSECT

The XPCB control block is shown in Figure 42 on page 166 followed by a detailed description of those fields that are most useful to your exit routine.

```
                             1         EKYRCDL1

                             3+**********************************************************************
                             4+*                                                                    *
                             5+*        E X T E N D E D   D A T A  B A S E  P C B  -- X P C B       *
                             6+*                                                                    *
                             7+**********************************************************************

000000                       9+XPCB     DSECT
000000                      10+XPCBEYE   DS    CL4     "XPCB" EYECATCHER
000004                      11+XPCBVER   DS    CL2     XPCB VERSION INDICATOR
000006                      12+XPCBREL   DS    CL2     XPCB RELEASE INDICATOR
000008                      13+XPCBEXIT  DS    CL8     SEGMENT USER EXIT NAME
000010                      14+XPCBRC    DS    H       RETURN-CODE
000012                      15+XPCBRSNC  DS    H       REASON-CODE
000014                      16+XPCBDBD   DS    CL8     PHYSICAL DATA BASE NAME
00001C                      17+XPCBVERA  DS    A       ADDRESS OF DBD VERSION ID
000020                      18+XPCBSEG   DS    CL8     PHYSICAL SEGMENT NAME
000028                      19+XPCBCALL  DS    CL4     'CALL FUNCTION' DEFINED BY IMS/ESA
                            20+*                          ISRT: INSERT
                            21+*                          REPL: REPLACE
                            22+*                          DLET: DELETE
                            23+*                          CASC: CASCADING DELETE
                            24+*                          DLLP: NOW ALSO DELETED FROM LOGICAL PATH
00002C                      25+XPCBPCALL DS    CL4     'PHYSICAL UPDATE TYPE' DEFINED BY IMS
                            26+*                          ISRT: INSERT
                            27+*                          REIN: RE-INSERT VIA LOGICAL PATH
                            28+*                          REPL: REPLACE
                            29+*                          DLET: DELETE
                            30+*                          DLPP: DELETED ONLY FROM PHYSICAL PATH
000030                      31+          DS    CL4     RESERVED
000034                      32+XPCBPCBA  DS    A       ADDRESS OF DB PCB
000038                      33+XPCBPCBN  DS    CL8     NAME OF DB PCB
000040                      34+XPCBINQA  DS    A       ADDRESS OF "INQY" OUTPUT
000044                      35+XPCBIOPA  DS    A       ADDRESS OF I/O PCB
000048                      36+          DS    H       RESERVED
00004A                      37+XPCBCKEYL DS    H       LENGTH OF CONCATENATED KEY
00004C                      38+XPCBCKEYA DS    A       ADDRESS OF CONCATENATED KEY
000050                      39+XPCBXSDBD DS    A       ADDRESS OF XSDB FOR DATA
000054                      40+XPCBXSDBB DS    A       ADDRESS OF XSDB FOR REPL DATA
000058                      41+XPCBXSDBP DS    A       ADDRESS OF XSDB FOR PATH DATA
00005C                      42+          DS    F       RESERVED
000060                      43+          DS    F       RESERVED
000064                      44+          DS    F       RESERVED
000068                      45+XPCBEXIWP DS    A       ADDRESS OF 256-BYTE AREA RESERVED FOR EXIT
00006C                      46+          DS    F       RESERVED
000070                      47+          DS    F       RESERVED
000074                      48+XPCBTIMST DS    CL8     TIMESTAMP OF CALL
00007C                      49+          DS    F       RESERVED
                   00080    50+XPCBLEN   EQU   *-XPCB  LENGTH OF XPCB
```

Figure 42. Extended Program Communication Block (XPCB)

XPCB Field Descriptions

The fields you need to use are:

XPCBDBD The physical database name.

XPCBVERA A pointer to a variable-length character string that identifies the DBD version. Unless the character string is set from the DBD VERSION= keyword, it is the time stamp of the DBDGEN. The first two bytes contain the length of the string followed by the string itself.

XPCBSEG The name of the updated physical segment type.

XPCBCALL Depending on the IMS call function, this field contains one of the following values:

REPL The IMS application generated a Replace call.

ISRT The IMS application generated an Insert call.

DLET The IMS application generated a Delete call.

CASC The IMS application generated a Delete call that resulted in a cascading delete of the IMS segment being processed by the current call of the Propagation exit routine.

The following value can be provided when logical parent segment types have an IMS Logical delete rule, and are involved in a unidirectional logical relationship. The value is encountered both for the logical parent segment type, and for its physical ancestors.

DLLP The IMS application generated a Delete call that resulted in a delete from the logical path. This value is provided as a result of deleting the last logical child of a logical parent that was no longer accessible through a physical path (the logical parent segment was only accessible through its logical path). When the delete is completed, the logical parent segment is no longer accessible, either through logical or physical paths.

Refer to *IMS/ESA Customization Guide* for more information on this field.

XPCBPCALL The physical update function. This differs from the IMS call function and from the content of XPCBCALL. For example, when an application inserts a concatenated logical parent or child that was deleted on the same path, IMS performs a physical replace of the logical parent instead of an insert.

The logic of your Propagation exit routine depends on the combination of values in XPCBCALL and XPCBPCALL. Refer to "The XPCBPCALL, XPCBCALL, and XSDBPHP Fields" on page 170 for examples of valid logic.

XPCBPCALL can have the following values:

REPL A segment is replaced.

ISRT A segment is inserted.

DLET A segment is deleted. If the segment is involved in a logical relationship, it is no longer accessible by either its physical or logical paths.

The following two values can be provided when you have an IMS delete rule of LOGICAL with a unidirectional logical relationship. The values can be provided for both the logical parent segment type and its physical ancestors. For more information, see the appropriate *Administrators Guide* for your propagation mode.

DLPP A segment has been deleted from the physical path of the current segment. The current segment is still accessible from its logical path.

REIN The reinsert of a segment that was no longer accessible from its physical path, but accessible through a logical path.

For more information on the XPCBPCALL, refer to *IMS/ESA Customization Guide*.

XPCBINQA Address of the output of an IMS INQY ENVIRON call. An implied IMS INQY call is done before calling the exit routine. Therefore, the information returned to an application program after an INQY call is available to the exit routine without having to generate the call. This information includes the PSBNAME, RECOVERY TOKEN, PCB LIST, and so forth. You can use this information to augment the data in the exit routine control blocks. See *IMS/ESA Application Programming: DL/I Calls* for more details about the INQY ENVIRON call.

XPCBCKEYL The length of the fully concatenated key. This field is zero if the fully concatenated key is not provided (for example, if the EXIT keyword of the DBD specifies the NOKEY data option).

XPCBCKEYA The address of the fully concatenated key. This field is zero if the fully concatenated key is not provided (for example, if the EXIT keyword of the DBD specifies the NOKEY data option).

XPCBXSDBD Address of the XSDB control block for the changed segment data. This field is zero if the XSDB is not provided (for example, if the EXIT keyword of the DBD specifies the NODATA data option).

XPCBXSDBB Address of the XSDB control block for the *before-image* of a replaced segment. This field is zero if the XSDB is not provided (for example, if the EXIT keyword of the DBD specifies the NODATA data option, or if the IMS change is not a replace).

XPCBXSDBP Pointer to the first XSDB on the descending hierarchic chain. This field is zero if the chain of XSDBs is zero (for example, if the EXIT keyword of the DBD specifies the NOPATH option, or if the changed segment is a root segment).

The XPCBRC, XPCBRSNC, and XPCBEXIWP fields are reserved for RUP use. Your exit routine must not modify them.

XSDB DSECT

The XSDB control block is shown in Figure 43, followed by a detailed description of those fields that are most useful to your exit routine.

```
                                   52+***********************************************************************
                                   53++*                                                                     *
                                   54++*       E X T E N D E D   S E G M E N T   D A T A   -- X S D B      *
                                   55++*                                                                     *
                                   56+***********************************************************************

000000                             58+XSDB      DSECT
000000                             59+XSDBEYE   DS    CL4     "XSDB" EYECATCHER
000004                             60+XSDBVER   DS    CL2     XSDB VERSION INDICATOR
000006                             61+XSDBREL   DS    CL2     XSDB RELEASE INDICATOR
000008                             62+XSDBNXSDB DS    A       NEXT XSDB POINTER
00000C                             63+XSDBDBD   DS    CL8     PHYSICAL DATA BASE NAME
000014                             64+XSDBSEG   DS    CL8     PHYSICAL SEGMENT NAME
00001C                             65+XSDBPHP   DS    CL1     PHYSICAL PATH ACCESSIBILITY
                            000E8  66+XSDBPHPY  EQU   C'Y'    ...SEGM ACCESSIBLE VIA PHYSICAL PATH
                            000D5  67+XSDBPHPN  EQU   C'N'    ...SEGM NOT ACCESSIBLE VIA PH. PATH
00001D                             68+          DS    CL3     RESERVED
000020                             69+XSDBSEGLV DS    H       SEGMENT DATA BASE LEVEL
000022                             70+XSDBKEYL  DS    H       LENGTH OF PHYSICAL KEY
000024                             71+XSDBKEYA  DS    A       ADDRESS OF PHYSICAL KEY
000028                             72+XSDBFIL1  DS    H       RESERVED
00002A                             73+XSDBSEGL  DS    H       LENGTH OF SEGMENT DATA
00002C                             74+XSDBSEGA  DS    A       ADDRESS OF SEGMENT DATA
000030                             75+XSDBFIL2  DS    F       RESERVED
000034                             76+XSDBFIL3  DS    F       RESERVED
000038                             77+XSDBFIL4  DS    F       RESERVED
                            0003C  78+XSDBLEN   EQU   *-XSDB  LENGTH OF XSDB
```

Figure 43. Extended Segment Data Block (XSDB)

XSDB Field Descriptions

The fields of the XSDB that you are likely to need are:

XSDBNXSDB If the XSDB describes path data, this field contains the address of the next XSDB. The XPCB points to the first XSDB, but there is more than one XSDB for path data. They are in hierarchical, top-down sequence. In this case, the XSDBs are chained together, with the last pointer set to zero to indicate the end of the chain.

If the XSDB does not describe path data, this field contains a zero.

XSDBSEG The physical segment name.

XSDBPHP Accessibility through the physical path.

This field describes whether a segment is accessible through its physical path. The field can have the following values:

Y (Yes) the segment is accessible through its physical path.
N (No) the segment is not accessible through its physical path.

This field is set to **Y**, unless you have an IMS *logical* delete rule for logical parent segment types. It can be set to **N** for such logical parents and their physical ancestors. Refer to *IMS/ESA Customization Guide* for more information about this field.

XSDBSEGLV The segment level in the database.

XSDBKEYL The length of the key field for this segment (the length is zero if the segment has no key).

XSDBKEYA The address of the key field for this segment.

XSDBSEGL The length of the physical segment.

XSDBSEGA The address of the physical segment.

The XPCBPCALL, XPCBCALL, and XSDBPHP Fields

If your Propagation exit routine does not need to support logical parent segments and their physical ancestors having a LOGICAL IMS delete rule and involved in a unidirectional IMS logical relationship, then you need to test only the value of the XPCBPCALL field. In this case, the logic of a Propagation exit routine performing a simple mapping can be summarized in the following table:

Figure 44. Exit Routine Action Based on the XPCBPCALL Field Value

XPCBPCALL	Meaning	Exit Routine Action
REPL	A segment is replaced	Propagate with SQL UPDATE statements
ISRT	A segment is inserted	Propagate with an SQL INSERT
DLET	A segment is deleted	Propagate with an SQL DELETE

More complex mapping (for example, mapping similar to generalized mapping case 2) propagates the ISRT of an extension segment with an SQL UPDATE statement.

The logic of your propagation exit routine becomes more complex if it needs to support a logical parent segment or one of its physical ancestors having a LOGICAL IMS delete rule and involved in a unidirectional IMS logical relationship. In this case, you first need to decide how the delete of the logical parent (or its physical ancestors) is propagated. You can do this in two ways:

1. Delete the DB2 target row as soon as the segment gets deleted on its physical path (even if the logical parent segment still has logical children and remains accessible through a logical path).
2. Delete the DB2 target row only when the segment gets both physically and logically deleted.

The sample Propagation exit routine illustrates the logic supporting the first choice. Its logic is summarized in Figure 45. For the various combinations of XPCBPCALL, XPCBCALL, and XSDBPHP field values, the table in the figure describes the action taken by the sample exit routine. When taking the described actions, the exit routine does not need to check if the updated segment is involved in logical relationships. A dash (**-**) in a column of the table below means that a test of that value is not performed in the sample exit routine for the combination of values in that row.

Figure 45. Exit Routine Action Based on the XPCBPCALL, XPCBCALL, and XSDBPHP Field Values

XPCBPCALL	XPCBCALL	XSDBPHP	Meaning	Exit Routine Action
REPL	-	Y	A segment accessible through its physical path is replaced.	Propagate with an SQL UPDATE.
REPL	-	N	A segment not accessible through its physical path is replaced through its logical path.	Ignored by exit routine.
ISRT	-	-	A segment is inserted.	Propagate with an SQL INSERT.
REIN	-	-	A segment previously physically deleted (but still accessible through its logical path) is physically reinserted.	Propagate with an SQL INSERT.
DLET	DLET or CASC	-	A segment is physically deleted (if involved in A logical relationship, it is neither accessible through the logical path nor through the physical path).	propagate with an SQL DELETE.
DLPP	DLET or CASC	-	A segment is physically deleted, but it remains accessible through a logical path.	Propagate with an SQL DELETE.
DLET	DLLP	-	A segment previously physically deleted is now also being logically deleted.	Ignored by exit routine.

Interface for RH-Propagation

The following section describes the interface used for RH-propagation. If your exit routine must not support RH-propagation, then you can skip this section and continue with the section "Exit Routine Processing" on page 182.

The HUP Exit Communication Block (HEC) is the second parameter passed to your Propagation exit routine when the HUP calls your routine. It provides the pointers to the areas received from the DB2 Data Capture (DB2CDC). These areas describe and contain the captured changed data, and are listed below:

QWHC Is the DB2 Instrumentation Facility standard header mapped by DSNDQWHC.

QWHS Is the DB2 Instrumentation Facility correlation data mapped by DSNDQWHS.

CDCDD Contains the Data Capture table description and is mapped by the QW0185 DSECT within DSNDQW02.

CDCDA Contains the Data Capture data row and is also mapped by the QW0185 DSECT within DSNDQW02

For inserts and deletes, there is one data row with the data of the inserted or deleted row. For updates, there is one data row containing the after-image and one data row with the before-image of the updated row.

Your exit routine must not modify the HEC or the data pointed to by this control block.

Figure 46 provides an overview of the interface defined through the HEC.

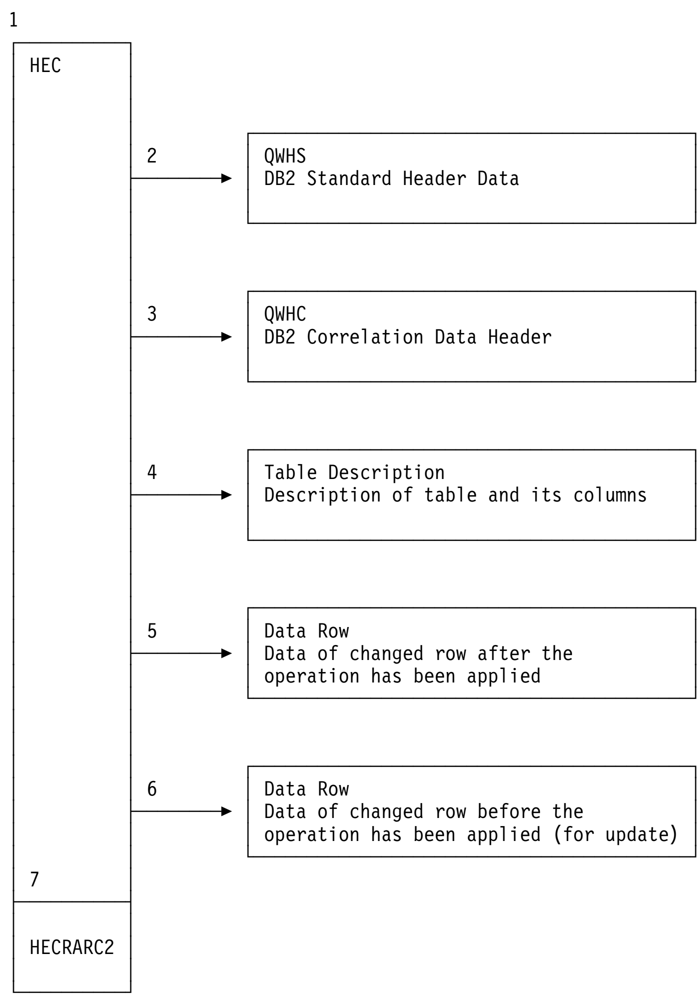

Figure 46. HEC, QWHS, QWHC, Table Description and Data Row Control Block Structures

As shown in the numbered sections of the figure, the interface consists of:

1. One HEC control block that provides various pointers.
2. A pointer to the DB2 Instrumentation Facility standard header data that contains specific DB2 information based on the active trace.
3. A pointer to the DB2 Instrumentation Facility correlation data header containing information about correlation and authorization.

4. A pointer to the Data Capture table description of the changed table and its columns.
5. A pointer to the Data Capture Data (data row) record containing the **after** image of the captured row. For SQL INSERT and DELETE, this is the only data row passed to your exit routine.
6. A pointer to the Data Capture Data (data row) record containing the **before** image of the captured row. This data row is only present for update operations.
7. A field containing the reason code returned by DB2 for the generated IFI call to retrieve the captured data. See *DB2 Messages and Codes* for a description of IFI reason codes.

The HEC Control Block

You can generate the following DSECT in your assembler exit routine by coding the EKYHCHEC macro statement. For HLL exit routines, you can include or copy one of the following members to map the HUP Exit Communication Block:

EKYHCHCC Exit routines written in COBOL
EKYHCHCP Exit routines written in PL/I
EKYHCHCK Exit routines written in C

```
                                          1          EKYHCHEC
                                          2+****************** START OF CONTROL BLOCK SPECIFICATION **************
                                          3+*                                                                    *
                                          4+*        CONTROL BLOCK NAME:                                         *
                                          5+*          EKYHCHEC (HEC)                                            *
                                          6+*                                                                    *
                                          7+*        DESCRIPTIVE NAME:                                           *
                                          8+*          DPROP HUP EXIT COMMUNICATION BLOCK                        *
                                          9+*                =   =    =                                          *
                                         10+*                                                                    *
                                         11+**********************************************************************
                                         12+*                                                                    *
                                         13+*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".      *
                                         14+*                                                                    *
                                         15+*          5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.              *
                                         16+*          ALL RIGHTS RESERVED.                                      *
                                         17+*                                                                    *
                                         18+*          U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                 *
                                         19+*          USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY             *
                                         20+*          GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                  *
                                         21+*                                                                    *
                                         22+*          LICENSED MATERIALS - PROPERTY OF IBM.                     *
                                         23+*                                                                    *
                                         24+**********************************************************************
                                         25+*                                                                    *
                                         26+*        STATUS: V1 R2 M0                                            *
                                         27+*                                                                    *
                                         28+*        FUNCTION:                                                   *
                                         29+*          THIS IS THE CONTROL BLOCK USED TO PASS INFORMATION        *
                                         30+*          GOT BY DPROP FROM THE DB2 CHANGED DATA CAPTURE EXIT       *
                                         31+*          (USING IFI CALLS) TO THE PROPAGATION EXIT ROUTINE         *
                                         32+*          AND / OR THE DB2 CHANGED DATA CAPTURE SUBEXIT ROUTINE.    *
                                         33+*                                                                    *
                                         34+*          THE HEC IS BUILD FOR EACH EXIT CALL NEW AND DOES          *
                                         35+*          CONTAIN DATA TO BE RETAINED BEETWEEN EXIT CALLS.          *
                                         36+*                                                                    *
                                         37+*        MODULE TYPE= MACRO                                          *
                                         38+*           PROCESSOR= ASSEMBLER H                                   *
                                         39+*                                                                    *
                                         40+*        INNER CONTROL BLOCKS: NONE                                  *
                                         41+*                                                                    *
                                         42+*        MACROS USED FROM MACRO LIBRARY: NONE                        *
                                         43+*                                                                    *
                                         44+*        CHANGE ACTIVITY:                                            *
                                         45+*                                                                    *
                                         46+******************** END OF CONTROL BLOCK SPECIFICATION **************

000000                                   48+HEC      DSECT ,                   START OF CONTROL BLOCK

                                         50+*------- EYE CATCHTERS
000000                                   51+HECEYE   DS    0CL8                EYE-CATCHER AREA
000000 C5D2E840                          52+HECEYE1  DC    CL4'EKY '           EYE-CATCHER DPROP
000004 C8C5C340                          53+HECEYE2  DC    CL4'HEC '           EYE-CATCHER CONTROL BLOCK
000008 0000000000000000                  54+HECRESV1 DC    2F'0'               RESERVED

                                         56+*------- POINTERS TO IFI HEADER AREAS
000010 00000000                          57+HECQWHS  DC    A(*-*)              ADDRESS OF THE DB2 IFI
                                         58+*                                  STANDARD HEADER AREA
000014 00000000                          59+HECQWHC  DC    A(*-*)              ADDRESS OF THE DB2 IFI
                                         60+*                                  CORRELATION DATA AREA

                                         62+*------- POINTERS TO CDC DATA AREAS
000018 00000000                          63+HECCDCDD DC    A(*-*)              ADDRESS OF CDC DATA DESCRIPT.
                                         64+*                                  ALWAYS PASSED TO EXIT
00001C 00000000                          65+HECCDCDA DC    A(*-*)              ADDRESS OF CDC DATA ROW
```

Figure 47 (Part 1 of 2). HUP Exit Communication Block

```
                                66+*                                         ALWAYS PASSED TO EXIT.
                                67+*                                         ONLY DATA FOR INSERT/DELETE
                                68+*                                         OR CONTAINS THE AFTER
                                69+*                                         IMAGE FOR UPDATE OPERATIONS
000020 00000000                 70+HECCDCDB DC    A(*-*)                     ADDRESS OF CDC DATA ROW.
                                71+*                                         ZERO FOR INSERT AND DELETE
                                72+*                                         OR BEFORE IMAGE OF ROW FOR
                                73+*                                         UPDATE OPERATIONS

                                75+*------- RETURN CODE FROM IFI CALL
000024 00000000                 76+HECRARC2 DC    F'0'                       IFCRC2 REASON CODE

                                78+*------- DBDNAME/SEGNAME/PCBLABEL AREA (MAPPED BY HECDSLDS BELOW)
000028 00000000                 79+HECDBSLA DC    A(*-*)                     ADDR. OF DBD/SEG/PCBLABEL AREA
00002C 00000000                 80+HECDBSLN DC    F'0'                       NUMBER OF ENTRIES IN THIS AREA

                                82+*------- RESERVED SPACE AND CB SIZE
000030 0000000000000000         83+HECRESV2 DC    4F'0'                      RESERVED
000040                          84+HECEND   DS    0D                         END OF CONTROL BLOCK
                         00040  85+HECLEN   EQU   *-HEC                      LENGTH OF CONTROL BLOCK

                                87+*--------------------------------------------------------------------*
                                88+*       FOR PROPAGATION EXIT ROUTINES ONLY, THE HECDBSLA FIELD      *
                                89+*       POINTS TO AN AREA (FOR DB2 SUBEXIT ROUTINES THIS FIELD IS    *
                                90+*       ZERO). THIS AREA CONTAINS 24 BYTE ENTRIES (IN TOP TO BOTTOM  *
                                91+*       HIERARCHY) WHICH WAS DEFINED TO DPROP FOR THE PR IN PROCESS. *
                                92+*       THE NUMBER OF ENTRIES IN THIS LIST IS CONTAINED IN THE       *
                                93+*       HECDBSLN FIELD.                                              *
                                94+*--------------------------------------------------------------------*
000040                          95+HECDSLDS DS    0D                         ENTRY FOR DBD/SEG/PCBLABEL
000040                          96+HECDBDNM DS    CL8                        - DBD NAME
000048                          97+HECSEGNM DS    CL8                        - SEGMENT NAME
000050                          98+HECPCBNM DS    CL8                        - PCB LABEL NAME
                         00018  99+HECDSLDL EQU   *-HECDSLDS                 LENGTH OF ONE ENTRY
                               100          END
```

Figure 47 (Part 2 of 2). HUP Exit Communication Block

The QWHS and QWHC Control Blocks

The IFI standard header data and IFI correlation data are passed as received from the DB2 Instrumentation Facility.

DSNDQWHS Is the DB2 provided macro which maps the standard header data.
DSNDQWHC Is the DB2 provided macro which maps the correlation data.

Refer to *DB2 Administration Guide* for information about these control blocks.

The Table Description and Data Row Control Blocks

The Data Capture Table Description contains a description of the captured data. It is always present when the HUP calls your Propagation exit routine.

The Data Capture Data (data row) contains a row's data. When the HUP calls your Propagation exit routine, it passes one or two data row areas, depending on the type of SQL operation that caused the data to be captured:

- For INSERT and DELETE, there is only one data row that contains either the inserted or deleted row.
- For UPDATE, there are two data rows, one containing the image of the row before the update, and one after the update operation.

Both data rows have the same format and are described by the same Data Capture table description, which is passed to your exit routine.

The table description and data row are composed of a header common to both, and a data part which is different for each control block type:

- The header part describes the table, using its qualified table name and the time stamp of the table description. For the data row, it also contains the RBAs of log records, the operation code, and the operation code qualifier.
- The data part of the table description contains a description of the columns of the table. The description is similar to the SQLDA.
- The data part of the data row contains the row data, as described in the table description data part.

You can generate the following DSECT (provided by DB2) in your assembler exit routine by coding the DSNDQW02 macro statement. This macro contains the QW0185 DSECT that represents the mapping of the table description and data row control blocks that the DB2 Data Capture uses.

For HLL exit routines, you can include or copy one of the following members to map the table description and data row control blocks:

EKYHCQ2C For exit routines written in COBOL
EKYHCQ2P For exit routines written in PL/I
EKYHCQ2K For exit routines written in C

```
                                       1         DSNDQW02

                                       3+*******************************************************************
                                        4+*  QW00185 IS WRITTEN FOR READS REQUESTS FOR IFCID 185.           *
                                        5+*  FOR IFCID 185, THE PRODUCT SECTION WILL PRECEDE THE DATA       *
                                        6+*  SECTION.  A SINGLE READS REQUEST FOR IFCID 185 MAY RESULT IN   *
                                        7+*  A SERIES OF 185 RECORDS.  ONLY THE FIRST 185 RECORD IN SUCH A  *
                                        8+*  A SERIES WILL CONTAIN A PRODUCT SECTION.  IFCID 185 RECORDS    *
                                        9+*  MAY BE BROKEN AT ANY POINT IN THE DATA.  IT IS UP TO THE       *
                                       10+*  READER OF THE RECORD TO INTERPRET SPANNED IFCID 185 RECORDS.   *
                                       11+*                                                                 *
                                       12+*  QW0185 CONTAINS A HEADER SECTION WHICH IS FOLLOWED BY A DATA   *
                                       13+*  SECTION.  THE DATA PORTION OF QW0185 BEGINS WITH FIELD         *
                                       14+*       - QW0185ID IF QW0185TP=S                                  *
                                       15+*       OR                                                        *
                                       16+*       - QW0185DR IF QW0185TP=D                                  *
                                       17+*******************************************************************
000000                                 18+QW0185    DSECT                   READS IFCID FOR DATA OF DB2CDC
000000                                 19+QW0185LN  DS     F                LENGTH OF TOTAL DB2CDC DATA
000004                                 20+QW0185TP  DS     CL1              TYPE: S = DB2CDC TABLE
                                       21+*                                           DESCRIPTION
                                       22+*                                     D = DB2CDC DATA ROW
000005                                 23+          DS     CL3              RESERVED
000008                                 24+QW0185RC  DS     CL4              REASON CODE DESCRIBING ERROR
                                       25+*                                 FOR THIS DATA PORTION
00000C                                 26+QW0185QT  DS    0CL26             QUALIFIED TABLE NAME
00000C                                 27+QW0185CR  DS     CL8              CREATOR OF TABLE (AUTH ID)
000014                                 28+QW0185TB  DS     CL18             TABLE NAME
000026                                 29+QW0185TS  DS     CL10             TIMESTAMP (INTERNAL FORMAT) OF
                                       30+*                                 TABLE DESCRIPTION FROM CATALOG
000030                                 31+QW0185TL  DS     CL10             TIMESTAMP (INTERNAL FORMAT) OF
                                       32+*                                 LOG BUFFER CI WHEN IT IS EXTERNAL-
                                       33+*                                 IZED OR WHEN THE BUFFER IS
                                       34+*                                 INITIALIZED
00003A                                 35+QW0185UR  DS     CL8              RBA OF THE FIRST LOG RECORD FOR
                                       36+*                                 THIS UNIT OF WORK.
000042                                 37+QW0185LR  DS     CL8              RBA OF LOG RECORD THAT THIS
                                       38+*                                DB2CDC DATA ROW WAS DERIVED FROM
00004A                                 39+QW0185PC  DS     CL2              OPERATION CODE.
                                       40+*                                 USED ONLY IF QW0185TP=D, IN
                                       41+*                                 WHICH CASE, QW0185PC MAY HAVE
                                       42+*                                 ANY OF THE FOLLOWING VALUES:
                                       43+*                                   IN - INSERT
                                       44+*                                   UB - UPDATE BEFORE IMAGE
                                       45+*                                   UA - UPDATE AFTER IMAGE
                                       46+*                                   DE - DELETE
                                       47+*                                   '0000'X IF QW0185TP = 'S'.
00004C                                 48+QW0185RI  DS     CL2              OPERATION CODE QUALIFIER.
                                       49+*                                   '0000'X IF QW0185TP = 'S'.
                                       50+*                                   'RI' IF THE OPERATION IS THE
                                       51+*                                     RESULT OF A REFERENTIAL
                                       52+*                                     CONSTRAINT ENFORCEMENT OF
                                       53+*                                     A DELETE SET NULL OR
                                       54+*                                     CASCADE OPERATION AND
                                       55+*                                     IF QW0185TP = 'D'.
00004E                                 56+          DS     CL6              RESERVED
                            00054      57+QW0185HL  EQU    84               TOTAL LENGTH OF HEADER PORTION
000054                                 58+QW0185DA  DS    0C                BEGIN OF DATA PORTION
                                       59+*******************************************************************
                                       60+*                                                                 *
                                       61+*  IFCID 185 DATA PORTION FOLLOWS                                 *
                                       62+*                                                                 *
                                       63+*  IF QW0185TP = S, THEN                                          *
                                       64+*    THE DATA PORTION CONSISTS OF FOUR VARIABLES FOLLOWED BY AN   *
                                       65+*    ARBITRARY NUMBER OF OCCURRENCES OF THE QW0185VR STRUCTURE.   *
                                       66+*                                                                 *
                                       67+*******************************************************************
```

Figure 48 (Part 1 of 2). Table Description and Data Row Control Blocks

```
000054                    00054    68+         ORG    QW0185DA
000054                             69+QW0185ID  DS     CL8              EYE CATCHER = 'CDCDD   '
00005C                             70+QW0185BC  DS     F                LENGTH OF THE CDCDD =
                                   71+*                                  (QW0185NO * 44) +16
000060                             72+QW0185NO  DS     H                TOTAL NUMBER OF OCCURRENCES OF
                                   73+*                                 QW0185VR
000062                             74+QW0185LD  DS     H                NUMBER OF COLUMNS DESCRIBED BY
                                   75+*                                 OCCURRENCES OF QW0185VR
000064                             76+QW0185VR  DS    0CL44             DESCRIBES A COLUMN IN A
                                   77+*                                 CAPTURED TABLE
000064                             78+QW0185ST  DS     H                TELLS THE DATA TYPE OF THE
                                   79+*                                 COLUMN AND WHETHER IT HAS AN
                                   80+*                                 ASSOCIATED INDICATOR VARIABLE
000066                             81+QW0185LE  DS     H                DEFINES THE EXTERNAL LENGTH OF
                                   82+*                                 A VALUE FROM THE COLUMN
000068                             83+QW0185SD  DS     F                CONTAINS THE CCSID (CODED CHAR
                                   84+*                                 SET ID IN BYTES 3 AND 4.
00006C                             85+QW0185SI  DS     F                OFFSET OF THIS COLUMN INTO THE
                                   86+*                                 DATA ROW
000070                             87+QW0185SN  DS    0C                LENGTH OF NAME AND NAME OF THE
                                   88+*                                 COLUMN
000070                             89+QW0185NL  DS     H                LENGTH OF COLUMN NAME OR LABEL
000072                             90+QW0185CN  DS     CL30             NAME OR LABEL OF COLUMN
                                   91+*
                                   92+*****************************************************************
                                   93+*                                                               *
                                   94+* IF QW0185TP = D, THEN                                         *
                                   95+*    THE DATA PORTION CONSISTS OF                               *
                                   96+*      - THE DATA ROW IF QW0185RC EQUAL 0.                      *
                                   97+*      OR                                                       *
                                   98+*      - AN ERROR MESSAGE IF QW0185RC NOT EQUAL 0.              *
                                   99+*                                                               *
                                  100+*  IN THIS CASE, LENGTH OF DATA PORTION IS QW0185LN - QW0185HL.  *
                                  101+*                                                               *
                                  102+*****************************************************************
000090                    00054   103+         ORG    QW0185DA
000054                            104+QW0185DR  DS    0C               DATA ROW OR ERROR MESSAGE
                                  105         END
```

Figure 48 (Part 2 of 2). Table Description and Data Row Control Blocks

The Table Description and Data Row Header

The following describes the fields of the table description and data row header part in more detail:

QW0185LN Length of total table description or data row (header and data).

QW0185TP Contains the CDC control block type and is:

- **S** For the DB2CDC table description
- **D** For the DB2CDC data row

QW0185RC Reason code describing errors for this table and used only for the data row. If a severe error was detected for this table, the HUP does not call your Propagation exit routine and enforce the rollback of the changes. Therefore, the only reason code that your Propagation exit routine must be able to handle, is the warning code X'00E60A0B'. This code indicates that although the date or time install option was specified as LOCAL, a date or time column value of the row has been returned in ISO format. The DB2 Data Capture never calls date and time exits.

QW0185QT The qualified table name, which is composed by the table creator (QW0185CR) and table name (QW0185TB).

QW0185CR Creator name (authorization ID), which is 8 bytes long and padded with blanks.

QW0185TB Table name, which is 18 bytes long and padded on the right with blanks.

QW0185TS Time stamp (internal format) of table description from the catalog.

QW0185TL Time stamp (internal format) of log record within the log buffer CI. This field is present only in the data row (QW0185TP=D).

QW0185UR RBA of the first log record for this unit of work. This field is present only in the data row (QW0185TP=D).

QW0185LR RBA of log record of this data row. This field is present only in the data row (QW0185TP=D).

QW0185PC Operation code describing the type of row image and the SQL operation that performed the data change. This field is present only in the data row (QW0185TP=D). The possible values of QW0185PC are:

Code	**Description**
IN	Insert
UB	Update before-image
UA	Update after-image
DE	Delete

QW0185RI Operation code qualifier present only in the data row (QW0185TP=D). This field is either blanks, or **RI** if the operation is a result of a referential constraint enforcement of a DELETE SET NULL or CASCADE operation.

The Table Description Data

The table description data portion contains a similar form of an **SQLDA** that describes the table. It is like the standard SQLDA external format, except for the field where you usually specify the address of the data area for a particular column. In the CDC table description this field is already set and contains the **offset to the column** within the data row data section, which is optionally prefixed by a null indicator variable.

The data portion of the table description consists of four variables, followed by an arbitrary number of occurrences of a sequence of five variables collectively called QW0185VR.

QW0185ID An eye catcher for storage dumps containing **CDCDD**.

QW0185BC The length of the table description data portion. It is (QW0185NO * 44) + 16.

QW0185NO Total number of occurrences of QW0185VR.

QW0185LD The number of columns described by occurrences of QW0185VR.

The following five variables are collectively called QW0185VR and occur QW0185NO times in the table description. Each occurrence of QW0185VR describes a column in the captured table.

QW0185ST Tells the data type of the column and whether it has an associated indicator variable. For a description of the type codes, see Figure 49 on page 181.

QW0185LE Defines the external length of a value of the column, as follows:

Data Type	**Content**
Character	Length attribute in bytes
Graphic	Length attribute in **bytes**
Decimal	byte 1 = precision byte 2 = scale
Float	4 (bytes) for single precision 8 (bytes) for double precision
Smallint	2 (bytes)
Integer	4 (bytes)
Date	10 (bytes) or LOCAL value
Time	8 (bytes) or LOCAL value
Time stamp	26 (bytes).

QW0185SD Contains the CCSID (Coded Character Set Identifier) in bytes 3 and 4. It is a two-byte (unsigned) binary number that uniquely identifies an encoding scheme and one or more pairs of character sets and code pages.

QW0185SI Contains a flag byte and the offset of this column into the data row. The flag byte indicates if the column can be nullable or not. If the column value can be NULL, then the column data in the data row is prefixed by an indicator variable (2 bytes). The offset points to the null indicator variable instead of the data for the column; the data immediately follows the indicator and starts at offset + 2. The indicator variable is a two-byte field in the data row containing X'FFFF' (value -1) if the field is null, or X'0000' if the field contains data.

The format of the QW0185SI field is:

Bytes	**Content**
1	Flag byte. If highest bit (bit 0) is on, then the column is prefixed with a null indicator variable, and the real data starts at offset + 2. The remaining bits are reserved.
2-4	Offset into the data, or indicator variable for this column. This offset must be added to the data row data portion address (QW0185DR) to compute the virtual storage address of the column data or indicator variable.

QW0185SN Length of name (QW0185NL) and name of the column (QW0185CN).

QW0185NL Contains the length of the column name.

QW0185CN Contains the name of the column.

The table below lists values of the QW0185ST field of the table description and their meanings. There are two values for each data type. The first value means that the column does not have a null indicator and does not allow nulls; the second means the column *has* a null indicator and allows nulls. For more information about data types, refer to *DB2 SQL Reference*.

Figure 49. Values of QW0185ST and Their Meanings

Values	Data Type
384/385	Date
388/389	Time
392/393	Time stamp
448/449	Variable-length character string
452/453	Fixed-length character string
456/457	Long character string
460/461	Variable-length, optionally null terminated character string (C)
464/465	Variable-length graphic string
468/469	Fixed-length graphic string
472/473	Long graphic string
480/481	Floating point
484/485	Decimal
496/497	Large Integer
500/501	Small Integer

The Data Row Data

The data row data portion starts at label QW0185DR. It contains actual data mapped according to the table description, with DB2-calculated **offsets** into the data for each column.

SQL inserts (IN) and SQL deletes (DE) are passed as one row pointed to by HECCDCDA, a single image that contains **all** the columns in the table.

SQL updates are passed as two rows, an after-image (UA) pointed to by HECCDCDA, and a before-image (UB) pointed to by HECCDCDB. Both images contain **all** the columns of the table.

As applicable, the rules of the external form of a table description dictate how the following data items are handled:

- A string of fields, ordered as they were specified in the external form of a table description of the table, and in standard SQL external format.
- EDITPROCs and FIELDPROCs are called as in standard SQL. The returned data is as decoded by an EDITPROC or any FIELDPROCs that apply, the same as standard SQL.
- DBCS data is supported as in standard SQL.
- VARCHARs are padded to maximum length, but they contain the actual length in the first two bytes of the data.

- Nulls are represented by an indicator variable (two bytes) that precedes the field, but this field is not included in the length.

Exit Routine Processing

Using the information in the control blocks described above (interface control block, XPCB, and XSDCB for HR-propagation, or interface control block, HEC, data description and data row for RH-propagation), you can propagate the changed data segment (pointed to by the XSDB) or DB2 row (pointed to by the data row) in any way you choose. This section describes considerations for developing your Propagation exit routine.

Calling Your Exit Routine

DPROP loads your Propagation exit routine before its first call, and keeps it in virtual storage until the OS/VS task terminates. In MPP regions, this spans multiple MPP executions. Before calling your exit routine, the RUP or HUP reads the Propagation interface control block, checks the propagation status, and traces the changed IMS data or DB2 data.

DPROP uses standard OS/VS conventions when calling your exit routine.

Register 1 Points to the parameter list described above.
Register 13 Contains the address of a register save area.
Register 14 Contains the return address.
Register 15 Contains the entry point address of the exit routine.

Upon entering the exit routine, the register contents must be saved into the caller's save area. If your exit routine calls other routines that use standard MVS linkage conventions, it must also provide a save area of its own. The exit routine must return to its caller using normal OS/VS conventions after restoring the registers. A return code must be provided in the interface control block, not in register 15. Also, like the other exit routines, your Propagation exit routine gains control in AMODE 31, and must return control in AMODE 31.

For HR-propagation, Propagation exit routines can be called multiple times during one IMS call if the call updates more than one segment type, or if multiple PRs exist for one segment type. The number of calls, and the order in which they are made, depends on these conditions and the type of IMS update being made.

- During processing of an updating IMS call, IMS calls the RUP once for each occurrence of a modified segment type. For ISRT and REPL operations, the call sequence is top-down. For DLET operations, the call sequence is usually bottom-up. Refer to *IMS/ESA Application Programming: DL/I Calls* for more information on the call sequence.
- During one call, the RUP needs to process multiple PRs propagating the modified segment occurrence. The RUP processes the PRs sequentially.
  - The RUP calls a Propagation exit routine for each one of the following active PRs belonging to a user mapping case.
    1. If defining PRs with DataRefresher, for each PR identifying the modified segment type in the PROPSEDG keyword. The PROPSEGM keyword is part of the MAPUPARM keyword of the DataRefresher UIM SUBMIT control statement.
    2. If defining PRs in the MVG input tables, for each PR having a DPRISEG row identifying the modified segment type.

For details on defining a PR, see "Telling DPROP About Your Propagation Exit" on page 186.

- The RUP also processes each active PR belonging to a generalized mapping case that identifies the modified segment occurrence as an entity segment or as an extension segment.

For RH-propagation, Propagation exit routines can be called multiple times

- If you have multiple PRs propagating the same table, or
- During the processing of an SQL statement, if the statement updates or deletes more than one row.

The number of calls, and the order in which they are made, depends on the DB2 process sequence of the rows and is unpredictable for DPROP and the Propagation exit routine.

Exit Routine Logic

Your exit routine must supply all the mapping logic, SQL statements, and IMS calls necessary for propagating the changed data to DB2 or IMS. For performance reasons, it is recommended that your exit routine generate static SQL calls. Avoid using functions that have a detrimental effect on the performance of the propagating program (such as performing an OPEN and CLOSE on an MVS file each time the exit routine is called). It is also recommended that the Database Request Modules (DBRMs) of your Propagation exits be package bound. The DB2 plans created for the propagating application programs must then list the packages.

You can also propagate data changes to more than one DB2 table or IMS database. For more information, see "Propagating Data To More Than One DB2 Table" on page 188.

Because the exit routine for synchronous propagation runs in the same environment as the propagating application program, it can generate the same type of IMS calls and SQL statements that the application program can. For LOG-ASYNC and user asynchronous propagation using the TSO Attach or CAF Attach, the exit routines do not execute in an IMS environment, and cannot generate IMS calls. For asynchronous propagation, therefore, create only SQL statements.

If the exit generates SQL statements, then the DBRM of your Propagation exit routine must be included in the DB2 plans of those application programs which synchronously propagate the changed data. For both LOG-ASYNC and user asynchronous propagation, the DBRM must be included in the DB2 plan of the receiver program.

For RH-propagation, your exit probably generates IMS calls. Use the AIB interface described in *IMS/ESA Application Programming: DL/I Calls*, which allows your exit routine to generate calls without the address of the IMS PCBs.

During synchronous propagation, any changes you make to propagated data from within your exit routine are not propagated.

A Propagation exit routine must not perform functions that are not supported by the environment in which it is running. For example, an exit routine running in an MPP region must not write to OS files, and the exit routine must not generate STIMER macros in an IMS environment.

It is recommended that you code and link-edit your program as reentrant. To simplify programming, DPROP provides a work space to your exit routine in the interface control block.

Return Codes and Error Handling Techniques

This section discusses how to return from your exit routine to DPROP, including return codes and a brief description of error handling techniques. For more information on how the RUP and HUP handle error situations, see the appropriate *Administrators Guide* for your propagation mode. First, though, remember that you must return control to the caller in AMODE 31, using the normal MVS conventions described in the previous section.

Return Codes

Below is a list of the return codes you can use when returning from your exit routine, including detailed descriptions of their meanings. The code must be returned in the PICXRETC field of the interface control block.

0 Used for normal returns.

4 Your exit routine must set return code of 4 when it encounters an SQL error code that it considers a propagation failure. If the SQL error code it encounters is considered a normal situation (not a propagation failure), your exit routine must use return code 0.

DPROP assumes that the SQLCA (located in the Propagation interface control block) was used to generate the last SQL statement, and that the last SQL statement was the one that failed. DB2 stores the type of SQL error in the SQLCA. DPROP then reads the SQLCA and, based on which type of error is indicated, proceeds with its usual error handling techniques. DPROP also uses the information in the SQLCA to write an error message describing the details of the error.

8 Your exit must set return code 8 when it encounters an IMS call error that it considers a propagation failure. If the IMS status code it encounters is considered a normal situation (not a propagation failure), your exit routine must use return code 0.

DPROP assumes that the AIB (located in the Propagation Interface Control Block) was used to generate the last IMS call, and that the last IMS call was the one that failed. IMS stores the status code in the failing PCB pointed to by the AIBRSA1 field of the AIB control block. DPROP then reads the AIB and PCB and, based on which type of error is indicated, proceeds with its usual error handling techniques. DPROP also uses the information in the AIB and PCB to write an error message describing the details of the error.

12 Your exit routine must set return code 12 if it encounters a propagation failure error that is not caused by an SQL error or IMS call error, and that DPROP considers as an unavailable resource problem. DPROP then executes its usual error handling techniques for unavailable resources.

16 This return code must be used for propagation failures that are not caused by an SQL error, an IMS call error, or an unavailable resource problem. DPROP again uses its usual error handling techniques for problems other than unavailable resources.

20 Your exit routine must set this return code if there is a severe error for which you want DPROP to ABEND, even if ERROPT=IGNORE is in effect.

Generating ABENDs from an exit routine is not recommended. Doing this results in loss of flexibility of DPROP's error handling techniques.

Error Handling Techniques

When you encounter an error in your exit routine, it is strongly recommended that your exit routine take advantage of DPROP's standard error handling logic. In the interface control block, you can supply a return code in PICXRETC, and an error message in PICXMESG. You must not return an error message in PICXMESG without providing an error return code, because this creates too many console messages.

By supplying DPROP with an error return code and message, you gain many advantages. When an exit returns with an error return code, DPROP traces or snaps the control blocks involved in the interface, and the data. The exits are included in DPROP's standardized error handling techniques; they can differentiate between ERROPT=BACKOUT and ERROPT=IGNORE, and respond based on the type of error encountered; they protect against excessive console messages. DPROP writes your error message using its standard message writing logic: WTO, trace data set (the IMS log, the //EKYLOG data set, or the //EKYTRACE data set), and audit trail.

If the exit routine generates its own messages or ABENDs, DPROP cannot include the exit routine in its standardized error handling, and cannot guard against excessive console messages. Therefore, it is not recommended that your exit routine generate its own messages or ABENDs when an error occurs.

Saving Information Across Calls

You can save information across calls to the exit routine. Save it in the PICSWORK field of the interface control block. If PICSWORK is not large enough, generate a GETMAIN and save the address of the storage in PICSWORK.

Updating Your Propagation Exit Routine

DPROP does not provide any online change logic to replace an existing load module copy of your exit routine with a new version of the load module. If you need to change your exit routine, stop the affected IMS regions and any asynchronous receiver programs before performing the change. A change of the exit routine without stopping the IMS regions and receiver programs causes unpredictable results. For example, some MPP regions use the new version of the exit routine, while other regions use the old version. After the change, you can restart the IMS regions.

Tracing Your Exit Routine

DPROP provides a trace facility that can assist you in detecting errors in your exit routines. DPROP creates trace output when it encounters propagation failures and when the user activates the trace facility.

You can activate the DPROP trace facility by providing a TRACE control statement in the //EKYIN data set of the job step where your exit routine runs. For synchronous propagation, you can also activate tracing by calling the SCU with a TRACE ON control statement.

If you include debug level 2 on the TRACE or TRACE ON statements, the trace output includes, for HR-propagation, the changed IMS segment, and, for RH-propagation, the changed DB2 row. Also, the PICDBLV2 bit of the interface control block is *on* when the exit routine is entered. When this bit is on, It is recommended that your exit routine also trace the propagating SQL statements for HR-propagation, or the propagating IMS calls for RH-propagation. See the appropriate *Administrators Guide* for your propagation mode for details on how to call the DPROP trace module directly from your exit routine.

If you include debug level 4 on the TRACE or TRACE ON statements, each time the exit routine returns to DPROP, the trace output includes:

For HR-propagation:

- The contents of the interface control block
- The XPCB and XSDBs
- The *before replace* image of changed segments
- The path data for the changed segment (if provided by the caller of the RUP)

For RH-propagation:

- The contents of the interface control block
- The HEC, QWHS, and QWHC
- The Data Capture Data Description
- The Data Capture Data area for the before- and after-image of the row.

If you include debug level 8 on the TRACE or TRACE ON statements, the trace output includes a record of each call to and each return from an exit routine.

Other useful debugging aids are the *exit entered* and *exit in control* flags in the interface control block. These flags help you determine if your exit routine is in control at the time of a failure.

Telling DPROP About Your Propagation Exit

This section describes how you can inform DPROP that you want to use a Propagation exit routine. During PR definition, specify which Propagation exit routines must be called when changes are made to specific IMS segment types or DB2 tables. The process you follow depends on whether or not you are creating your PRs using DataRefresher.

Creating a PR Using DataRefresher

Defining a PR that uses a Propagation exit routine is much the same as defining a PR used with the generalized mapping cases. The most significant difference is that, on the MAPUPARM operand of the DataRefresher SUBMIT statement, you must:

- Specify the PRTYPE parameter as PRTYPE=U.
- Give the load module name of the exit routine on the EXITNAME= parameter.
- Identify the list of the segment types propagated by the PR on the PROPSEGM= keyword.

This tells DPROP that you want to use a Propagation exit routine, which exit routine must be called, and which segment types and table are propagated.

For HR-propagation, one segment type is usually propagated by only one PR. However, one segment type can be propagated by multiple PRs, belonging to generalized and user mapping cases. If the segment type is specified on the PROPSEGM= keyword of more than one PR, the RUP calls your exit routine once for each associated PR.

For RH-propagation, one table is usually propagated by only one PR. However, one table can be propagated by multiple PRs, but they must all belong to user mapping cases.

Creating a PR Using the MVG Input Tables

This section discusses how to define a PR for a Propagation exit using the MVG Input Tables. The input into the tables is similar to that used for the generalized mapping cases. When specifying a Propagation exit routine, your PR must have at least one row in the PR table, one row in the DPRISEG (or SEG) table, and one row in the DPRITAB (or TAB) table.

In the PR table, you must specify the PRTYPE column as U. Also, specify the load module name of the exit routine using the EXITNAME column. When you define the PR for your exit routine, leave the MAPCASE column blank. The PROPSUP column is ignored.

For HR-propagation, you must include in the SEG table one row for each segment type that, when changed, is propagated by the Propagation exit routine associated with the PR being defined. When one of these segments is changed, the RUP calls the exit routine to propagate the segment.

Typically, one segment type is propagated by only one PR, and only one PR has a SEG row for that segment type. However, one segment type can be propagated by multiple PRs that belong to generalized and user mapping cases. If the segment type is specified on the SEG row of more than one PR, the RUP calls your exit routine once for each associated PR.

For RH-propagation, include in the TAB table one row for each table that, when changed, is propagated by the Propagation exit routine associated with the PR being defined.

Typically, one table is propagated by only one PR, and only one PR has a TAB row for that table. However, one table can be propagated by multiple PRs, belonging to user mapping cases. If the table is specified on the TAB row of more than one PR, the HUP calls your exit routine once for each associated PR.

The SEGEXIT, SEGEXITL, and SEGEXITF columns of the SEG row do not apply to user mapping cases, and are ignored; but they are copied to the SEG mapping table. Also, DPROP ignores the ROLE column, but still must be set to a value **(P, E, or X)** or blank.

In the TAB table, the columns are the same as those for the generalized mapping cases. Also, DPROP performs the same checks. The only difference is that, for a user mapping, you can specify more than one row in the table. For more information about multiple DB2 tables, see the next section.

You can also use the DPRIFLD (or FLD) table to provide information on the fields to be propagated to DB2. However, DPROP does not use the information in this table, and you are not required to provide it.

Propagating Data To More Than One DB2 Table

Using a Propagation exit routine, you can propagate your changed IMS data to more than one DB2 table. The SQL calls involved are created by you, but you must let DPROP know that more than one table is involved. You can only do this through the MVG input tables. To inform DPROP that you want to use more than one DB2 table, add one row in the MVG TAB table for each DB2 table that receives the data changes.

You can define PRs that propagate to multiple tables if you are defining them with the MVG input tables, but not with DataRefresher. However, with DataRefresher, you can define multiple PRs, each propagating the same data to another target DB2 table.

Propagating Data To More Than One IMS Segment

Using a Propagation exit routine, you can propagate your changed DB2 data to more than one IMS segment. The IMS calls involved are created by you, but you must let DPROP know that more than one database or segment is involved. You can do this using either DataRefresher or the MVG input tables. When using DataRefresher, you must use one DataRefresher SEGMENT statement for each segment to which you want to propagate the PR. If you use the MVG input tables, then add one row in the MVG SEG table for each IMS segment that receives the data changes.

Binding the PR

Use the name of the propagation exit as the member name when binding the PR.

First Sample Propagation Exit Routine

Figure 52 on page 190 shows the first example of a Propagation exit routine for HR-propagation only. This example shows you the basic principles for mapping a data change involving path data, although this is already supported by the generalized mapping case capabilities of DPROP Version 1 Release 2. The purpose of this sample exit is to illustrate typical aspects of the logic that a Propagation user exit needs to provide and how to call the DPROP trace module within such an exit routine.

In this case, the sample exit is mapping fields from an entity segment, and nonkey path data located in the segment's parent, to the target DB2 table.

Because this kind of mapping is supported by the DataRefresher mapping logic, the data extract in this case can be performed by DataRefresher.

Mapping Performed By the Sample Exit Routine

Figure 50 on page 189 illustrates the overview of the propagation done on IMS fields by the sample Propagation exit routine.

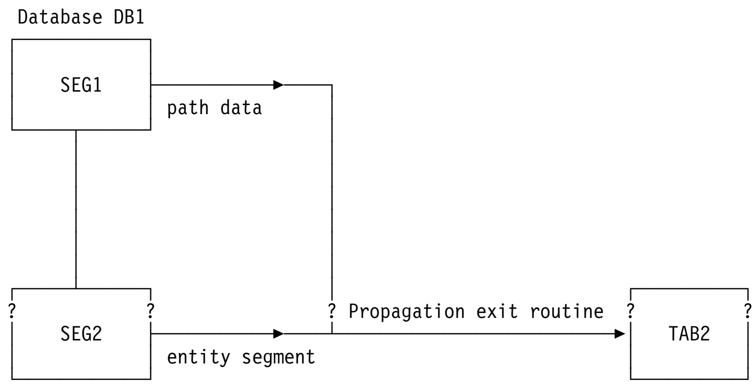

Figure 50. Overview of the Propagation Performed By the Exit Routine

Figure 51 shows the mapping of individual IMS source fields to the DB2 target columns.

Figure 51. Mapping of IMS Source Fields to DB2 Target Columns

Segment Name	Field Name	Key attribute	Column Name	Column Type
SEG1	SEG1KEY1	Key field	TAB2COL1	Part of primary Key
SEG1	SEG1DAT1		TAB2COL6	
SEG1	SEG1DAT2			
SEG1	SEG1DAT3			
SEG2	SEG2KEY1	Key subfield	TAB2COL2	Part of primary Key
SEG2	SEG2KEY2	Key subfield	TAB2COL3	Part of primary Key
SEG2	SEG2DAT1		TAB2COL4	
SEG2	SEG1DAT2		TAB2COL5	

Sample Exit Routine Source Code

The example in Figure 52 on page 190 is intentionally simplified to emphasize the fundamental logic involved. Your Propagation exit routine will likely be more complex to meet your propagation requirements.

The source code below is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYEPR1A. The following source code shows sample module EKYEPR1A after the DB2 precompiler processed it.

Following the source code are definitions related to the sample Propagation exit routine.

```
         MACRO
         SQLSECT &TYPE
         GBLC  &SQLSECT
         AIF ('&TYPE' EQ 'RESTORE').REST
&SQLSECT SETC  '&SYSECT'
         MEXIT
.REST    ANOP
&SQLSECT CSECT
         MEND
********** START OF SPECIFICATIONS ************************************
*    MODULE NAME = EKYEPR1A                                           *
*                                                                     *
*    DESCRIPTIVE NAME = SAMPLE 'PROPAGATION USER EXIT ROUTINE'        *
*                                                                     *
*    STATUS: V1 R2 M0                                                 *
*                                                                     *
*    FUNCTION = EKYEPR1A IS A SAMPLE DPROP                            *
*               'PROPAGATION USER EXIT ROUTINE'.                      *
*                                                                     *
*               EKYEPR1A ILLUSTRATES TYPICAL ASPECTS OF THE LOGIC THAT*
*               A 'PROPAGATION USER EXIT ROUTINES' NEEDS TO PROVIDE.  *
*                                                                     *
*               THIS PARTICULAR SAMPLE EXIT ROUTINE PROPAGATES THE    *
*               CHANGE OF THE DL/I SEGMENT 'SEG2' TO A DB2 TABLE      *
*               'TAB2'.                                               *
*               THE DL/I SOURCE FIELDS FOR THE PROPAGATION ARE        *
*               LOCATED IN:                                           *
*                 - THE FULLY CONCATENATED DL/I KEY OF 'SEG2'         *
*                 - IN THE DATA PORTION OF 'SEG2'                     *
*                 - AND IN THE DATA PORTION OF THE PARENT SEGMENT     *
*                   'SEG1' OF SEG2 (FIELDS IN THE DATA PORTION        *
*                   OF THE PARENT ARE REFERRED TO AS 'PATH DATA').    *
*                                                                     *
*               NOTE THAT MAPPING INVOLVING 'PATH DATA' IS            *
*               SUPPORTED BY THE GENERALIZED MAPPING LOGIC OF DPROP   *
*               V1R2. THEREFORE, IN REAL LIFE, DPROP INSTALLATIONS    *
*               WILL NOT NEED TO PROVIDE A PROPAGATION EXIT ROUTINE   *
*               TO PERFORM THE MAPPING DESCRIBED IN THIS SAMPLE       *
*               EXIT ROUTINE; INSTEAD THEY WILL USE THE GENERALIZED   *
*               MAPPING LOGIC OF DPROP.                               *
*                                                                     *
*              THE FIGURE BELOW PROVIDES AN OVERVIEW OF               *
*              THE DL1-TO-DB2 MAPPING PERFORMED BY THIS SAMPLE EXIT.  *
*                                                                     *
*              *---------------------*   *-------------------------*  *
*              ‘     DL/I WORLD      ‘   ‘      DB2 WORLD          ‘  *
*              *---------------------    *-------------------------*  *
*                                                                     *
*              *---------------------*   *-------------------------*  *
*              ‘  SEGMENT 'SEG1'     ‘   ‘   TABLE 'TAB2'          ‘  *
*              *---------------------*   *-------------------------*  *
*              ‘SEG1KEY1 KEY FLD     ‘--> ‘ TAB2COL1 PRIMARY KEY COL‘  *
*              ‘SEG1DAT1             ‘--> ‘ TAB2COL6                ‘  *
*              ‘SEG1DAT2             ‘   ‘ -                       ‘  *
*              ‘SEG1DAT3             ‘   ‘ -                       ‘  *
*              *---------------------*   ‘                         ‘  *
*                        ‘               ‘                         ‘  *
*                        ‘               ‘                         ‘  *
*                        ‘               ‘                         ‘  *
*                        V               ‘                         ‘  *
*              *---------------------*   ‘                         ‘  *
*              ‘  SEGMENT 'SEG2'     ‘   ‘                         ‘  *
*              *---------------------*   ‘                         ‘  *
*              ‘SEG2KEY1 SUB-KEY FLD‘ -->‘ TAB2COL2 PRIMARY KEY COL‘  *
*              ‘SEG2KEY2 SUB-KEY FLD‘ -->‘ TAB2COL3 PRIMARY KEY COL‘  *
*              ‘SEG2DAT1            ‘ -->‘ TAB2COL4                ‘  *
*              ‘SEG2DAT2            ‘ -->‘ TAB2COL5                ‘  *
*              *---------------------*   *-------------------------*  *
```

Figure 52 (Part 1 of 40). First Sample Propagation Exit Routine (Assembler)

```
*                                                                     *
*            THE PROPAGATION OF A DL/I REPL OF SEG2 RESULTS IN:      *
*                A SQL UPDATE STATEMENT FOR THE THREE COLUMNS         *
*                WHICH ARE NOT PART OF THE PRIMARY DB2 KEY OF TAB2.  *
*                THE 'WHERE CLAUSE' OF THE SQL UPDATE STATEMENT       *
*                PROVIDES THE VALUES FOR THE THREE COLUMNS WHICH      *
*                MAKES UP THE PRIMARY DB2 KEY OF TAB2.                *
*                                                                     *
*            THE PROPAGATION OF A DL/I ISRT OF SEG2 RESULTS IN:      *
*                A SQL INSERT STATEMENT OF A ROW INTO TAB2            *
*                WITH ALL 6 COLUMNS SHOWN IN THE ABOVE TABLE.         *
*                                                                     *
*            THE PROPAGATION OF A DL/I DLET OF SEG2 RESULTS IN:      *
*                A SQL DELETE STATEMENT OF A ROW INTO TAB2.           *
*                THE 'WHERE CLAUSE' OF THE SQL DELETE STATEMENT       *
*                PROVIDES THE VALUES FOR THE THREE COLUMNS WHICH      *
*                MAKES UP THE PRIMARY DB2 KEY OF TAB2.                *
*                                                                     *
*       DISCLAIMERS:                                                  *
*       ------------                                                  *
*       1) THIS SAMPLE EXIT IS BY PURPOSE VERY SIMPLE, IN             *
*          ORDER TO AVOID TO OBSCURE THE MOST ESSENTIAL               *
*          ASPECTS OF THE LOGIC OF A PROPAGATION USER EXIT.           *
*          IN REAL-LIFE, MOST PROPAGATION USER EXITS WILL             *
*          BE MORE COMPLEX THAN THIS SAMPLE BECAUSE THEY              *
*          MIGHT NEED TO PROVIDE LOGIC IN ORDER TO SUPPORT            *
*          FOR EXAMPLE:                                               *
*             - FIELD FORMAT CONVERSION                               *
*             - CONVERSION TO A DB2 'NULL' VALUE                      *
*             - VARIABLE LENGTH SEGMENTS                              *
*             - DL/I FIELDS HAVING A VARIABLE START POSITION          *
*               WITHIN THE SEGMENT.                                   *
*                                                                     *
*       2) NOTE ALSO THAT THIS SAMPLE EXIT DOES ***NOT*** PROPAGATE   *
*          CHANGE OF 'PATH DATA' (I.E THE FIELD SEG1DAT1 OF SEGMENT   *
*          SEG1) TO TAB2.                                             *
*          I.E.: THIS EXIT PROPAGATES THE PATH DATA LOCATED IN SEG1   *
*                ONLY WHEN A SEG2 SEGMENT IS BEING UPDATED. THIS      *
*                EXIT DOES NOT PROPAGATE DATA LOCATED IN SEG1 WHEN    *
*                A SEG1 SEGMENT IS BEING UPDATED.                     *
*                                                                     *
*          IN REAL-LIFE THE USER HAS AT LEAST TWO OPTIONS TO          *
*          PROPAGATE SUCH CHANGES:                                    *
*           A) HE CAN DEFINE FOR THE PROPAGATION OF SUCH CHANGES      *
*              ANOTHER PR AND PROVIDE ANOTHER PROPAGATION USER EXIT   *
*              ROUTINE TO PERFORM THE REQUIRED PROPAGATION.           *
*           OR:                                                       *
*           B) HE CAN PERFORM THE PROPAGATION OF THESE CHANGES WITH   *
*              THE SAME PR AND WITH THE SAME PROPAGATION USER EXIT    *
*              ROUTINE AS THE PROPAGATION OF CHANGES TO SEG2.         *
*              HE SHOULD THEN EXPAND THE LOGIC OF EKYEPR1A IN ORDER   *
*              TO INCLUDE PROPAGATING SQL UPDATE STATEMENTS IN ORDER  *
*              TO PROPAGATE TO TAB2 DL/I REPL OF SEG1 WHICH           *
*              RESULTS IN A CHANGE OF SEG1DAT1.                       *


*                                                                     *
*    NOTES =                                                          *
*       DEPENDENCIES ON DBDGEN SPECIFICATIONS                         *
*       -------------------------------------                         *
*                                                                     *
*              FOR THE PROPAGATION OF REPL AND ISRT OF SEG2,          *
*              EKYEPR1A NEEDS DL/I DATA STORED IN:                    *
*                    - THE FULLY CONCATENATED KEY OF SEG2             *
*                    - THE DATA OF SEGM2                              *
*                    - THE DATA OF THE PARENT SEGM1 ('PATH DATA').    *
```

Figure 52 (Part 2 of 40). First Sample Propagation Exit Routine (Assembler)

```
*                                                                      *
*             FOR THE PROPAGATION OF DLET, EKYEPR1A                    *
*             NEEDS ONLY THOSE DL/I FIELDS WHICH ARE MAPPED            *
*             TO THE COLUMNS OF THE DB2 PRIMARY KEY. ALL THESE         *
*             DL/I FIELDS ARE LOCATED IN                               *
*                   - THE FULLY CONCATENATED KEY OF SEG2.              *
*                                                                      *
*          1) THEREFORE, EXIT= SPECIFICATIONS DURING DBDGEN SHOULD     *
*             SPECIFY:                                                 *
*                       *--------------------------------*             *
*                       'EXIT=((EKYRUP00,KEY,PATH,DATA))'              *
*                       *--------------------------------*             *
*             THESE SPECIFICATIONS ALLOW  TO SATISFY THE               *
*             EKYEPR1A DATA REQUIREMENTS FOR THE PROPAGATION OF        *
*             REPL, ISRT AND DLET OPERATIONS.                          *
*                                                                      *
*          2) ***IF*** THE TARGET DB2 TABLES ARE ***NOT*** INVOLVED    *
*             IN REFERENTIAL INTEGRITY CONSTRAINTS ALLOWING TO USE     *
*             THE DL/I DBDGEN 'NOCASCADE' OPTION, THEN PROPAGATION     *
*             OF DL/I DLET REQUIRES THE DBDGEN OPTION OF               *
*             'CASCADE' ('CASCADE' IS A DBDGEN DEFAULT OPTION).        *
*             THE DL/I DBDGEN CASCADE OPTION:                          *
*                 - MUST SPECIFY (OR DEFAULT TO) THE 'KEY' SUBOPTION   *
*                   (BECAUSE EKYEPR1A NEEDS THE FULLY CONCATENATED     *
*                   KEY OF SEG2 TO PROPAGATE CASCADING DELETES OF      *
*                   SEG2),                                             *
*                 - CAN SPECIFY THE 'NODATA' AND 'NOPATH' OPTIONS      *
*                   (BECAUSE EKYEPR1A NEEDS NEITHER SEG2 DATA NOR      *
*                   PATH DATA TO PROPAGATE DELETES OF SEG2).           *
*             THEREFORE THE CASCADE OPTION IN DBDGEN WILL TYPICALLY    *
*             BE SPECIFIED AS:                                         *
*                       *----------------------------*                 *
*                       '(CASCADE,KEY,NODATA,NOPATH)'                  *
*                       *----------------------------*                 *
*             IT IS ALSO OK TO TAKE THE DEFAULT CASCADE OPTIONS,       *
*             WHICH ARE:                                               *
*                       *----------------------------*                 *
*                       '(CASCADE,KEY,DATA,NOPATH)  '                  *
*                       *----------------------------*                 *
*                                                                      *
*      DEPENDENCIES ON LINKAGE EDITING                                 *
*      -------------------------------                                 *
*          1) EKYEPR1A MUST BE LINK EDITED WITH THE 'RIGHT'            *
*             DB2 LANGUAGE INTERFACE ROUTINE (DB2 HAS DIFFERENT        *
*             LANGUAGE INTERFACE ROUTINES FOR EACH UNIQUE              *
*             ENVIRONMENT:  ONE LANGUAGE INTERFACE ROUTINE FOR         *
*             IMS ENVIRONMENTS, ANOTHER FOR TSO ENVIRONMENTS,          *
*             AND ANOTHER FOR CAF ENVIRONMENTS).                       *
*                                                                      *
*             IF USING EKYEPR1A FOR DPROP ASYNCHRONOUS PROPAGATION     *
*               OR USER ASYNCHRONOUS PROPAGATION USING A CAF ATTACH    *
*                - THE INSTALLATION MUST LINK EKYEPR1A WITH THE        *
*                  DB2 LANGUAGE INTERFACE FOR THE CAF ATTACH.          *
*                                                                      *
*             IF USING EKYEPR1A FOR ASYNCH PROPAGATION IN AN IMS       *
*             ENVIRONMENT:                                             *
*                - THE INSTALLATION MUST LINK EKYEPR1A WITH THE        *
*                  DB2 LANGUAGE INTERFACE FOR THE IMS ATTACH.          *
*                                                                      *
*             IF USING EKYEPR1A FOR ASYNCH PROPAGATION IN              *
*             A TSO ATTACH ENVIRONMENT:                                *
*                - THE INSTALLATION MUST LINK EKYEPR1A WITH THE        *
*                  DB2 LANGUAGE INTERFACE FOR THE TSO ATTACH.          *
*                                                                      *
```

Figure 52 (Part 3 of 40). First Sample Propagation Exit Routine (Assembler)

```
*             IF USING EKYEPR1A FOR SYNCHRONOUS PROPAGATION:        *
*                - THE INSTALLATION MUST LINK EKYEPR1A WITH THE     *
*                  DB2 LANGUAGE INTERFACE FOR THE IMS ATTACH.       *
*                                                                   *
*                                                                   *
*          2) EKYEPR1A MUST ALSO BE LINK EDITED WITH THE DPROP      *
*             TRACE MODULE EKYR410X.                                *
*                                                                   *
*      RESTRICTIONS = NONE                                          *
*      REGISTER CONVENTIONS=                                        *
*                   R13= ADDRESS OF SAVE AREA                       *
*                   R12= MODULE BASE REGISTER                       *
*                   R11= BAS REGISTER TO CALL SUBROUTINE            *
*                   R10= ADDRESS OF XPCB                            *
*                   R9 = ADDRESS OF PIC                             *
*                   R8 = ADDRESS OF XSDB                            *
*                   R7 = ADDRESS OF FULLY CONCATENATED KEY          *
*                   R6 = ADDRESS OF SEGMENT DATA                    *
*                   R5 = ADDRESS OF PATH DATA                       *
*                   R4 = A(SQLDSECT) / A(TRB) / A(TED)              *
*      PATCH LABEL  = - (NONE)                                      *
*                                                                   *
*   MODULE TYPE = PROCEDURE                                         *
*      PROCESSOR = ASSEMBLER                                        *
*      MODULE SIZE = APPROXIMATELY 3200 BYTES                       *
*      ATTRIBUTES = REENTRANT                                       *
*      RMODE      = ANY                                             *
*      AMODE      = 31                                              *
*                                                                   *
*   ENTRY POINT = EKYEPR1A                                          *
*      PURPOSE = SEE FUNCTION                                       *
*      LINKAGE = STANDARD OS/VS ASSEMBLER LINKAGE CONVENTIONS.      *
*                                                                   *
*   INPUT  : R1 = POINTING TO A STANDARD PARAMETER ADDRESS LIST.    *
*                 1ST PARAMETER: ADDRESS OF PIC (PIC IS THE         *
*                                EXIT INTERFACE CONTROL BLOCK)      *
*                 2ND PARAMETER: ADDRESS OF DL/I XPCB               *
*                                                                   *
*   OUTPUT : THE CHANGED DL/I SEGMENT HAS BEEN PROPAGATED           *
*                                                                   *
*   EXIT-NORMAL=                                                    *
*      STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.                 *
*      RETURN CODES = 0                                             *
*                                                                   *
*   EXIT-ERROR=                                                     *
*      STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.                 *
*      RETURN CODE  = 4 : SQL  ERROR                                *
*                     20: SEVERE ERRORS                             *
*                                                                   *
*                                                                   *
*   ABEND-CODE OF EKYEPR1A = NONE                                   *
*      ABEND-REASON CODES  = NONE                                   *
*                                                                   *
*   ERROR MESSAGES ISSUED BY EKYEPR1A                               *
*       EKYEPR0E   : PROPAGATION FAILURE FOR TABLE=XXXXXXXX         *
*                    FAILING SQL STATEMENT=XXXXX SQL ERROR CODE=XXXX *
*       EKYEPR1E   : UNEXPECTED DBD- OR SEGNAME FOR EKYEPR1A        *
*                    DBDNAME=XXXXX SEGNAME=XXXXXX FUNC=XXXX         *
*       EKYEPR2E   : KEY OF SEG2 NOT PROVIDED BY DL/I CAPTURE       *
*                    DBDNAME=XXXXX SEGNAME=XXXXXX FUNC=XXXX         *
*       EKYEPR3E   : DATA OF SEG2 NOT PROVIDED BY DL/I CAPTURE      *
*                    DBDNAME=XXXXX SEGNAME=XXXXXX FUNC=XXXX         *
*       EKYEPR4E   : PATH DATA NOT PROVIDED BY DL/I CAPTURE         *
*                    DBDNAME=XXXXX SEGNAME=XXXXXX FUNC=XXXX         *
*       EKYEPR5E   : UNEXPECTED CALL FUNCTION IN DL/I XPCB          *
*                    DBDNAME=XXXXX SEGNAME=XXXXXX FUNC=XXXX         *
```

Figure 52 (Part 4 of 40). First Sample Propagation Exit Routine (Assembler)

```
*                                                                  *
*                                                                  *
*   EXTERNAL REFERENCES                                            *
*                                                                  *
*      ROUTINES=      = SQL LANGUAGE INTERFACE                     *
*                       EKYR410X : DPROP TRACE MODULE              *
*                                                                  *
*      DATA AREAS     = SEE CONTROL BLOCKS                         *
*                                                                  *
*      CONTROL BLOCKS = PIC   INTERFACE CB FOR PROPAGATION EXIT    *
*                       XPCB  DL/I CAPTURE EXTENDED PCB            *
*                       XSDB  DL/I CAPTURE EXTENDED SEGMENT        *
*                             DESCRIPTION                          *
*                       TRB   TRACE REQUEST BLOCK                  *
*                       TED   TRACE ELEMENT DESCRIPTION            *
*                                                                  *
*   MACROS CODED IN MODULE=                                        *
*          SETTED   - SET INFORMATION INTO A TED                   *
*                                                                  *
*   MACROS USED FROM MACRO LIBRARY=                                *
*          SAVE     - SAVE REGISTERS                               *
*          GETMAIN  - OS/VS GETMAIN                                *
*                                                                  *
*          EKYRCPIC - INTERFACE CB FOR PROPAGATION EXIT            *
*          EKYRCDL1 - DL/I CAPTURE INTERFACE CONTROL BLOCKS        *
*          EKYTRB   - TRACE REQUEST BLOCK                          *
*          EKYTED   - TRACE ELEMENT DESCRIPTOR                     *
*                                                                  *
*                                                                  *
*   TABLES= NONE                                                   *
*                                                                  *
*   INCLUDE CODE FROM LIBRARY= NONE                                *
*                                                                  *
*   CHANGE ACTIVITY=                                               *
*          KMP0046: SUPPORT OF LOGICAL PARENT SEGMENTS HAVING      *
*                   A 'LOGICAL' IMS DELETE RULE AND INVOLVED       *
*                   IN A UNIDIRECTIONAL LOGICAL RELATIONSHIP.      *
*                                                                  *
************* END OF SPECIFICATIONS ********************************
************ LOGIC OF EKYEPR1A ***************************************
*                                                                 *
*                                                                 *
*   MAIN LINE LOGIC:                                              *
*   ================                                              *
*                                                                 *
*   1) MODULE ENTRY LOGIC:                                        *
*   ----------------------                                        *
*         - PROVIDE REGISTER EQUATES                              *
*         - GENERATE A MODULE SAVEID                              *
*         - SAVE REGISTERS AND ESTABLISH MODULE-BASE REGISTER     *
*         - LOAD ADDRESSES OF CALL PARAMETERS                     *
*         - SET 'MODULE ENTERED' AND 'MODULE IN CONTROL' FLAGS    *
*           INTO PIC.                                             *
*         - SET TABLE QUALIFIER  AND TABLE NAME INTO PIC          *
*         - IF FIRST INVOCATION OF THE EXIT:                      *
*             - GETMAIN AN AREA CONTAINING AMONG OTHER            *
*               A MODULE SAVE AREA AND MODULE WORKSPACE.          *
*             - SAVE ADDRESS OF GETMAINED AREA.                   *
*             - CLEAR THE GETMAINED AREA.                         *
*         - CHAIN MODULE SAVE AREA AND SAVE AREA OF CALLER.       *
*                                                                 *
*   2) VERIFY INFORMATION PROVIDED BY DL/I CAPTURE AND/OR DPROP   *
*   --------------------------------------------------------      *
*         - VERIFY THAT THE EXIT IS INVOKED TO PROPAGATE THE      *
*           RIGHT DBD/SEGNAME.                                    *
```

Figure 52 (Part 5 of 40). First Sample Propagation Exit Routine (Assembler)

```
*         - VERIFY THAT DL/I CAPTURE PROVIDES THE                      *
*           FULLY CONCATENATED KEY OF THE SEGMENT                      *
*         - FOR ISRT AND REPL OPERATIONS:                              *
*           VERIFY THAT DL/I CAPTURE PROVIDES:                         *
*               - THE SEGMENT DATA                                     *
*               - PATH DATA.                                           *
*                                                                      *
*   3) BRANCH ACCORDING TO TYPE OF DL1 UPDATE OPERATION.               *
*   ----------------------------------------------------               *
*                                                                      *
*                                                                      *
*   4) FOR A DL/I REPL:                                                *
*   --------------------                                               *
*         - ISSUE A SQL UPDATE STATEMENT FOR A ROW WITH COLUMNS        *
*           ORIGINATING FROM:                                          *
*               - THE DATA PORTION OF SEG2                             *
*               - PATH DATA (I.E FROM THE DATA PORTION OF THE          *
*                 PARENT SEGMENT)                                      *
*           THE 'WHERE CLAUSE' OF THE UPDATE STATEMENT PROVIDES        *
*           THE VALUES OF THE DB2 COLUMNS WHICH MAKES UP THE           *
*           PRIMARY DB2 KEY.                                           *
*                                                                      *
*         - IF THE SQL UPDATE RESULTS IN AN ERROR OR WARNING:          *
*               - B TO SQLERR ('SQL ERROR LOGIC).                      *
*         - IF THE SQL UPDATE IS OK:                                   *
*               - B TO TRACRET ('TRACE AND RETURN TO CALLER')          *
*                                                                      *
*                                                                      *
*   5) FOR A DL/I ISRT:                                                *
*   --------------------                                               *
*         - ISSUE A SQL INSERT STATEMENT TO INSERT A ROW WITH          *
*           COLUMNS ORIGINATING FROM:                                  *
*               - THE FULLY CONCATENATED KEY OF SEG2                   *
*               - THE DATA PORTION OF SEG2                             *
*               - PATH DATA (I.E FROM THE DATA PORTION OF THE          *
*                 PARENT SEGMENT)                                      *
*                                                                      *
*         - IF THE SQL INSERT RESULTS IN AN ERROR OR WARNING:          *
*               - B TO SQLERR ('SQL ERROR LOGIC).                      *
*         - IF THE SQL INSERT IS OK:                                   *
*               - B TO TRACRET ('TRACE AND RETURN TO CALLER')          *
*                                                                      *
*                                                                      *
*   6) FOR A DL/I DLET:                                                *
*   --------------------                                               *
*         - ISSUE A SQL DELETE STATEMENT TO DELETE THE TARGET ROW.     *
*           THE 'WHERE CLAUSE' OF THE DELETE STATEMENT PROVIDES        *
*           THE VALUES OF THE DB2 COLUMNS WHICH MAKES UP THE           *
*           PRIMARY DB2 KEY.                                           *
*                                                                      *
*         - IF THE SQL DELETE RESULTS IN A WARNING OR AN ERROR         *
*           OTHER THAN 'NOT FOUND':                                    *
*               - B TO SQLERR ('SQL ERROR LOGIC).                      *
*         - IF THE SQL DELETE RESULTS IN A 'NOT FOUND' AND THE         *
*           DL/I DELETE WAS NOT A CASCADING DELETE                     *
*               - B TO SQLERR ('SQL ERROR LOGIC).                      *
*         - IF THE SQL DELETE IS OK,                                   *
*           OR IF THE SQL DELETE RESULTS IN A 'NOT FOUND' AND THE      *
*           DL/I DELETE WAS A CASCADING DELETE:                        *
*               - B TO TRACRET ('TRACE AND RETURN TO CALLER')          *
*                                                                      *
*                                                                      *
*   7) RETURN LOGIC                                                    *
*   ----------------                                                   *
*         - IF THE USER REQUESTED A TRACING OF THE PROPAGATING         *
*           SQL STATEMENTS:                                            *
```

Figure 52 (Part 6 of 40). First Sample Propagation Exit Routine (Assembler)

```
                               398 *               - CALL THE TRACE SUBROUTINE IN ORDER TO TRACE      *
                               399 *                 THE PROPAGATING SQL STATEMENT.                   *
                               400 *         - RESTORE REGISTERS OF THE CALLER                        *
                               401 *         - RETURN TO THE CALLER.                                  *
                               402 *                                                                  *
                               403 *                                                                  *
                               404 *   MISCELLANEOUS (ERROR LOGIC AND TRACING)                        *
                               405 *   =======================================                        *
                               406 *                                                                  *
                               407 *   A) SQL ERROR LOGIC                                             *
                               408 *   ------------------                                             *
                               409 *         - SET RETURN CODE 4                                      *
                               410 *         - FORMAT AN ERROR MESSAGE                                *
                               411 *         - CALL THE TRACE SUBROUTINE TO TRACE THE FAILING         *
                               412 *           SQL STATEMENT.                                         *
                               413 *         - RETURN TO THE CALLER.                                  *
                               414 *                                                                  *
                               415 *                                                                  *
                               416 *   B) ERRORS OTHER THAN SQL ERRORS                                *
                               417 *   -------------------------------                                *
                               418 *         - SET RETURN CODE 4                                      *
                               419 *         - FORMAT AN ERROR MESSAGE                                *
                               420 *         - RETURN TO THE CALLER.                                  *
                               421 *                                                                  *
                               422 *                                                                  *
                               423 *   C) TRACE SUBROUTINE:                                           *
                               424 *   ---------------------                                          *
                               425 *         - FILL INFORMATION INTO THE 'TRACE REQUEST BLOCK (TRB)'  *
                               426 *           LOCATED IN THE GETMAINED AREA.                         *
                               427 *         - FOR EACH ITEM/ELEMENT TO BE INCLUDED IN THE            *
                               428 *           TRACE OUTPUT:                                          *
                               429 *           -- CALL A SETTED MACRO TO IDENTIFY THE INFORMATION     *
                               430 *              TO BE INCLUDED IN THE TRACE OUTPUT.                 *
                               431 *         - CALL THE DPROP TRACER.                                 *
                               432 *         - RETURN TO THE CALLER OF THE TRACE SUBROUTINE           *
                               433 *                                                                  *
                               434 ************ END-OF-LOGIC ******************************************
                               436 *******************************************************************
                               437 *******************************************************************
                               438 *******************************************************************
                               439 ****                                                           ****
                               440 ****          MODULE ENTRY LOGIC                               ****
                               441 ****                                                           ****
                               442 *******************************************************************
                               443 *******************************************************************
                               444 *******************************************************************

000000                         446 EKYEPR1A START
                               447 *
                               448 EKYEPR1A AMODE 31             EXIT EXPECTS TO BE CALLED IN AMODE-31
                               449 EKYEPR1A RMODE ANY            EXIT CAN BE LOADED ANYWHERE
                               450 *
                               451 *-------------------------------------------------------------------*
                               452 *       DEFINITION OF REGISTER EQUATES                              *
                               453 *-------------------------------------------------------------------*
                               454 *
                       00000   455 R0       EQU   0
                       00001   456 R1       EQU   1
                       00002   457 R2       EQU   2
                       00003   458 R3       EQU   3
                       00004   459 R4       EQU   4                A(TED)/A(TRB)/A(SQLDSECT)
                       00005   460 R5       EQU   5                A(DATA OF PARENT SEGMENT)
                       00006   461 R6       EQU   6                A(DATA OF CHANGED SEGMENT)
                       00007   462 R7       EQU   7                A(FULLY CONCATENATED KEY)
                       00008   463 R8       EQU   8                A(XSDB)
                       00009   464 R9       EQU   9                A(PIC)
```

Figure 52 (Part 7 of 40). First Sample Propagation Exit Routine (Assembler)

```
                        0000A   465 R10      EQU   10            A(DL/I XPCB)
                        0000B   466 R11      EQU   11            BAS REGISTER TO CALL SUBROUTINES
                        0000C   467 R12      EQU   12            MODULE BASE REGISTER
                        0000D   468 R13      EQU   13            A(SAVEAREA)
                        0000E   469 R14      EQU   14
                        0000F   470 R15      EQU   15

                                472 *-------------------------------------------------------------------*
                                473 *        GENERATE SAVE-ID CONSISTING OF EXIT NAME,                  *
                                474 *        COMPILATION DATE AND COMPILATION TIME.                     *
                                475 *-------------------------------------------------------------------*

                                477          LCLC  &SAVEID
                                478 &SAVEID  SETC  'EKYEPR1A DPR110'.'-'.'&SYSDATE'.'-'.'&SYSTIME'

                                480 *-------------------------------------------------------------------*
                                481 *        SAVE REGISTERS AND ESTABLISH MODULE-BASE REGISTER          *
                                482 *-------------------------------------------------------------------*

                                484          SAVE  (14,12),,&SAVEID SAVE REGISTERS
000000 47F0 F024        00024   485+         B     36(0,15)                        BRANCH AROUND ID
000004 1E                       486+         DC    AL1(30)                         LENGTH OF IDENTIFIER
000005 C5D2E8C5D7D9F1C1         487+         DC    CL8'EKYEPR1A'                   IDENTIFIER
00000D 40C4D7D9F1F1F060         488+         DC    CL8' DPR110-'                   IDENTIFIER
000015 F0F361F2F361F9F3         489+         DC    CL8'03/23/93'                   IDENTIFIER
00001D 60F1F14BF0F2             490+         DC    CL6'-11.02'                     IDENTIFIER
000023 00
000024 90EC D00C        0000C   491+         STM   14,12,12(13)                    SAVE REGISTERS

000028 18CF                     493          LR    R12,R15       R12=ENTRY POINT OF THIS EXIT
                        00000   494          USING EKYEPR1A,R12  ESTABLISH BASE REGISTER

                                496 *-------------------------------------------------------------------*
                                497 *        LOAD ADDRESS OF CALL PARAMETERS                            *
                                498 *-------------------------------------------------------------------*

00002A 989A 1000        00000   500          LM    R9,R10,0(R1)  LOAD ADDRESS OF TWO CALL PARAMETERS
                        00000   501          USING PIC,R9        R9=BASE FOR INTERFACE CONTROL BLOCK
                        00000   502          USING XPCB,R10      R10=BASE FOR DL/I XPCB

                                504 *-------------------------------------------------------------------*
                                505 *        SET IN THE INTERFACE BLOCK THE                             *
                                506 *        'EXIT ENTERED' AND 'EXIT IN CONTROL' FLAGS.                *
                                507 *-------------------------------------------------------------------*

00002E 92E7 909C  0009C         509          MVI   PICENTRD,C'X'  SET 'EXIT ENTERED'
000032 92E7 909D  0009D         510          MVI   PICINCTL,C'X'  SET 'EXIT IN CONTROL'

                                512 *-------------------------------------------------------------------*
                                513 *        SET IN THE INTERFACE BLOCK THE                             *
                                514 *        TABLE NAME QUALIFIER AND THE TABLE NAME                    *
                                515 *-------------------------------------------------------------------*

000036 D207 91D4 CA20 001D4 00A20   517      MVC   PICTABQ,=CL8' '        UNKNOWN QUALIFIER
00003C D211 91DC CB44 001DC 00B44   518      MVC   PICTABN,=CL18'TAB2'    SET TABLE NAME

                                520 *-------------------------------------------------------------------*
                                521 *        IF THIS IS THE FIRST INVOCATION:                           *
                                522 *         - GETMAIN AN AREA CONTAINING                              *
                                523 *             -- OUR SAVE AREA                                      *
                                524 *             -- MODULE WORKSPACE                                   *
                                525 *         - CLEAR THE GETMAINED AREA WITH BINARY ZEROES             *
                                526 *-------------------------------------------------------------------*
```

Figure 52 (Part 8 of 40). First Sample Propagation Exit Routine (Assembler)

```
000042 58B0 9200           00200   528          L     R11,PICSWORK    R11=A(GETMAINED AREA)
000046 12BB                        529          LTR   R11,R11         IS THIS ADDRESS ZERO?
000048 4770 C074           00074   530          BNZ   NOTFIRST        ...NO>>>FIRST TIME PROCESSING DONE

                                   532          GETMAIN RU,LV=GETML,LOC=ANY GETMAIN AN AREA
00004C                             533+         CNOP  0,4
00004C 47F0 C058           00058   534+         B     *+12-4*0-2*0       BRANCH AROUND DATA
000050 00000319                    535+         DC    A(GETML)           LENGTH
000054 00                          536+IHB0002F DC    AL1(0)             RESERVED
000055 00                          537+         DC    AL1(0)             RESERVED
000056 00                          538+         DC    AL1(0)             SUBPOOL
000057 72                          539+         DC    BL1'01110010'                 MODE BYTE         @G860P30
000058 5800 C050           00050   540+         L     0,*-8+2*0          LOAD LENGTH
00005C 58F0 C054           00054   541+         L     15,IHB0002F                       LOAD GETMAIN PARMS
000060 1B11                        542+         SR    1,1                     ZERO RESERVED REG 1
000062 0A78                        543+         SVC   120                     ISSUE GETMAIN SVC

000064 18B1                        545          LR    R11,R1          R11=A(GETMAINED AREA)
000066 50B0 9200           00200   546          ST    R11,PICSWORK    SAVE ADDRESS GETMAINED AREA

00006A 1801                        548          LR    R0,R1           SET UP
00006C 4110 0319           00319   549          LA    R1,GETML        ...FOR A
000070 1BFF                        550          SR    R15,R15         ...ZEROING
000072 0E0E                        551          MVCL  R0,R14          ...MVCL
000074                             552 NOTFIRST DS    0H

                                   554 *---------------------------------------------------------------------*
                                   555 *        CHAIN TOGETHER OUR SAVEAREA AND THE HIGHER-LEVEL SAVEAREA    *
                                   556 *        AND LOAD INTO R13 THE ADDRESS OF OUR SAVEAREA                *
                                   557 *---------------------------------------------------------------------*

000074 50BD 0008           00008   559          ST    R11,8(R13)      CHAIN OUR SAVEAREA INTO HIGHER
000078 50DB 0004           00004   560          ST    R13,4(R11)      CHAIN HIGHER SAVEAREA INTO OUR
00007C 18DB                        561          LR    R13,R11         R13=A(OUR SAVEAREA)
                           00000   562          USING GETM,R13        ESTABLISH BASE REGISTER FOR WORKAREA
                                   564 ********************************************************************
                                   565 ********************************************************************
                                   566 ********************************************************************
                                   567 ****                                                            ****
                                   568 ****          VERIFY THAT:                                      ****
                                   569 ****            - THE EXIT IS INVOKED TO PROPAGATE THE RIGHT    ****
                                   570 ****              DBD-/SEG-NAME.                                ****
                                   571 ****            - THE DBDGEN EXIT= SPECIFICATIONS ARE SUCH, THAT ****
                                   572 ****              DL/I CAPTURE PROVIDES ALL REQUIRED INFORMATION. ****
                                   573 ****                                                            ****
                                   574 ********************************************************************
                                   575 ********************************************************************
                                   576 ********************************************************************

                                   578 *---------------------------------------------------------------------*
                                   579 *        VERIFY, THAT THE EXIT IS CALLED FOR THE PROPAGATION OF      *
                                   580 *        THE CORRECT DBDNAME AND SEGMENT NAME.                       *
                                   581 *---------------------------------------------------------------------*

00007E D507 A014 CA28 00014 00A28  583          CLC   XPCBDBD,=CL8'DB1'   EXPECTED DBDNAME?
000084 4770 C4DA           004DA   584          BNE   INVDBSEG            ...NO>>>THIS IS AN ERROR
000088 D507 A020 CA30 00020 00A30  585          CLC   XPCBSEG,=CL8'SEG2'  EXPECTED SEGMENT-NAME?
00008E 4770 C4DA           004DA   586          BNE   INVDBSEG            ...NO>>>THIS IS AN ERROR
```

Figure 52 (Part 9 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                      588 *--------------------------------------------------------------------*
                                      589 *        THE FULLY CONCATENATED KEY OF THE CHANGED  DL/I SEGMENT IS   *
                                      590 *        REQUIRED TO BUILD THE PRIMARY KEY OF THE TARGET DB2          *
                                      591 *        TABLE AND TO IDENTIFY THE TARGET DB2 ROW.                    *
                                      592 *                                                                     *
                                      593 *        THE EXIT THEREFORE VERIFIES, THAT DL/I CAPTURE PROVIDES      *
                                      594 *        THE FULLY CONCATENATED KEY OF THE CHANGED DL/I SEGMENT       *
                                      595 *--------------------------------------------------------------------*

000092 5870 A04C            0004C     597          L     R7,XPCBCKEYA        R7=A(FULLY CONCATENATED KEY)
000096 1277                           598          LTR   R7,R7               KEY PROVIDED BY DL/I CAPTURE?
000098 4780 C4EE            004EE     599          BZ    KEYMISS             ...NO>>>THIS IS AN ERROR
                            00000     600          USING FCKEY,R7

                                      602 *--------------------------------------------------------------------*
                                      603 *        FOR ISRT AND REPL UPDATES, THE EXIT REQUIRES BOTH THE        *
                                      604 *        DATA FROM THE CHANGED SEGMENT AND FROM ITS PARENT            *
                                      605 *        SEGMENT (WHICH IS THE ROOT).                                 *
                                      606 *                                                                     *
                                      607 *        THE EXIT THEREFORE VERIFIES, THAT DL/I CAPTURE PROVIDES      *
                                      608 *        THE DATA OF THE CHANGED SEGMENT AND THE PATH DATA FROM       *
                                      609 *        ITS PARENT SEGMENT                                           *
                                      610 *--------------------------------------------------------------------*

00009C D503 A02C CAC8 0002C 00AC8     612          CLC   XPCBPCALL,=CL4'ISRT' IS IT AN ISRT?
0000A2 4780 C0BE            000BE     613          BE    DATAREQ             ...YES>>>DATA REQUIRED
0000A6 D503 A02C CACC 0002C 00ACC     614          CLC   XPCBPCALL,=CL4'REPL' IS IT AN REPL?
0000AC 4780 C0BE            000BE     615          BE    DATAREQ             ...YES>>>DATA REQUIRED
0000B0 D503 A02C CAD0 0002C 00AD0     616          CLC   XPCBPCALL,=CL4'REIN' IS IT A RE-INSERT OF A
0000B6 4780 C0BE            000BE     617          BE    DATAREQ             LOGICAL PARENT?
0000BA 47F0 C0E6            000E6     618          B     DATANREQ            ...NO>>>DATA IS NOT REQUIRED

0000BE                                620 DATAREQ  DS    0H
0000BE 5880 A050            00050     621          L     R8,XPCBXSDBD        R8=A(XSDB OF CHANGED SEGMENT)
                            00000     622          USING XSDB,R8             R8=BASE OF DL/I XSDB
0000C2 1288                           623          LTR   R8,R8               DATA PROVIDED BY DL/I CAPTURE?
0000C4 4780 C502            00502     624          BZ    DATAMISS            ...NO>>>THIS IS AN ERROR
0000C8 5860 802C            0002C     625          L     R6,XSDBSEGA         R6=A(CHANGED DATA)
0000CC 1266                           626          LTR   R6,R6               DATA PROVIDED BY DL/I CAPTURE?
0000CE 4780 C502            00502     627          BZ    DATAMISS            ...NO>>>THIS IS AN ERROR
                            00000     628          USING SEG2,R6

0000D2 58F0 A058            00058     630          L     R15,XPCBXSDBP       R15=A(XSDB FOR PATH DATA)
0000D6 12FF                           631          LTR   R15,R15             IS THIS XSDB PROVIDED?
0000D8 4780 C516            00516     632          BZ    PATHMISS            ...NO>>>THIS IS AN ERROR
0000DC 585F 002C            0002C     633          L     R5,XSDBSEGA-XSDB(R15) R5=A(PATH DATA)
0000E0 1255                           634          LTR   R5,R5               DATA PROVIDED BY DL/I CAPTURE?
0000E2 4780 C516            00516     635          BZ    PATHMISS            ...NO>>>THIS IS AN ERROR
                            00000     636          USING SEG1,R5
0000E6                                637 DATANREQ DS    0H
                                      639 ***********************************************************************
                                      640 ***********************************************************************
                                      641 ***********************************************************************
                                      642 ****                                                               ****
                                      643 ****     BRANCH ACCORDING  TO TYPE OF DL/I UPDATE                  ****
                                      644 ****                                                               ****
                                      645 ***********************************************************************
                                      646 ***********************************************************************
                                      647 ***********************************************************************

0000E6 D503 A02C CACC 0002C 00ACC     649          CLC   XPCBPCALL,=CL4'REPL' IS IT AN REPL?
0000EC 4780 C11C            0011C     650          BE    REPL                ...YES>>>B
0000F0 D503 A02C CAC8 0002C 00AC8     651          CLC   XPCBPCALL,=CL4'ISRT' IS IT AN ISRT?
0000F6 4780 C242            00242     652          BE    ISRT                ...YES>>>B
0000FA D503 A02C CAD0 0002C 00AD0     653          CLC   XPCBPCALL,=CL4'REIN' RE-INSERT OF LOGICAL PARENT
000100 4780 C242            00242     654          BE    ISRT
```

Figure 52 (Part 10 of 40). First Sample Propagation Exit Routine (Assembler)

```
000104 D503 A02C CAD4 0002C 00AD4   655         CLC   XPCBPCALL,=CL4'DLET' IS IT A DLET?
00010A 4780 C360            00360   656         BE    DLET                ...YES>>>B
00010E D503 A02C CAD8 0002C 00AD8   657         CLC   XPCBPCALL,=CL4'DLPP' PHYSICAL-DELETE-ONLY OF A
000114 4780 C360            00360   658         BE    DLET                ...LOGICAL PARENT?
000118 47F0 C52A            0052A   659         B     INVCALL             INVALID CALL FUNCTION
                                    661 ***********************************************************************
                                    662 ***********************************************************************
                                    663 ***********************************************************************
                                    664 ****                                                              ****
                                    665 ****     DL/I SEGMENT HAS BEEN REPLACED:                          ****
                                    666 ****          THIS RESULTS IN A PROPAGATING 'SQL UPDATE'          ****
                                    667 ****          OF THE TARGET DB2 ROW.                              ****
                                    668 ****                                                              ****
                                    669 ***********************************************************************
                                    670 ***********************************************************************
                                    671 ***********************************************************************

00011C                              673 REPL     DS    0H
00011C 95E8 801C      0001C         674         CLI   XSDBPHP,XSDBPHPY    RETURN, IF SEGM NOT ACCES-
000120 4770 C464            00464   675         BNE   RETURN              SIBLE VIA PHYSICAL PATH.

000124 D207 D048 CA38 00048 00A38   677         MVC   OPER,=CL8'UPDATE'   IDENTIFY TYPE OF SQL OPERATION

                                    679 *---------------------------------------------------------------------*
                                    680 *        ISSUE A SQL UPDATE STATEMENT TO UPDATE THE TAB2 ROW         *
                                    681 *---------------------------------------------------------------------*

00012A 4140 D058            00058   683         LA    R4,WORKSQL          ESTABLISH ADDRESSABILITY
                            00000   684         USING SQLDSECT,R4         ...OF SQL DSECT

                                    686 ***$$$
                                    687 *        EXEC  SQL  UPDATE                                            C
                                                       TAB2                                                   C
                                                       SET   TAB2COL4 = :SEG2DAT1 ,                           C
                                                             TAB2COL5 = :SEG2DAT2 ,                           C
                                                             TAB2COL6 = :SEG1DAT1                             C
                                                       WHERE TAB2COL1 = :FCK_SEG1KEY1 AND                     C
                                                             TAB2COL2 = :FCK_SEG2KEY1 AND                     C
                                                             TAB2COL3 = :FCK_SEG2KEY2
00012E 47F0 C14E            0014E   688         B     *+32
000132 00288000001E                 689         DC    H'40',X'8000',H'30'
000138 E740404040404040             690         DC    CL8'X       ',XL8'14E73C84030FCAB4',H'1'
00014A 029300EA                     691         DC    H'659,234'
00014E D217 4004 C132 00004 00132   692         MVC   SQLPLLEN(24),*-28
000154 D203 4028 C14A 00028 0014A   693         MVC   SQLSTNUM(4),*-10
00015A 41F0 9310            00310   694         LA    15,SQLCA
00015E 50F0 401C            0001C   695         ST    15,SQLCODEP
000162 41F0 6008            00008   696         LA    15,SEG2DAT1
000166 50F0 4034            00034   697         ST    15,SQLPVARS+8
00016A D201 4030 CB56 00030 00B56   698         MVC   SQLPVARS+4(2),=X'01C4'
000170 D201 4032 CB58 00032 00B58   699         MVC   SQLPVARS+6(2),=H'8'
000176 1FFF                         700         SLR   15,15
000178 50F0 4038            00038   701         ST    15,SQLPVARS+12
00017C 41F0 6010            00010   702         LA    15,SEG2DAT2
000180 50F0 4040            00040   703         ST    15,SQLPVARS+20
000184 D201 403C CB56 0003C 00B56   704         MVC   SQLPVARS+16(2),=X'01C4'
00018A D201 403E CB58 0003E 00B58   705         MVC   SQLPVARS+18(2),=H'8'
000190 1FFF                         706         SLR   15,15
000192 50F0 4044            00044   707         ST    15,SQLPVARS+24
000196 41F0 5005            00005   708         LA    15,SEG1DAT1
00019A 50F0 404C            0004C   709         ST    15,SQLPVARS+32
00019E D201 4048 CB56 00048 00B56   710         MVC   SQLPVARS+28(2),=X'01C4'
0001A4 D201 404A CB5A 0004A 00B5A   711         MVC   SQLPVARS+30(2),=H'7'
0001AA 1FFF                         712         SLR   15,15
0001AC 50F0 4050            00050   713         ST    15,SQLPVARS+36
0001B0 41F0 7000            00000   714         LA    15,FCK_SEG1KEY1
```

Figure 52 (Part 11 of 40). First Sample Propagation Exit Routine (Assembler)

```
0001B4 50F0 4058            00058   715          ST    15,SQLPVARS+44
0001B8 D201 4054 CB56 00054 00B56   716          MVC   SQLPVARS+40(2),=X'01C4'
0001BE D201 4056 CB5C 00056 00B5C   717          MVC   SQLPVARS+42(2),=H'5'
0001C4 1FFF                         718          SLR   15,15
0001C6 50F0 405C            0005C   719          ST    15,SQLPVARS+48
0001CA 41F0 7005            00005   720          LA    15,FCK_SEG2KEY1
0001CE 50F0 4064            00064   721          ST    15,SQLPVARS+56
0001D2 D201 4060 CB56 00060 00B56   722          MVC   SQLPVARS+52(2),=X'01C4'
0001D8 D201 4062 CB5E 00062 00B5E   723          MVC   SQLPVARS+54(2),=H'2'
0001DE 1FFF                         724          SLR   15,15
0001E0 50F0 4068            00068   725          ST    15,SQLPVARS+60
0001E4 41F0 7007            00007   726          LA    15,FCK_SEG2KEY2
0001E8 50F0 4070            00070   727          ST    15,SQLPVARS+68
0001EC D201 406C CB56 0006C 00B56   728          MVC   SQLPVARS+64(2),=X'01C4'
0001F2 D201 406E CB60 0006E 00B60   729          MVC   SQLPVARS+66(2),=H'6'
0001F8 1FFF                         730          SLR   15,15
0001FA 50F0 4074            00074   731          ST    15,SQLPVARS+72
0001FE D203 402C CADC 0002C 00ADC   732          MVC   SQLPVARS(4),=F'76'
000204 41F0 402C            0002C   733          LA    15,SQLPVARS
000208 50F0 4020            00020   734          ST    15,SQLVPARM
00020C D203 4024 CAE0 00024 00AE0   735          MVC   SQLAPARM,=XL4'00000000'
000212 4110 4004            00004   736          LA    1,SQLPLLEN
000216 5010 4000            00000   737          ST    1,SQLPLIST
00021A 9680 4000      00000         738          OI    SQLPLIST,X'80'
00021E 4110 4000            00000   739          LA    1,SQLPLIST
000222 58F0 CAE4            00AE4   740          L     15,=V(DSNHLI)
000226 05EF                         741          BALR  14,15
                                    742 ***$$$

                                    744          DROP  R4


                                    746 *--------------------------------------------------------------------*
                                    747 *        CHECK SQL ERROR CODE AND SQL WARNINGS                       *
                                    748 *--------------------------------------------------------------------*

000228 5820 931C            0031C   750          L     R2,SQLCODE          R2=SQL ERROR CODE
00022C 1222                         751          LTR   R2,R2               IS IT ZERO?
00022E 4770 C23A            0023A   752          BNZ   REPLFAIL            ...NO>>>THIS IS AN ERROR
000232 95E6 9388      00388         753          CLI   SQLWARN,C'W'        A SQL WARNING?
000236 4770 C23E            0023E   754          BNE   REPLOK              ...NO>>>SQL WAS SUCCESSFUL

00023A                              756 REPLFAIL DS    0H
00023A 47F0 C474            00474   757          B     SQLERR              PROPAGATING SQL STMT FAILED

00023E                              759 REPLOK   DS    0H
00023E 47F0 C450            00450   760          B     TRACRET             PROPAGATING SQL STMT WAS OK
                                    762 *********************************************************************
                                    763 *********************************************************************
                                    764 *********************************************************************
                                    765 ****                                                             ****
                                    766 ****     DL/I SEGMENT HAS BEEN INSERTED:                         ****
                                    767 ****         THIS RESULTS IN A PROPAGATING 'SQL INSERT'          ****
                                    768 ****         OF THE TARGET DB2 ROW.                              ****
                                    769 ****                                                             ****
                                    770 *********************************************************************
                                    771 *********************************************************************
                                    772 *********************************************************************

000242                              774 ISRT     DS    0H
000242 D207 D048 CA40 00048 00A40   775          MVC   OPER,=CL8'INSERT'   IDENTIFY TYPE OF SQL OPERATION
```

Figure 52 (Part 12 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                 777 *-----------------------------------------------------------------------*
                                 778 *       ISSUE A SQL INSERT STATEMENT TO INSERT THE ROW                  *
                                 779 *-----------------------------------------------------------------------*

000248 4140 D058           00058   781         LA    R4,WORKSQL          ESTABLISH ADDRESSABILITY
                           00000   782         USING SQLDSECT,R4         ...OF SQL DSECT

                                   784 ***$$$
                                   785 *       EXEC  SQL  INSERT                                            C
                                                     INTO TAB2                                              C
                                                     (    TAB2COL1,                                         C
                                                          TAB2COL2,                                         C
                                                          TAB2COL3,                                         C
                                                          TAB2COL4,                                         C
                                                          TAB2COL5,                                         C
                                                          TAB2COL6 )                                        C
                                                     VALUES                                                 C
                                                     (    FCK_SEG1KEY1,                                     C
                                                          FCK_SEG2KEY1,                                     C
                                                          FCK_SEG2KEY2,                                     C
                                                          SEG2DAT1,                                         C
                                                          SEG2DAT2,                                         C
                                                          SEG1DAT1 )
00024C 47F0 C26C           0026C   786         B     *+32
000250 00288000001E                787         DC    H'40',X'8000',H'30'
000256 E740404040404040            788         DC    CL8'X       ',XL8'14E73C84030FCAB4',H'2'
000268 02C400E8                    789         DC    H'708,232'
00026C D217 4004 C250 00004 00250  790         MVC   SQLPLLEN(24),*-28
000272 D203 4028 C268 00028 00268  791         MVC   SQLSTNUM(4),*-10
000278 41F0 9310           00310   792         LA    15,SQLCA
00027C 50F0 401C           0001C   793         ST    15,SQLCODEP
000280 41F0 7000           00000   794         LA    15,FCK_SEG1KEY1
000284 50F0 4034           00034   795         ST    15,SQLPVARS+8
000288 D201 4030 CB56 00030 00B56  796         MVC   SQLPVARS+4(2),=X'01C4'
00028E D201 4032 CB5C 00032 00B5C  797         MVC   SQLPVARS+6(2),=H'5'
000294 1FFF                        798         SLR   15,15
000296 50F0 4038           00038   799         ST    15,SQLPVARS+12
00029A 41F0 7005           00005   800         LA    15,FCK_SEG2KEY1
00029E 50F0 4040           00040   801         ST    15,SQLPVARS+20
0002A2 D201 403C CB56 0003C 00B56  802         MVC   SQLPVARS+16(2),=X'01C4'
0002A8 D201 403E CB5E 0003E 00B5E  803         MVC   SQLPVARS+18(2),=H'2'
0002AE 1FFF                        804         SLR   15,15
0002B0 50F0 4044           00044   805         ST    15,SQLPVARS+24
0002B4 41F0 7007           00007   806         LA    15,FCK_SEG2KEY2
0002B8 50F0 404C           0004C   807         ST    15,SQLPVARS+32
0002BC D201 4048 CB56 00048 00B56  808         MVC   SQLPVARS+28(2),=X'01C4'
0002C2 D201 404A CB60 0004A 00B60  809         MVC   SQLPVARS+30(2),=H'6'
0002C8 1FFF                        810         SLR   15,15
0002CA 50F0 4050           00050   811         ST    15,SQLPVARS+36
0002CE 41F0 6008           00008   812         LA    15,SEG2DAT1
0002D2 50F0 4058           00058   813         ST    15,SQLPVARS+44
0002D6 D201 4054 CB56 00054 00B56  814         MVC   SQLPVARS+40(2),=X'01C4'
0002DC D201 4056 CB58 00056 00B58  815         MVC   SQLPVARS+42(2),=H'8'
0002E2 1FFF                        816         SLR   15,15
0002E4 50F0 405C           0005C   817         ST    15,SQLPVARS+48
0002E8 41F0 6010           00010   818         LA    15,SEG2DAT2
0002EC 50F0 4064           00064   819         ST    15,SQLPVARS+56
0002F0 D201 4060 CB56 00060 00B56  820         MVC   SQLPVARS+52(2),=X'01C4'
0002F6 D201 4062 CB58 00062 00B58  821         MVC   SQLPVARS+54(2),=H'8'
0002FC 1FFF                        822         SLR   15,15
0002FE 50F0 4068           00068   823         ST    15,SQLPVARS+60
000302 41F0 5005           00005   824         LA    15,SEG1DAT1
000306 50F0 4070           00070   825         ST    15,SQLPVARS+68
00030A D201 406C CB56 0006C 00B56  826         MVC   SQLPVARS+64(2),=X'01C4'
000310 D201 406E CB5A 0006E 00B5A  827         MVC   SQLPVARS+66(2),=H'7'
000316 1FFF                        828         SLR   15,15
```

Figure 52 (Part 13 of 40). First Sample Propagation Exit Routine (Assembler)

```
000318 50F0 4074           00074   829          ST    15,SQLPVARS+72
00031C D203 402C CADC 0002C 00ADC  830          MVC   SQLPVARS(4),=F'76'
000322 41F0 402C           0002C   831          LA    15,SQLPVARS
000326 50F0 4020           00020   832          ST    15,SQLVPARM
00032A D203 4024 CAE0 00024 00AE0  833          MVC   SQLAPARM,=XL4'00000000'
000330 4110 4004           00004   834          LA    1,SQLPLLEN
000334 5010 4000           00000   835          ST    1,SQLPLIST
000338 9680 4000      00000        836          OI    SQLPLIST,X'80'
00033C 4110 4000           00000   837          LA    1,SQLPLIST
000340 58F0 CAE4           00AE4   838          L     15,=V(DSNHLI)
000344 05EF                        839          BALR  14,15
                                   840 ***$$$

                                   842          DROP  R4


                                   844 *-------------------------------------------------------------------*
                                   845 *        CHECK SQL ERROR CODE AND SQL WARNINGS                      *
                                   846 *-------------------------------------------------------------------*

000346 5820 931C           0031C   848          L     R2,SQLCODE          R2=SQL ERROR CODE
00034A 1222                        849          LTR   R2,R2               IS IT ZERO?
00034C 4770 C358           00358   850          BNZ   ISRTFAIL            ...NO>>>THIS IS AN ERROR
000350 95E6 9388      00388        851          CLI   SQLWARN,C'W'        A SQL WARNING?
000354 4770 C35C           0035C   852          BNE   ISRTOK              ...NO>>>SQL WAS SUCCESSFUL

000358                             854 ISRTFAIL DS    0H
000358 47F0 C474           00474   855          B     SQLERR              PROPAGATING SQL STMT FAILED

00035C                             857 ISRTOK   DS    0H
00035C 47F0 C450           00450   858          B     TRACRET             PROPAGATING SQL STMT WAS OK
                                   860 **********************************************************************
                                   861 **********************************************************************
                                   862 **********************************************************************
                                   863 ****                                                             ****
                                   864 ****     DL/I SEGMENT HAS BEEN DELETED:                          ****
                                   865 ****          THIS RESULTS IN A PROPAGATING 'SQL DELETE'         ****
                                   866 ****          OF THE TARGET DB2 ROW.                             ****
                                   867 ****                                                             ****
                                   868 ****          NOTE:                                              ****
                                   869 ****          IF THE 'SQL DELETE' RESULTS IN A 'NOT FOUND'       ****
                                   870 ****          AND IF THE DL/I DELETE WAS A 'CASCADING DELETE':   ****
                                   871 ****          THE SQL 'NOT FOUND' CAN BE A NORMAL SITUATION;     ****
                                   872 ****          THE WILL THEREFORE NOT CONSIDER THIS SITUATION     ****
                                   873 ****          AS AN ERROR.                                       ****
                                   874 ****                                                             ****
                                   875 **********************************************************************
                                   876 **********************************************************************
                                   877 **********************************************************************

000360                             879 DLET     DS    0H
000360 D503 A028 CAE8 00028 00AE8  880          CLC   XPCBCALL,=C'DLLP'   RETURN IF LOGICAL DELETE OF
000366 4780 C464           00464   881          BE    RETURN              A PREVIOUSLY PHYSICALLY          C
                                                                          DELETED SEGMENT

00036A D207 D048 CA48 00048 00A48  883          MVC   OPER,=CL8'DELETE'   IDENTIFY TYPE OF SQL OPERATION

                                   885 *-------------------------------------------------------------------*
                                   886 *        ISSUE A SQL DELETE STATEMENT TO DELETE THE ROW             *
                                   887 *-------------------------------------------------------------------*

000370 4140 D058           00058   889          LA    R4,WORKSQL          ESTABLISH ADDRESSABILITY
                           00000   890          USING SQLDSECT,R4         ...OF SQL DSECT
```

Figure 52 (Part 14 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                  892 ***$$$
                                  893 *        EXEC  SQL  DELETE                                   C
                                                     FROM TAB2                                     C
                                                     WHERE TAB2COL1 = :FCK_SEG1KEY1 AND            C
                                                           TAB2COL2 = :FCK_SEG2KEY1 AND            C
                                                           TAB2COL3 = :FCK_SEG2KEY2
000374 47F0 C394           00394  894          B     *+32
000378 00288000001E               895          DC    H'40',X'8000',H'30'
00037E E740404040404040           896          DC    CL8'X       ',XL8'14E73C84030FCAB4',H'3'
000390 030700E9                   897          DC    H'775,233'
000394 D217 4004 C378 00004 00378 898          MVC   SQLPLLEN(24),*-28
00039A D203 4028 C390 00028 00390 899          MVC   SQLSTNUM(4),*-10
0003A0 41F0 9310           00310  900          LA    15,SQLCA
0003A4 50F0 401C           0001C  901          ST    15,SQLCODEP
0003A8 41F0 7000           00000  902          LA    15,FCK_SEG1KEY1
0003AC 50F0 4034           00034  903          ST    15,SQLPVARS+8
0003B0 D201 4030 CB56 00030 00B56 904          MVC   SQLPVARS+4(2),=X'01C4'
0003B6 D201 4032 CB5C 00032 00B5C 905          MVC   SQLPVARS+6(2),=H'5'
0003BC 1FFF                       906          SLR   15,15
0003BE 50F0 4038           00038  907          ST    15,SQLPVARS+12
0003C2 41F0 7005           00005  908          LA    15,FCK_SEG2KEY1
0003C6 50F0 4040           00040  909          ST    15,SQLPVARS+20
0003CA D201 403C CB56 0003C 00B56 910          MVC   SQLPVARS+16(2),=X'01C4'
0003D0 D201 403E CB5E 0003E 00B5E 911          MVC   SQLPVARS+18(2),=H'2'
0003D6 1FFF                       912          SLR   15,15
0003D8 50F0 4044           00044  913          ST    15,SQLPVARS+24
0003DC 41F0 7007           00007  914          LA    15,FCK_SEG2KEY2
0003E0 50F0 404C           0004C  915          ST    15,SQLPVARS+32
0003E4 D201 4048 CB56 00048 00B56 916          MVC   SQLPVARS+28(2),=X'01C4'
0003EA D201 404A CB60 0004A 00B60 917          MVC   SQLPVARS+30(2),=H'6'
0003F0 1FFF                       918          SLR   15,15
0003F2 50F0 4050           00050  919          ST    15,SQLPVARS+36
0003F6 D203 402C CAEC 0002C 00AEC 920          MVC   SQLPVARS(4),=F'40'
0003FC 41F0 402C           0002C  921          LA    15,SQLPVARS
000400 50F0 4020           00020  922          ST    15,SQLVPARM
000404 D203 4024 CAE0 00024 00AE0 923          MVC   SQLAPARM,=XL4'00000000'
00040A 4110 4004           00004  924          LA    1,SQLPLLEN
00040E 5010 4000           00000  925          ST    1,SQLPLIST
000412 9680 4000      00000       926          OI    SQLPLIST,X'80'
000416 4110 4000           00000  927          LA    1,SQLPLIST
00041A 58F0 CAE4           00AE4  928          L     15,=V(DSNHLI)
00041E 05EF                       929          BALR  14,15
                                  930 ***$$$

                                  932          DROP  R4


                                  934 *-------------------------------------------------------------------*
                                  935 *        CHECK SQL ERROR CODE AND SQL WARNINGS                      *
                                  936 *-------------------------------------------------------------------*

000420 5820 931C           0031C  938          L     R2,SQLCODE          R2=SQL ERROR CODE
000424 1222                       939          LTR   R2,R2               IS IT ZERO?
000426 4780 C440           00440  940          BZ    DLET090             ...YES>>>B
00042A 5920 CAF0           00AF0  941          C     R2,=F'100'          IS IT A 'NOT FOUND'?
00042E 4770 C448           00448  942          BNE   DLETFAIL            ...NO>>>THIS IS AN ERROR
000432 D503 A028 CAF4 00028 00AF4 943          CLC   XPCBCALL,=C'CASC'   IS IT A CASCADING DELETE?
000438 4780 C44C           0044C  944          BE    DLETOK              ...YES>>>THIS IS (PERHAPS) OK
00043C 4770 C448           00448  945          BNE   DLETFAIL            ...NO>>>THIS IS AN ERROR
000440                            946 DLET090  DS    0H
000440 95E6 9388      00388       947          CLI   SQLWARN,C'W'        A SQL WARNING?
000444 4770 C44C           0044C  948          BNE   DLETOK              ...NO>>>SQL WAS SUCCESSFUL
```

Figure 52 (Part 15 of 40). First Sample Propagation Exit Routine (Assembler)

```
000448                              950 DLETFAIL DS    0H
000448 47F0 C474            00474   951          B     SQLERR              PROPAGATING SQL STMT FAILED

00044C                              953 DLETOK   DS    0H
00044C 47F0 C450            00450   954          B     TRACRET             PROPAGATING SQL STMT WAS OK
                                    956 ***********************************************************************
                                    957 ***********************************************************************
                                    958 ***********************************************************************
                                    959 ****                                                               ****
                                    960 ****     RETURN LOGIC:                                             ****
                                    961 ****          - IF USER REQUESTED TRACING: TRACE THE PROPAGATING   ****
                                    962 ****            SQL STATEMENT.                                     ****
                                    963 ****          - RETURN TO CALLER OF EXIT                           ****
                                    964 ****                                                               ****
                                    965 ***********************************************************************
                                    966 ***********************************************************************
                                    967 ***********************************************************************
                                    968 ***********************************************************************

000450                              970 TRACRET  DS    0H
                                    971 *---------------------------------------------------------------------*
                                    972 *        IF USER REQUESTED TRACING:                                   *
                                    973 *           TRACE THE PROPAGATING SQL STATEMENT                       *
                                    974 *---------------------------------------------------------------------*

000450 9102 9012      00012         976          TM    PICDBLEV,PICDBLV2   USER REQUESTING TRACING
000454 4780 C464            00464   977          BZ    RETURN              ...NO>>>B AROUND

000458 4140 D118            00118   979          LA    R4,WORKTRB          R4=A(TRACE REQUEST BLOCK)
                            00000   980          USING TRB,R4
00045C 92E8 4034      00034         981          MVI   TRBSOLI,TRBSOLY     SET 'A SOLICITED TRACE'
                                    982          DROP  R4
000460 4DB0 C56C            0056C   983          BAS   R11,TRACE           TRACE THE SQL STATEMENT
000464                              984 RETURN   DS    0H

                                    986 *---------------------------------------------------------------------*
                                    987 *        RETURN TO CALLER OF THIS EXIT                                *
                                    988 *---------------------------------------------------------------------*

000464 58DD 0004            00004   990          L     R13,4(R13)          R13=A(HIGHER SAVEAREA)
000468 98EC D00C            0000C   991          LM    R14,R12,12(R13)     RELOAD REGISTERS OF CALLER
00046C 9601 D00F      0000F         992          OI    15(R13),X'01'       SET RETURN INDICATION
000470 1BFF                         993          SR    R15,R15             SET ZERO RETURN-CODE
000472 07FE                         994          BR    R14                 RETURN LOGIC
                                    996 ***********************************************************************
                                    997 ***********************************************************************
                                    998 ***********************************************************************
                                    999 ****                                                               ****
                                   1000 ****     SQL ERROR LOGIC:                                          ****
                                   1001 ****          - SET IN THE INTERFACE BLOCK THE RETURN CODE         ****
                                   1002 ****            AND THE ADDRESS OF THE SQL-COMMUNICATION-AREA      ****
                                   1003 ****          - BUILD IN THE INTERFACE CONTROL BLOCK AN            ****
                                   1004 ****            ERROR MESSAGE                                      ****
                                   1005 ****          - CALL THE TRACE SUBROUTINE                          ****
                                   1006 ****          - RETURN TO THE CALLER OF THE EXIT                   ****
                                   1007 ****                                                               ****
                                   1008 ***********************************************************************
                                   1009 ***********************************************************************
                                   1010 ***********************************************************************

000474                             1012 SQLERR   DS    0H
                                   1013 *---------------------------------------------------------------------*
                                   1014 *        SET IN THE PIC THE ERROR CODE AND THE SQL CODE               *
                                   1015 *---------------------------------------------------------------------*
000474 D201 909E CB62 0009E 00B62  1017          MVC   PICXRETC,=H'4'      TELL A 'SQL ERROR'
```

Figure 52 (Part 16 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                   1019 *-------------------------------------------------------------------*
                                   1020 *       PROVIDE AN ERROR MESSAGE CONTAINING:                        *
                                   1021 *             - MESSAGE ID                                          *
                                   1022 *             - TYPE OF SQL UPDATE OPERATION                        *
                                   1023 *             - TABLE NAME                                          *
                                   1024 *-------------------------------------------------------------------*

00047A D207 90A0 CA50 000A0 00A50  1026          MVC   MSGSID,=CL8'EKYEPR0E' SET MESSAGE ID
000480 9240 90A8      000A8        1027          MVI   MSGSBL1,C' '         SET A BLANK
000484 D21D 90A9 CB64 000A9 00B64  1028          MVC   MSGSTXT,=CL30'PROPAGATION FAILURE FOR TABLE='
00048A D211 90C7 CB44 000C7 00B44  1029          MVC   MSGSTABLE,=CL18'TAB2'
000490 D215 90E6 CB82 000E6 00B82  1030          MVC   MSGSTXT2,=CL22'FAILING SQL STATEMENT='
000496 D207 90FC D048 000FC 00048  1031          MVC   MSGSTXTO,OPER
00049C D20F 9104 CA58 00104 00A58  1032          MVC   MSGSTXT3,=CL16' SQL ERROR CODE='

                                   1034 *-------------------------------------------------------------------*
                                   1035 *       TRANSLATE THE SQL ERROR CODE INTO PRINTABLE CHARACTERS      *
                                   1036 *-------------------------------------------------------------------*

0004A2 4E20 D050            00050  1038          CVD   R2,DBLW             CONVERT SQL CODE TO DECIMAL
0004A6 F321 9115 D056 00115 00056  1039          UNPK  MSGSSQLC(3),DBLW+6(2) UNPACK SQL CODE
0004AC 96F0 9117      00117        1040          OI    MSGSSQLC+2,X'F0'    FORCE PRINTABLE CHARACTER
0004B0 9240 9114      00114        1041          MVI   MSGSSQLCS,C' '      PRESET SIGN TO BLANKS
0004B4 1222                        1042          LTR   R2,R2
0004B6 4780 C4CA            004CA  1043          BZ    SQLERR04            B, IF SQLCODE IS ZERO
0004BA 4740 C4C6            004C6  1044          BM    SQLERR02            B, IF SQLCODE IS NEGATIVE
0004BE 924E 9114      00114        1045          MVI   MSGSSQLCS,C'+'      SET '+' SIGN
0004C2 47F0 C4CA            004CA  1046          B     SQLERR04
0004C6                             1047 SQLERR02 DS    0H
0004C6 9260 9114      00114        1048          MVI   MSGSSQLCS,C'-'      SET '-' SIGN
0004CA                             1049 SQLERR04 DS    0H

                                   1051 *-------------------------------------------------------------------*
                                   1052 *       CALL TRACE SUBROUTINE, IN ORDER TO PERFORM A TRACE OF THE   *
                                   1053 *       FAILING SQL STATEMENT.                                      *
                                   1054 *-------------------------------------------------------------------*

0004CA 4140 D118            00118  1056          LA    R4,WORKTRB          R4=A(TRACE REQUEST BLOCK)
                            00000  1057          USING TRB,R4
0004CE 92D5 4034      00034        1058          MVI   TRBSOLI,TRBSOLN     SET 'NOT A SOLICITED TRACE'
                                   1059          DROP  R4
0004D2 4DB0 C56C            0056C  1060          BAS   R11,TRACE           TRACE THE FAILED SQL STATEMENT
0004D6 47F0 C464            00464  1061          B     RETURN              RETURN TO CALLER
                                   1063 *********************************************************************
                                   1064 *********************************************************************
                                   1065 *********************************************************************
                                   1066 ****                                                             ****
                                   1067 ****    ERRORS OTHER THEN SQL ERRORS:                            ****
                                   1068 ****          - BUILD IN THE INTERFACE CONTROL BLOCK AN          ****
                                   1069 ****            ERROR MESSAGE CONTAINING:                        ****
                                   1070 ****               - A 8-BYTE MESSAGE ID                         ****
                                   1071 ****               - A DESCRIPTION OF THE TYPE OF FAILURE        ****
                                   1072 ****               - THE DBDNAME,  THE SEGMENT NAME, AND THE TYPE ****
                                   1073 ****                 OF DL/I UPDATE.                             ****
                                   1074 ****         - SET A RETURN CODE IN THE INTERFACE CONTROL BLOCK  ****
                                   1075 ****         - RETURN TO CALLER OF THE EXIT                      ****
                                   1076 ****                                                             ****
                                   1077 *********************************************************************
                                   1078 *********************************************************************
                                   1079 *********************************************************************

0004DA                             1081 INVDBSEG DS    0H
0004DA D207 90A0 CA68 000A0 00A68  1082          MVC   MSGOID,=CL8'EKYEPR1E'
0004E0 9240 90A8      000A8        1083          MVI   MSGOBL1,C' '
0004E4 D226 90A9 CBC6 000A9 00BC6  1084          MVC   MSGOTXT(39),=C'UNEXPECTED DBD- OR SEGNAME FOR EKYEPR1A'
0004EA 47F0 C53E            0053E  1085          B     ERRCOM
```

Figure 52 (Part 17 of 40). First Sample Propagation Exit Routine (Assembler)

```
0004EE                          1087 KEYMISS  DS    0H
0004EE D207 90A0 CA70 000A0 00A70  1088          MVC   MSGOID,=CL8'EKYEPR2E'
0004F4 9240 90A8      000A8        1089          MVI   MSGOBL1,C' '
0004F8 D227 90A9 CA78 000A9 00A78  1090          MVC   MSGOTXT(40),=C'KEY OF SEG2 NOT PROVIDED BY DL/I CAPTURE'
0004FE 47F0 C53E            0053E  1091          B     ERRCOM

000502                          1093 DATAMISS DS    0H
000502 D207 90A0 CAA0 000A0 00AA0  1094          MVC   MSGOID,=CL8'EKYEPR3E'
000508 9240 90A8      000A8        1095          MVI   MSGOBL1,C' '
00050C D228 90A9 CBED 000A9 00BED  1096          MVC   MSGOTXT(41),=C'DATA OF SEG2 NOT PROVIDED BY DL/I CAPTUREC
                                                       '
000512 47F0 C53E            0053E  1097          B     ERRCOM

000516                          1099 PATHMISS DS    0H
000516 D207 90A0 CAA8 000A0 00AA8  1100          MVC   MSGOID,=CL8'EKYEPR4E'
00051C 9240 90A8      000A8        1101          MVI   MSGOBL1,C' '
000520 D225 90A9 CB98 000A9 00B98  1102          MVC   MSGOTXT(38),=C'PATH DATA NOT PROVIDED BY DL/I CAPTURE'
000526 47F0 C53E            0053E  1103          B     ERRCOM

00052A                          1105 INVCALL  DS    0H
00052A D207 90A0 CAB0 000A0 00AB0  1106          MVC   MSGOID,=CL8'EKYEPR5E'
000530 9240 90A8      000A8        1107          MVI   MSGOBL1,C' '
000534 D224 90A9 CC16 000A9 00C16  1108          MVC   MSGOTXT(37),=C'UNEXPECTED CALL FUNCTION IN DL/I XPCB'
00053A 47F0 C53E            0053E  1109          B     ERRCOM


00053E                          1111 ERRCOM   DS    0H
00053E D207 90E6 CAB8 000E6 00AB8  1112          MVC   MSGOTXT2,=CL08'DBDNAME='
000544 D207 90EE A014 000EE 00014  1113          MVC   MSGODBD,XPCBDBD
00054A D208 90F6 CC3B 000F6 00C3B  1114          MVC   MSGOTXT3,=CL09' SEGNAME='
000550 D207 90FF A020 000FF 00020  1115          MVC   MSGOSEG,XPCBSEG
000556 D205 9107 CBBE 00107 00BBE  1116          MVC   MSGOTXT4,=CL06' FUNC='
00055C D203 910D A02C 0010D 0002C  1117          MVC   MSGOFUNC,XPCBPCALL

000562 D201 909E CBC4 0009E 00BC4  1119          MVC   PICXRETC,=H'20'     SET 'SHOULD NOT OCCUR' RC
000568 47F0 C464            00464  1120          B     RETURN
                                   1122 ***********************************************************************
                                   1123 ***********************************************************************
                                   1124 ***********************************************************************
                                   1125 ****                                                               ****
                                   1126 ****     TRACING OF PROPAGATING SQL STATEMENT.                     ****
                                   1127 ****                                                               ****
                                   1128 ****     TRACING OF THE PROPAGATING SQL STATEMENT IS ASSISTED      ****
                                   1129 ****     THROUGH USAGE OF THE SAMPLE 'SETTED' MACRO WHICH IS       ****
                                   1130 ****     PROVIDED AND DESCRIBED BELOW.                             ****
                                   1131 ****                                                               ****
                                   1132 ****     FOR EACH ITEM TO BE INCLUDED IN THE TRACE, THIS SAMPLE    ****
                                   1133 ****     EXIT INVOKES THE SETTED SAMPLE MACRO, WHICH IDENTIFIES    ****
                                   1134 ****     TO THE DPROP TRACER THE INFORMATION TO BE INCLUDED IN     ****
                                   1135 ****     THE TRACE.                                                ****
                                   1136 ****                                                               ****
                                   1137 ***********************************************************************
                                   1138 ***********************************************************************
                                   1139 ***********************************************************************



                                   1141 *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
                                   1142 *        SAMPLE SETTED MACRO                                          *
                                   1143 *                                                                     *
                                   1144 *        SETTED IS A SAMPLE MACRO USED TO SUPPORT/EASE CALLS TO       *
                                   1145 *        THE DPROP TRACER.                                            *
                                   1146 *                                                                     *
                                   1147 *        SETTED IS CALLED ONCE FOR EACH ELEMENT/ITEM TO BE            *
                                   1148 *        INCLUDED IN THE TRACE (I.E SETTED IS CALLED ONCE FOR EACH    *
                                   1149 *        'TRACE ELEMENT DESCRIPTOR (TED)').                           *
```

Figure 52 (Part 18 of 40). First Sample Propagation Exit Routine (Assembler)

```
*                                                               *
*        SETTED PERFORMS THE FOLLOWING:                         *
*           - IT DESCRIBES IN THE TED THE ELEMENT TO BE INCLUDED *
*             IN THE TRACE.                                     *
*           - IT STORES THE ADDRESS OF THE TED INTO THE         *
*             CALL PARAMETER LIST USED TO INVOKE THE DPROP TRACER. *
*        TO BE INCLUDED IN THE TRACE OUTPUT.                    *
*                                                               *
*        SETTED IS INVOKED IN ONE OF THE THREE FOLLOWING WAYS:  *
*         1) FOR A 'HEADER-TED':                                *
*            SETTED NBR=...,TYPE=HEADER,TXT=......              *
*         2) FOR A 'SUB-HEADER TED':                            *
*            SETTED NBR=...,TYPE=SUBH,TXT=....                  *
*         3) FOR A 'DATA-TED':                                  *
*            SETTED NBR=...,TYPE=DATA,TXT=....,DATA=.....       *
*                                                               *
*        THE NBR= KEYWORD OPERAND IS USED TO IDENTIFY THE       *
*        RELATIVE NUMBER OF THE TED.                            *
*                                                               *
*        THE TXT= KEYWORD OPERAND IS USED TO PROVIDE THE NAME   *
*        OF AN ASSEMBLER FIELD CONTAINING THE DESCRIPTIVE TEXT  *
*        ASSOCIATED WITH THE TED.                               *
*                                                               *
*        THE DATA= KEYWORD OPERAND IS USED TO PROVIDE THE NAME  *
*        OF AN ASSEMBLER FIELD CONTAINING THE DATA TO BE INCLUDED *
*        IN THE TRACE.                                          *
*                                                               *
*        NOTE THAT 'SETTED' IS ***NOT*** A GENERAL PURPOSE MACRO *
*        AND CAN ***NOT*** BE USED 'AS IS' IN USER-PROGRAMMED   *
*        EXIT ROUTINES. INSTEAD OF USING THE SETTED MACRO 'AS IS' *
*        IN OTHER EXIT ROUTINES, PROGRAMMERS OF THE CUSTOMER MAY *
*        USE 'SETTED' AS A MODEL, WHICH CAN HELP THEM DEVELOP   *
*        THEIR OWN MACRO WHICH IS ADAPTED TO THEIR REQUIREMENTS. *
*                                                               *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**

*-*-*-*- START OF SAMPLE 'SETTED' MACRO *-*-*-*-*-*-*-*-*-*-*-*-*-*-**
         MACRO
&LABEL   SETTED &NBR=,&TYPE=,&TXT=,&DATA=
.*
.***           GET ADDRESS OF TED AND STORE ITS ADDRESS INTO
.***           THE TRACE PARAMETER LIST
.*
         AIF   (T'&NBR EQ 'N').NBROK  CHECK THAT NBR= IS NUMERIC
         MNOTE 8,'VALUE OF NBR= KEYWORD OPERAND MUST BE NUMERIC'
         MEXIT
.NBROK   ANOP
&LABEL   LA    R4,WORKTED+((&NBR-1)*TEDLEN) R4=A(TED)
         USING TED,R4
         ST    R4,TRAPARML+(4*&NBR)         SET A(TED) INTO PARMLIST
.*
.***           SET EYE CATCHER INTO TED
.*
         MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
.*
.***           VALIDATE THE VALUE OF THE TYPE= KEYWORD OPERAND
.***           AND SET TYPE OF TED.
.*
         AIF   ('&TYPE' EQ 'HEADER').HDR
         AIF   ('&TYPE' EQ 'SUBH').SUBH
         AIF   ('&TYPE' EQ 'DATA').DATA
         MNOTE 8,'INVALID OR MISSING VALUE FOR TYPE= OPERAND'
         MEXIT
.HDR     MVI   TEDTYPE,TEDTYPH         SET 'THIS IS A HEADER-TED'
```

Figure 52 (Part 19 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                             AGO   .TYPCOM
                                        .SUBH    MVI   TEDTYPE,TEDTYPS        SET 'THIS IS A SUBHEADER-TED'
                                                 AGO   .TYPCOM
                                        .DATA    MVI   TEDTYPE,TEDTYPD        SET 'THIS IS A DATA-TED'
                                                 AGO   .TYPCOM
                                        .TYPCOM  ANOP
                                        .*
                                        .***           CHECK THAT TXT= KEYWORD OPERAND HAS BEEN PROVIDED
                                        .***           AND SET ADDRESS AND LENGTH OF TEXT INTO TED
                                        .*
                                                 AIF   (T'&TXT NE 'O').TXTOK   B, IF TXT= NOT OMITTED
                                                 MNOTE 8,'TXT= KEYWORD OPERAND VALUE IS MISSING'
                                                 MEXIT
                                        .TXTOK   ANOP
                                                 LA    R15,&TXT               R15=A(TEXT)
                                                 ST    R15,TEDTXTA            STORE A(TEXT) INTO TED
                                                 MVC   TEDTXTL,=A(L'&TXT)     SET LENGTH OF TEXT STRING
                                        .*
                                        .***           IF TYPE=DATA:
                                        .***           CHECK THAT TXT= KEYWORD OPERAND HAS BEEN PROVIDED
                                        .***           AND SET ADDRESS AND LENGTH OF DATA INTO TED
                                        .*
                                                 AIF   ('&TYPE' NE 'DATA').NOTDATA
                                                 AIF   (T'&DATA NE 'O').DATAOK B, IF DATA= NOT OMITTED
                                                 MNOTE 8,'DATA= KEYWORD OPERAND VALUE IS MISSING'
                                                 MEXIT
                                        .DATAOK  ANOP
                                                 LA    R15,&DATA              R15=A(DATA)
                                                 ST    R15,TEDMA              STORE A(DATA) INTO TED
                                                 MVC   TEDALEN,=A(L'&DATA)    STORE LENGTH OF DATA
                                                 MVI   TEDALIGN,TEDALIGL      REQUEST 'LEFT ALIGNMENT'
                                        .NOTDATA ANOP
                                                 MEND
                                        *-*-*-*- END OF SAMPLE 'SETTED' MACRO *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-**
                                        **********************************************************************
                                        *        TRACE SUBROUTINE                                            *
                                        **********************************************************************

                                        *--------------------------------------------------------------------*
                                        *        LOGIC COMMON FOR THE TRACING OF INSERT/UPDATE/DELETE        *
                                        *        SQL STATEMENTS:                                             *
                                        *             - PROVIDE INFORMATION IN THE TRACE PARAMETER BLOCK     *
                                        *             - PROVIDE INFORMATION IN A 'HEADER TED'                *
                                        *             - PROVIDE A TED FOR THE SQL CODE                       *
                                        *--------------------------------------------------------------------*

00056C                                  TRACE    DS    0H
                                        *
                                        ***            PROVIDE INFORMATION IN THE TRACE PARAMETER BLOCK (TRB)
                                        *
00056C 4140 D118            00118                LA    R4,WORKTRB
                            00000                USING TRB,R4
000570 5040 D0E8            000E8                ST    R4,TRATRB          STORE A (TRB) INTO PARMLIST
000574 D203 4000 CAF8 00000 00AF8                MVC   TRBEYE,=CL4'TRB '  SET EYE CATCHER
00057A D203 4004 9014 00004 00014                MVC   TRBPTD,PICPTD      COPY A(PTD) TO TRB
000580 D207 4008 CA20 00008 00A20                MVC   TRBTABQ,=CL8' '    SET TABLE NAME QUALIFIER TO BLANKS
000586 D211 4010 CB44 00010 00B44                MVC   TRBTABN,=CL18'TAB2' SET TABLE NAME
                                        *
                                        ***            PROVIDE 1ST TED (HEADER WITH NAME OF PROPAGATED TABLE)
                                        *
00058C D207 D2C8 CAC0 002C8 00AC0                MVC   TRHP0,=CL8'EKYEPR1A' SET MODULE NAME CREATING THE SNAP
000592 D210 D2D0 CC44 002D0 00C44                MVC   TRHP1,=C' PROPAGATING SQL-'
000598 D207 D2E1 D048 002E1 00048                MVC   TRHP2,OPER         SET TYPE OF SQL STATEMENT
00059E D20A D2E9 CC55 002E9 00C55                MVC   TRHP3,=C' FOR TABLE='
0005A4 D211 D2F4 CB44 002F4 00B44                MVC   TRHP4,=CL18'TAB2'  SET TABLE NAME
```

Figure 52 (Part 20 of 40). First Sample Propagation Exit Routine (Assembler)

```
                             1281         SETTED NBR=1,TYPE=HEADER,TXT=TRHEADER
0005AA 4140 D160       00160 1282+        LA    R4,WORKTED+((1-1)*TEDLEN)    R4=A(TED)
                       00000 1283+        USING TED,R4
0005AE 5040 D0EC       000EC 1284+        ST    R4,TRAPARML+(4*1)            SET A(TED) INTO PARMLIST
0005B2 D203 4000 CAFC 00000 00AFC 1285+   MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
0005B8 92C8 4004      00004  1286+        MVI   TEDTYPE,TEDTYPH         SET 'THIS IS A HEADER-TED'
0005BC 41F0 D2C8       002C8 1287+        LA    R15,TRHEADER            R15=A(TEXT)
0005C0 50F0 4008       00008 1288+        ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0005C4 D203 400C CB00 0000C 00B00 1289+   MVC   TEDTXTL,=A(L'TRHEADER)  SET LENGTH OF TEXT STRING
                             1290 *
                             1291 ***           PROVIDE 2ND TED (SQL ERROR CODE)
                             1292 *
0005CA D212 D306 CC60 00306 00C60 1293    MVC   TXTSQLC,=C'SQL ERROR CODE=-NNN' MOVE TEXT
0005D0 5820 931C       0031C 1294         L     R2,SQLCODE          R2=SQL CODE
0005D4 4E20 D050       00050 1295         CVD   R2,DBLW             CONVERT SQL CODE TO DECIMAL
0005D8 F321 D316 D056 00316 00056 1296    UNPK  TXTSQLCC(3),DBLW+6(2) UNPACK SQL CODE
0005DE 96F0 D318      00318  1297         OI    TXTSQLCC+2,X'F0'    FORCE PRINTABLE CHARACTER
0005E2 9240 D315      00315  1298         MVI   TXTSQLCS,C' '       PRESET SIGN TO BLANKS
0005E6 1222                  1299         LTR   R2,R2
0005E8 4780 C5FC       005FC 1300         BZ    TRACE019            B, IF SQLCODE IS ZERO
0005EC 4740 C5F8       005F8 1301         BM    TRACE012            B, IF SQLCODE IS NEGATIVE
0005F0 924E D315      00315  1302         MVI   TXTSQLCS,C'+'       SET '+' SIGN
0005F4 47F0 C5FC       005FC 1303         B     TRACE019
0005F8                       1304 TRACE012 DS   0H
0005F8 9260 D315      00315  1305         MVI   TXTSQLCS,C'-'       SET '-' SIGN
0005FC                       1306 TRACE019 DS   0H

                             1308         SETTED NBR=2,TYPE=SUBH,TXT=TXTSQLC
0005FC 4140 D184       00184 1309+        LA    R4,WORKTED+((2-1)*TEDLEN)    R4=A(TED)
                       00000 1310+        USING TED,R4
000600 5040 D0F0       000F0 1311+        ST    R4,TRAPARML+(4*2)            SET A(TED) INTO PARMLIST
000604 D203 4000 CAFC 00000 00AFC 1312+   MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
00060A 92E2 4004      00004  1313+        MVI   TEDTYPE,TEDTYPS         SET 'THIS IS A SUBHEADER-TED'
00060E 41F0 D306       00306 1314+        LA    R15,TXTSQLC             R15=A(TEXT)
000612 50F0 4008       00008 1315+        ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
000616 D203 400C CB04 0000C 00B04 1316+   MVC   TEDTXTL,=A(L'TXTSQLC)   SET LENGTH OF TEXT STRING


                             1318 *-----------------------------------------------------------------*
                             1319 *       BRANCH ACCORDING TO TYPE OF SQL UPDATE STATEMENT          *
                             1320 *-----------------------------------------------------------------*

00061C D507 D048 CA38 00048 00A38 1322    CLC   OPER,=CL8'UPDATE'   A SQL UPDATE STATEMENT?
000622 4780 C634       00634 1323         BE    TRACEU
000626 D507 D048 CA40 00048 00A40 1324    CLC   OPER,=CL8'INSERT'   A SQL INSERT STATEMENT?
00062C 4780 C7A8       007A8 1325         BE    TRACEI
000630 47F0 C8FC       008FC 1326         B     TRACED
                             1328 *-----------------------------------------------------------------*
                             1329 *       TRACE THE SQL UPDATE STATEMENT                            *
                             1330 *                                                                 *
                             1331 *       FOR EACH ELEMENT TO BE INCLUDED IN THE TRACE: INVOKE      *
                             1332 *           A SETTED MACRO DESCRIBING THE ELEMENT.                *
                             1333 *       SET INTO THE ADDRESS OF THE LAST TED THE 'VL BIT'         *
                             1334 *           IDENTIFYING THE END OF THE CALL PARAMETER LIST FOR    *
                             1335 *           THE DPROP_TRACER.                                     *
                             1336 *-----------------------------------------------------------------*

000634                       1338 TRACEU   DS    0H
                             1339 *
                             1340 ***           PROVIDE 3RD TED (SUBHEADER 'COLUMNS IN WHERE CLAUSE')
                             1341 *
                             1342         SETTED NBR=3,TYPE=SUBH,TXT=TXTWH
000634 4140 D1A8       001A8 1343+        LA    R4,WORKTED+((3-1)*TEDLEN)    R4=A(TED)
                       00000 1344+        USING TED,R4
000638 5040 D0F4       000F4 1345+        ST    R4,TRAPARML+(4*3)            SET A(TED) INTO PARMLIST
00063C D203 4000 CAFC 00000 00AFC 1346+   MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
```

Figure 52 (Part 21 of 40). First Sample Propagation Exit Routine (Assembler)

```
000642 92E2 4004      00004        1347+         MVI   TEDTYPE,TEDTYPS         SET 'THIS IS A SUBHEADER-TED'
000646 41F0 C9C6            009C6  1348+         LA    R15,TXTWH               R15=A(TEXT)
00064A 50F0 4008            00008  1349+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
00064E D203 400C CB08 0000C 00B08  1350+         MVC   TEDTXTL,=A(L'TXTWH)     SET LENGTH OF TEXT STRING
                                   1351 *
                                   1352 ***            PROVIDE 4TH TED (DATA FOR 1ST COLUMN IN WHERE CLAUSE)
                                   1353 *
                                   1354          SETTED NBR=4,TYPE=DATA,TXT=TXTCOL1,DATA=FCK_SEG1KEY1
000654 4140 D1CC            001CC  1355+         LA    R4,WORKTED+((4-1)*TEDLEN)    R4=A(TED)
                            00000  1356+         USING TED,R4
000658 5040 D0F8            000F8  1357+         ST    R4,TRAPARML+(4*4)            SET A(TED) INTO PARMLIST
00065C D203 4000 CAFC 00000 00AFC  1358+         MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
000662 92C4 4004      00004        1359+         MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
000666 41F0 C9EF            009EF  1360+         LA    R15,TXTCOL1             R15=A(TEXT)
00066A 50F0 4008            00008  1361+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
00066E D203 400C CB0C 0000C 00B0C  1362+         MVC   TEDTXTL,=A(L'TXTCOL1)   SET LENGTH OF TEXT STRING
000674 41F0 7000            00000  1363+         LA    R15,FCK_SEG1KEY1        R15=A(DATA)
000678 50F0 4010            00010  1364+         ST    R15,TEDMA               STORE A(DATA) INTO TED
00067C D203 4014 CB10 00014 00B10  1365+         MVC   TEDALEN,=A(L'FCK_SEG1KEY1)                              X
                                      +                                STORE LENGTH OF DATA
000682 92D3 4005      00005        1366+         MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                                   1367 *
                                   1368 ***            PROVIDE 5TH TED (DATA FOR 2ND COLUMN IN WHERE CLAUSE)
                                   1369 *
                                   1370          SETTED NBR=5,TYPE=DATA,TXT=TXTCOL2,DATA=FCK_SEG2KEY1
000686 4140 D1F0            001F0  1371+         LA    R4,WORKTED+((5-1)*TEDLEN)    R4=A(TED)
                            00000  1372+         USING TED,R4
00068A 5040 D0FC            000FC  1373+         ST    R4,TRAPARML+(4*5)            SET A(TED) INTO PARMLIST
00068E D203 4000 CAFC 00000 00AFC  1374+         MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
000694 92C4 4004      00004        1375+         MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
000698 41F0 C9F7            009F7  1376+         LA    R15,TXTCOL2             R15=A(TEXT)
00069C 50F0 4008            00008  1377+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0006A0 D203 400C CB14 0000C 00B14  1378+         MVC   TEDTXTL,=A(L'TXTCOL2)   SET LENGTH OF TEXT STRING
0006A6 41F0 7005            00005  1379+         LA    R15,FCK_SEG2KEY1        R15=A(DATA)
0006AA 50F0 4010            00010  1380+         ST    R15,TEDMA               STORE A(DATA) INTO TED
0006AE D203 4014 CB18 00014 00B18  1381+         MVC   TEDALEN,=A(L'FCK_SEG2KEY1)                              X
                                      +                                STORE LENGTH OF DATA
0006B4 92D3 4005      00005        1382+         MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                                   1383 *
                                   1384 ***            PROVIDE 6TH TED (DATA FOR 3RD COLUMN IN WHERE CLAUSE)
                                   1385 *
                                   1386          SETTED NBR=6,TYPE=DATA,TXT=TXTCOL3,DATA=FCK_SEG2KEY2
0006B8 4140 D214            00214  1387+         LA    R4,WORKTED+((6-1)*TEDLEN)    R4=A(TED)
                            00000  1388+         USING TED,R4
0006BC 5040 D100            00100  1389+         ST    R4,TRAPARML+(4*6)            SET A(TED) INTO PARMLIST
0006C0 D203 4000 CAFC 00000 00AFC  1390+         MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
0006C6 92C4 4004      00004        1391+         MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
0006CA 41F0 C9FF            009FF  1392+         LA    R15,TXTCOL3             R15=A(TEXT)
0006CE 50F0 4008            00008  1393+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0006D2 D203 400C CB1C 0000C 00B1C  1394+         MVC   TEDTXTL,=A(L'TXTCOL3)   SET LENGTH OF TEXT STRING
0006D8 41F0 7007            00007  1395+         LA    R15,FCK_SEG2KEY2        R15=A(DATA)
0006DC 50F0 4010            00010  1396+         ST    R15,TEDMA               STORE A(DATA) INTO TED
0006E0 D203 4014 CB20 00014 00B20  1397+         MVC   TEDALEN,=A(L'FCK_SEG2KEY2)                              X
                                      +                                STORE LENGTH OF DATA
0006E6 92D3 4005      00005        1398+         MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                                   1399 *
                                   1400 ***            PROVIDE 7TH TED (SUBHEADER 'PROPAGATED COLUMNS')
                                   1401 *
                                   1402          SETTED NBR=7,TYPE=SUBH,TXT=TXTPRC
0006EA 4140 D238            00238  1403+         LA    R4,WORKTED+((7-1)*TEDLEN)    R4=A(TED)
                            00000  1404+         USING TED,R4
0006EE 5040 D104            00104  1405+         ST    R4,TRAPARML+(4*7)            SET A(TED) INTO PARMLIST
0006F2 D203 4000 CAFC 00000 00AFC  1406+         MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
0006F8 92E2 4004      00004        1407+         MVI   TEDTYPE,TEDTYPS         SET 'THIS IS A SUBHEADER-TED'
0006FC 41F0 C9DD            009DD  1408+         LA    R15,TXTPRC              R15=A(TEXT)
000700 50F0 4008            00008  1409+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
000704 D203 400C CB24 0000C 00B24  1410+         MVC   TEDTXTL,=A(L'TXTPRC)    SET LENGTH OF TEXT STRING
```

Figure 52 (Part 22 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                    1411 *
                                    1412 ***          PROVIDE 8TH TED (DATA FOR 1ST PROPAGATED COLUMN)
                                    1413 *
                                    1414        SETTED NBR=8,TYPE=DATA,TXT=TXTCOL4,DATA=SEG2DAT1
00070A 4140 D25C               0025C 1415+       LA    R4,WORKTED+((8-1)*TEDLEN)    R4=A(TED)
                               00000 1416+       USING TED,R4
00070E 5040 D108               00108 1417+       ST    R4,TRAPARML+(4*8)            SET A(TED) INTO PARMLIST
000712 D203 4000 CAFC 00000 00AFC 1418+       MVC   TEDEYE,=CL4'TED'       SET EYE CATCHER INTO TED
000718 92C4 4004      00004        1419+       MVI   TEDTYPE,TEDTYPD        SET 'THIS IS A DATA-TED'
00071C 41F0 CA07               00A07 1420+       LA    R15,TXTCOL4            R15=A(TEXT)
000720 50F0 4008               00008 1421+       ST    R15,TEDTXTA            STORE A(TEXT) INTO TED
000724 D203 400C CB28 0000C 00B28 1422+       MVC   TEDTXTL,=A(L'TXTCOL4)  SET LENGTH OF TEXT STRING
00072A 41F0 6008               00008 1423+       LA    R15,SEG2DAT1           R15=A(DATA)
00072E 50F0 4010               00010 1424+       ST    R15,TEDMA              STORE A(DATA) INTO TED
000732 D203 4014 CB2C 00014 00B2C 1425+       MVC   TEDALEN,=A(L'SEG2DAT1)  STORE LENGTH OF DATA
000738 92D3 4005      00005        1426+       MVI   TEDALIGN,TEDALIGL      REQUEST 'LEFT ALIGNMENT'
                                    1427 *
                                    1428 ***          PROVIDE 9TH TED (DATA FOR 2ND PROPAGATED COLUMN)
                                    1429 *
                                    1430        SETTED NBR=9,TYPE=DATA,TXT=TXTCOL5,DATA=SEG2DAT2
00073C 4140 D280               00280 1431+       LA    R4,WORKTED+((9-1)*TEDLEN)    R4=A(TED)
                               00000 1432+       USING TED,R4
000740 5040 D10C               0010C 1433+       ST    R4,TRAPARML+(4*9)            SET A(TED) INTO PARMLIST
000744 D203 4000 CAFC 00000 00AFC 1434+       MVC   TEDEYE,=CL4'TED'       SET EYE CATCHER INTO TED
00074A 92C4 4004      00004        1435+       MVI   TEDTYPE,TEDTYPD        SET 'THIS IS A DATA-TED'
00074E 41F0 CA0F               00A0F 1436+       LA    R15,TXTCOL5            R15=A(TEXT)
000752 50F0 4008               00008 1437+       ST    R15,TEDTXTA            STORE A(TEXT) INTO TED
000756 D203 400C CB30 0000C 00B30 1438+       MVC   TEDTXTL,=A(L'TXTCOL5)  SET LENGTH OF TEXT STRING
00075C 41F0 6010               00010 1439+       LA    R15,SEG2DAT2           R15=A(DATA)
000760 50F0 4010               00010 1440+       ST    R15,TEDMA              STORE A(DATA) INTO TED
000764 D203 4014 CB34 00014 00B34 1441+       MVC   TEDALEN,=A(L'SEG2DAT2)  STORE LENGTH OF DATA
00076A 92D3 4005      00005        1442+       MVI   TEDALIGN,TEDALIGL      REQUEST 'LEFT ALIGNMENT'
                                    1443 *
                                    1444 ***          PROVIDE 10TH TED (DATA FOR 3RD PROPAGATED COLUMN)
                                    1445 *
                                    1446        SETTED NBR=10,TYPE=DATA,TXT=TXTCOL6,DATA=SEG1DAT1
00076E 4140 D2A4               002A4 1447+       LA    R4,WORKTED+((10-1)*TEDLEN)   R4=A(TED)
                               00000 1448+       USING TED,R4
000772 5040 D110               00110 1449+       ST    R4,TRAPARML+(4*10)           SET A(TED) INTO PARMLIST
000776 D203 4000 CAFC 00000 00AFC 1450+       MVC   TEDEYE,=CL4'TED'       SET EYE CATCHER INTO TED
00077C 92C4 4004      00004        1451+       MVI   TEDTYPE,TEDTYPD        SET 'THIS IS A DATA-TED'
000780 41F0 CA17               00A17 1452+       LA    R15,TXTCOL6            R15=A(TEXT)
000784 50F0 4008               00008 1453+       ST    R15,TEDTXTA            STORE A(TEXT) INTO TED
000788 D203 400C CB38 0000C 00B38 1454+       MVC   TEDTXTL,=A(L'TXTCOL6)  SET LENGTH OF TEXT STRING
00078E 41F0 5005               00005 1455+       LA    R15,SEG1DAT1           R15=A(DATA)
000792 50F0 4010               00010 1456+       ST    R15,TEDMA              STORE A(DATA) INTO TED
000796 D203 4014 CB3C 00014 00B3C 1457+       MVC   TEDALEN,=A(L'SEG1DAT1)  STORE LENGTH OF DATA
00079C 92D3 4005      00005        1458+       MVI   TEDALIGN,TEDALIGL      REQUEST 'LEFT ALIGNMENT'
                                    1459 *
                                    1460 ***          SET INTO PARAMETER LIST THE 'HIGH ORDER BIT'
                                    1461 ***          (I.E. THE 'VL BIT') WHICH SIGNALS THE END OF THE
                                    1462 ***          PARAMETER LIST.
                                    1463 *
0007A0 9680 D110      00110        1464        OI    TRATED10,X'80'      SET VL-BIT INTO TRACE PARMLIST
0007A4 47F0 C9BA               009BA 1465        B     TRACECO             GO TO COMMON TRACE LOGIC
                                    1467 *-------------------------------------------------------------------*
                                    1468 *      TRACE THE SQL INSERT STATEMENT                               *
                                    1469 *                                                                   *
                                    1470 *      FOR EACH ELEMENT TO BE INCLUDED IN THE TRACE: INVOKE         *
                                    1471 *          A SETTED MACRO DESCRIBING THE ELEMENT.                   *
                                    1472 *      SET INTO THE ADDRESS OF THE LAST TED THE 'VL BIT'            *
                                    1473 *          IDENTIFYING THE END OF THE CALL PARAMETER LIST FOR       *
                                    1474 *          THE DPROP TRACER.                                        *
                                    1475 *-------------------------------------------------------------------*
```

Figure 52 (Part 23 of 40). First Sample Propagation Exit Routine (Assembler)

```
0007A8                          1477 TRACEI   DS    0H
                                1478 *
                                1479 ***            PROVIDE 3RD TED (SUBHEADER 'PROPAGATED COLUMNS'
                                1480 *
                                1481          SETTED NBR=3,TYPE=SUBH,TXT=TXTPRC
0007A8 4140 D1A8          001A8 1482+         LA    R4,WORKTED+((3-1)*TEDLEN)    R4=A(TED)
                          00000 1483+         USING TED,R4
0007AC 5040 D0F4          000F4 1484+         ST    R4,TRAPARML+(4*3)            SET A(TED) INTO PARMLIST
0007B0 D203 4000 CAFC 00000 00AFC 1485+       MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
0007B6 92E2 4004      00004     1486+         MVI   TEDTYPE,TEDTYPS         SET 'THIS IS A SUBHEADER-TED'
0007BA 41F0 C9DD          009DD 1487+         LA    R15,TXTPRC              R15=A(TEXT)
0007BE 50F0 4008          00008 1488+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0007C2 D203 400C CB24 0000C 00B24 1489+       MVC   TEDTXTL,=A(L'TXTPRC)    SET LENGTH OF TEXT STRING
                                1490 *
                                1491 ***            PROVIDE 4TH TED (DATA FOR 1ST PROPAGATED COLUMN)
                                1492 *
                                1493          SETTED NBR=4,TYPE=DATA,TXT=TXTCOL1,DATA=FCK_SEG1KEY1
0007C8 4140 D1CC          001CC 1494+         LA    R4,WORKTED+((4-1)*TEDLEN)    R4=A(TED)
                          00000 1495+         USING TED,R4
0007CC 5040 D0F8          000F8 1496+         ST    R4,TRAPARML+(4*4)            SET A(TED) INTO PARMLIST
0007D0 D203 4000 CAFC 00000 00AFC 1497+       MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
0007D6 92C4 4004      00004     1498+         MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
0007DA 41F0 C9EF          009EF 1499+         LA    R15,TXTCOL1             R15=A(TEXT)
0007DE 50F0 4008          00008 1500+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0007E2 D203 400C CB0C 0000C 00B0C 1501+       MVC   TEDTXTL,=A(L'TXTCOL1)   SET LENGTH OF TEXT STRING
0007E8 41F0 7000          00000 1502+         LA    R15,FCK_SEG1KEY1        R15=A(DATA)
0007EC 50F0 4010          00010 1503+         ST    R15,TEDMA               STORE A(DATA) INTO TED
0007F0 D203 4014 CB10 00014 00B10 1504+       MVC   TEDALEN,=A(L'FCK_SEG1KEY1)                        X
                                   +                                    STORE LENGTH OF DATA
0007F6 92D3 4005      00005     1505+         MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                                1506 *
                                1507 ***            PROVIDE 5TH TED (DATA FOR 2ND PROPAGATED COLUMN)
                                1508 *
                                1509          SETTED NBR=5,TYPE=DATA,TXT=TXTCOL2,DATA=FCK_SEG2KEY1
0007FA 4140 D1F0          001F0 1510+         LA    R4,WORKTED+((5-1)*TEDLEN)    R4=A(TED)
                          00000 1511+         USING TED,R4
0007FE 5040 D0FC          000FC 1512+         ST    R4,TRAPARML+(4*5)            SET A(TED) INTO PARMLIST
000802 D203 4000 CAFC 00000 00AFC 1513+       MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
000808 92C4 4004      00004     1514+         MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
00080C 41F0 C9F7          009F7 1515+         LA    R15,TXTCOL2             R15=A(TEXT)
000810 50F0 4008          00008 1516+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
000814 D203 400C CB14 0000C 00B14 1517+       MVC   TEDTXTL,=A(L'TXTCOL2)   SET LENGTH OF TEXT STRING
00081A 41F0 7005          00005 1518+         LA    R15,FCK_SEG2KEY1        R15=A(DATA)
00081E 50F0 4010          00010 1519+         ST    R15,TEDMA               STORE A(DATA) INTO TED
000822 D203 4014 CB18 00014 00B18 1520+       MVC   TEDALEN,=A(L'FCK_SEG2KEY1)                        X
                                   +                                    STORE LENGTH OF DATA
000828 92D3 4005      00005     1521+         MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                                1522 *
                                1523 ***            PROVIDE 6TH TED (DATA FOR 3RD PROPAGATED COLUMN)
                                1524 *
                                1525          SETTED NBR=6,TYPE=DATA,TXT=TXTCOL3,DATA=FCK_SEG2KEY2
00082C 4140 D214          00214 1526+         LA    R4,WORKTED+((6-1)*TEDLEN)    R4=A(TED)
                          00000 1527+         USING TED,R4
000830 5040 D100          00100 1528+         ST    R4,TRAPARML+(4*6)            SET A(TED) INTO PARMLIST
000834 D203 4000 CAFC 00000 00AFC 1529+       MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
00083A 92C4 4004      00004     1530+         MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
00083E 41F0 C9FF          009FF 1531+         LA    R15,TXTCOL3             R15=A(TEXT)
000842 50F0 4008          00008 1532+         ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
000846 D203 400C CB1C 0000C 00B1C 1533+       MVC   TEDTXTL,=A(L'TXTCOL3)   SET LENGTH OF TEXT STRING
00084C 41F0 7007          00007 1534+         LA    R15,FCK_SEG2KEY2        R15=A(DATA)
000850 50F0 4010          00010 1535+         ST    R15,TEDMA               STORE A(DATA) INTO TED
000854 D203 4014 CB20 00014 00B20 1536+       MVC   TEDALEN,=A(L'FCK_SEG2KEY2)                        X
                                   +                                    STORE LENGTH OF DATA
00085A 92D3 4005      00005     1537+         MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                                1538 *
                                1539 ***            PROVIDE 7TH TED (DATA FOR 4TH PROPAGATED COLUMN)
```

Figure 52 (Part 24 of 40). First Sample Propagation Exit Routine (Assembler)

```
                              1540 *
                              1541        SETTED NBR=7,TYPE=DATA,TXT=TXTCOL4,DATA=SEG2DAT1
00085E 4140 D238        00238 1542+       LA    R4,WORKTED+((7-1)*TEDLEN)    R4=A(TED)
                        00000 1543+       USING TED,R4
000862 5040 D104        00104 1544+       ST    R4,TRAPARML+(4*7)           SET A(TED) INTO PARMLIST
000866 D203 4000 CAFC 00000 00AFC 1545+   MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
00086C 92C4 4004      00004   1546+       MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
000870 41F0 CA07        00A07 1547+       LA    R15,TXTCOL4             R15=A(TEXT)
000874 50F0 4008        00008 1548+       ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
000878 D203 400C CB28 0000C 00B28 1549+   MVC   TEDTXTL,=A(L'TXTCOL4)   SET LENGTH OF TEXT STRING
00087E 41F0 6008        00008 1550+       LA    R15,SEG2DAT1            R15=A(DATA)
000882 50F0 4010        00010 1551+       ST    R15,TEDMA               STORE A(DATA) INTO TED
000886 D203 4014 CB2C 00014 00B2C 1552+   MVC   TEDALEN,=A(L'SEG2DAT1)  STORE LENGTH OF DATA
00088C 92D3 4005      00005   1553+       MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                              1554 *
                              1555 ***          PROVIDE 8TH TED (DATA FOR 5TH PROPAGATED COLUMN)
                              1556 *
                              1557        SETTED NBR=8,TYPE=DATA,TXT=TXTCOL5,DATA=SEG2DAT2
000890 4140 D25C        0025C 1558+       LA    R4,WORKTED+((8-1)*TEDLEN)    R4=A(TED)
                        00000 1559+       USING TED,R4
000894 5040 D108        00108 1560+       ST    R4,TRAPARML+(4*8)           SET A(TED) INTO PARMLIST
000898 D203 4000 CAFC 00000 00AFC 1561+   MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
00089E 92C4 4004      00004   1562+       MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
0008A2 41F0 CA0F        00A0F 1563+       LA    R15,TXTCOL5             R15=A(TEXT)
0008A6 50F0 4008        00008 1564+       ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0008AA D203 400C CB30 0000C 00B30 1565+   MVC   TEDTXTL,=A(L'TXTCOL5)   SET LENGTH OF TEXT STRING
0008B0 41F0 6010        00010 1566+       LA    R15,SEG2DAT2            R15=A(DATA)
0008B4 50F0 4010        00010 1567+       ST    R15,TEDMA               STORE A(DATA) INTO TED
0008B8 D203 4014 CB34 00014 00B34 1568+   MVC   TEDALEN,=A(L'SEG2DAT2)  STORE LENGTH OF DATA
0008BE 92D3 4005      00005   1569+       MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                              1570 *
                              1571 ***          PROVIDE 9TH TED (DATA FOR 6TH PROPAGATED COLUMN)
                              1572 *
                              1573        SETTED NBR=9,TYPE=DATA,TXT=TXTCOL6,DATA=SEG1DAT1
0008C2 4140 D280        00280 1574+       LA    R4,WORKTED+((9-1)*TEDLEN)    R4=A(TED)
                        00000 1575+       USING TED,R4
0008C6 5040 D10C        0010C 1576+       ST    R4,TRAPARML+(4*9)           SET A(TED) INTO PARMLIST
0008CA D203 4000 CAFC 00000 00AFC 1577+   MVC   TEDEYE,=CL4'TED'        SET EYE CATCHER INTO TED
0008D0 92C4 4004      00004   1578+       MVI   TEDTYPE,TEDTYPD         SET 'THIS IS A DATA-TED'
0008D4 41F0 CA17        00A17 1579+       LA    R15,TXTCOL6             R15=A(TEXT)
0008D8 50F0 4008        00008 1580+       ST    R15,TEDTXTA             STORE A(TEXT) INTO TED
0008DC D203 400C CB38 0000C 00B38 1581+   MVC   TEDTXTL,=A(L'TXTCOL6)   SET LENGTH OF TEXT STRING
0008E2 41F0 5005        00005 1582+       LA    R15,SEG1DAT1            R15=A(DATA)
0008E6 50F0 4010        00010 1583+       ST    R15,TEDMA               STORE A(DATA) INTO TED
0008EA D203 4014 CB3C 00014 00B3C 1584+   MVC   TEDALEN,=A(L'SEG1DAT1)  STORE LENGTH OF DATA
0008F0 92D3 4005      00005   1585+       MVI   TEDALIGN,TEDALIGL       REQUEST 'LEFT ALIGNMENT'
                              1586 *
                              1587 ***          SET INTO PARAMETER LIST THE 'HIGH ORDER BIT'
                              1588 ***          (I.E. THE 'VL BIT') WHICH SIGNALS THE END OF THE
                              1589 ***          PARAMETER LIST.
                              1590 *
0008F4 9680 D10C      0010C   1591        OI    TRATED9,X'80'       SET VL-BIT INTO TRACE PARMLIST
0008F8 47F0 C9BA        009BA 1592        B     TRACECO             GO TO COMMON TRACE LOGIC
                              1594 *-------------------------------------------------------------------*
                              1595 *      TRACE THE SQL DELETE STATEMENT                               *
                              1596 *                                                                   *
                              1597 *      FOR EACH ELEMENT TO BE INCLUDED IN THE TRACE: INVOKE         *
                              1598 *          A SETTED MACRO DESCRIBING THE ELEMENT.                   *
                              1599 *      SET INTO THE ADDRESS OF THE LAST TED THE 'VL BIT'            *
                              1600 *          IDENTIFYING THE END OF THE CALL PARAMETER LIST FOR       *
                              1601 *          THE DPROP TRACER.                                        *
                              1602 *-------------------------------------------------------------------*

0008FC                        1604 TRACED  DS    0H
                              1605 *
                              1606 ***          PROVIDE 3RD TED (SUBHEADER 'COLUMNS IN WHERE CLAUSE')
```

Figure 52 (Part 25 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                   1607 *
                                   1608         SETTED NBR=3,TYPE=SUBH,TXT=TXTWH
0008FC 4140 D1A8             001A8 1609+        LA    R4,WORKTED+((3-1)*TEDLEN)    R4=A(TED)
                             00000 1610+        USING TED,R4
000900 5040 D0F4             000F4 1611+        ST    R4,TRAPARML+(4*3)            SET A(TED) INTO PARMLIST
000904 D203 4000 CAFC 00000 00AFC  1612+        MVC   TEDEYE,=CL4'TED'         SET EYE CATCHER INTO TED
00090A 92E2 4004       00004       1613+        MVI   TEDTYPE,TEDTYPS          SET 'THIS IS A SUBHEADER-TED'
00090E 41F0 C9C6             009C6 1614+        LA    R15,TXTWH                R15=A(TEXT)
000912 50F0 4008             00008 1615+        ST    R15,TEDTXTA              STORE A(TEXT) INTO TED
000916 D203 400C CB08 0000C 00B08  1616+        MVC   TEDTXTL,=A(L'TXTWH)      SET LENGTH OF TEXT STRING
                                   1617 *
                                   1618 ***           PROVIDE 4TH TED (DATA FOR 1ST COLUMN IN WHERE CLAUSE)
                                   1619 *
                                   1620         SETTED NBR=4,TYPE=DATA,TXT=TXTCOL1,DATA=FCK_SEG1KEY1
00091C 4140 D1CC             001CC 1621+        LA    R4,WORKTED+((4-1)*TEDLEN)    R4=A(TED)
                             00000 1622+        USING TED,R4
000920 5040 D0F8             000F8 1623+        ST    R4,TRAPARML+(4*4)            SET A(TED) INTO PARMLIST
000924 D203 4000 CAFC 00000 00AFC  1624+        MVC   TEDEYE,=CL4'TED'         SET EYE CATCHER INTO TED
00092A 92C4 4004       00004       1625+        MVI   TEDTYPE,TEDTYPD          SET 'THIS IS A DATA-TED'
00092E 41F0 C9EF             009EF 1626+        LA    R15,TXTCOL1              R15=A(TEXT)
000932 50F0 4008             00008 1627+        ST    R15,TEDTXTA              STORE A(TEXT) INTO TED
000936 D203 400C CB0C 0000C 00B0C  1628+        MVC   TEDTXTL,=A(L'TXTCOL1)    SET LENGTH OF TEXT STRING
00093C 41F0 7000             00000 1629+        LA    R15,FCK_SEG1KEY1         R15=A(DATA)
000940 50F0 4010             00010 1630+        ST    R15,TEDMA                STORE A(DATA) INTO TED
000944 D203 4014 CB10 00014 00B10  1631+        MVC   TEDALEN,=A(L'FCK_SEG1KEY1)                              X
                                      +                                STORE LENGTH OF DATA
00094A 92D3 4005       00005       1632+        MVI   TEDALIGN,TEDALIGL        REQUEST 'LEFT ALIGNMENT'
                                   1633 *
                                   1634 ***           PROVIDE 5TH TED (DATA FOR 2ND COLUMN IN WHERE CLAUSE)
                                   1635 *
                                   1636         SETTED NBR=5,TYPE=DATA,TXT=TXTCOL2,DATA=FCK_SEG2KEY1
00094E 4140 D1F0             001F0 1637+        LA    R4,WORKTED+((5-1)*TEDLEN)    R4=A(TED)
                             00000 1638+        USING TED,R4
000952 5040 D0FC             000FC 1639+        ST    R4,TRAPARML+(4*5)            SET A(TED) INTO PARMLIST
000956 D203 4000 CAFC 00000 00AFC  1640+        MVC   TEDEYE,=CL4'TED'         SET EYE CATCHER INTO TED
00095C 92C4 4004       00004       1641+        MVI   TEDTYPE,TEDTYPD          SET 'THIS IS A DATA-TED'
000960 41F0 C9F7             009F7 1642+        LA    R15,TXTCOL2              R15=A(TEXT)
000964 50F0 4008             00008 1643+        ST    R15,TEDTXTA              STORE A(TEXT) INTO TED
000968 D203 400C CB14 0000C 00B14  1644+        MVC   TEDTXTL,=A(L'TXTCOL2)    SET LENGTH OF TEXT STRING
00096E 41F0 7005             00005 1645+        LA    R15,FCK_SEG2KEY1         R15=A(DATA)
000972 50F0 4010             00010 1646+        ST    R15,TEDMA                STORE A(DATA) INTO TED
000976 D203 4014 CB18 00014 00B18  1647+        MVC   TEDALEN,=A(L'FCK_SEG2KEY1)                              X
                                      +                                STORE LENGTH OF DATA
00097C 92D3 4005       00005       1648+        MVI   TEDALIGN,TEDALIGL        REQUEST 'LEFT ALIGNMENT'
                                   1649 *
                                   1650 ***           PROVIDE 6TH TED (DATA FOR 3RD COLUMN IN WHERE CLAUSE)
                                   1651 *
                                   1652         SETTED NBR=6,TYPE=DATA,TXT=TXTCOL3,DATA=FCK_SEG2KEY2
000980 4140 D214             00214 1653+        LA    R4,WORKTED+((6-1)*TEDLEN)    R4=A(TED)
                             00000 1654+        USING TED,R4
000984 5040 D100             00100 1655+        ST    R4,TRAPARML+(4*6)            SET A(TED) INTO PARMLIST
000988 D203 4000 CAFC 00000 00AFC  1656+        MVC   TEDEYE,=CL4'TED'         SET EYE CATCHER INTO TED
00098E 92C4 4004       00004       1657+        MVI   TEDTYPE,TEDTYPD          SET 'THIS IS A DATA-TED'
000992 41F0 C9FF             009FF 1658+        LA    R15,TXTCOL3              R15=A(TEXT)
000996 50F0 4008             00008 1659+        ST    R15,TEDTXTA              STORE A(TEXT) INTO TED
00099A D203 400C CB1C 0000C 00B1C  1660+        MVC   TEDTXTL,=A(L'TXTCOL3)    SET LENGTH OF TEXT STRING
0009A0 41F0 7007             00007 1661+        LA    R15,FCK_SEG2KEY2         R15=A(DATA)
0009A4 50F0 4010             00010 1662+        ST    R15,TEDMA                STORE A(DATA) INTO TED
0009A8 D203 4014 CB20 00014 00B20  1663+        MVC   TEDALEN,=A(L'FCK_SEG2KEY2)                              X
                                      +                                STORE LENGTH OF DATA
0009AE 92D3 4005       00005       1664+        MVI   TEDALIGN,TEDALIGL        REQUEST 'LEFT ALIGNMENT'
                                   1665 *
                                   1666 ***           SET INTO PARAMETER LIST THE 'HIGH ORDER BIT'
                                   1667 ***           (I.E. THE 'VL BIT') WHICH SIGNALS THE END OF THE
                                   1668 ***           PARAMETER LIST.
                                   1669 *
```

Figure 52 (Part 26 of 40). First Sample Propagation Exit Routine (Assembler)

```
0009B2 9680 D100      00100        1670          OI    TRATED6,X'80'       SET VL BIT INTO TRACE PARMLIST
0009B6 47F0 C9BA            009BA  1671          B     TRACECO             GO TO COMMON TRACE LOGIC


                                   1673 *--------------------------------------------------------------------*
                                   1674 *              CALL DPROP TRACE MODULE WITH THE PREVIOUSLY           *
                                   1675 *              FORMATTED PARAMETER LIST.                             *
                                   1676 *--------------------------------------------------------------------*

0009BA                             1678 TRACECO  DS    0H

0009BA 4110 D0E8            000E8  1680          LA    R1,TRAPARML         R1=TRACE PARAMETER LIST FOR TRACER
0009BE 58F0 CB40            00B40  1681          L     R15,=V(EKYR410X)    CALL
0009C2 0DEF                        1682          BASR  R14,R15             ...DPROP TRACER MODULE

0009C4 07FB                        1684          BR    R11                 RETURN TO CALLER OF SUBROUTINE


                                   1686 *--------------------------------------------------------------------*
                                   1687 *              TEXT USED FOR TRACING                                 *
                                   1688 *--------------------------------------------------------------------*

0009C6 C3D6D3E4D4D5E240            1690 TXTWH    DC    C'COLUMNS IN WHERE CLAUSE'   TEXT FOR TRACE SUBHEADER
0009DD D7D9D6D7C1C7C1E3            1691 TXTPRC   DC    C'PROPAGATED COLUMNS'        TEXT FOR TRACE SUBHEADER
0009EF E3C1C2F2C3D6D3F1            1692 TXTCOL1  DC    CL8'TAB2COL1'                TEXT FOR TRACE
0009F7 E3C1C2F2C3D6D3F2            1693 TXTCOL2  DC    CL8'TAB2COL2'                TEXT FOR TRACE
0009FF E3C1C2F2C3D6D3F3            1694 TXTCOL3  DC    CL8'TAB2COL3'                TEXT FOR TRACE
000A07 E3C1C2F2C3D6D3F4            1695 TXTCOL4  DC    CL8'TAB2COL4'                TEXT FOR TRACE
000A0F E3C1C2F2C3D6D3F5            1696 TXTCOL5  DC    CL8'TAB2COL5'                TEXT FOR TRACE
000A17 E3C1C2F2C3D6D3F6            1697 TXTCOL6  DC    CL8'TAB2COL6'                TEXT FOR TRACE


000A20                             1699          LTORG
000A20 4040404040404040            1700                =CL8' '
000A28 C4C2F14040404040            1701                =CL8'DB1'
000A30 E2C5C7F240404040            1702                =CL8'SEG2'
000A38 E4D7C4C1E3C54040            1703                =CL8'UPDATE'
000A40 C9D5E2C5D9E34040            1704                =CL8'INSERT'
000A48 C4C5D3C5E3C54040            1705                =CL8'DELETE'
000A50 C5D2E8C5D7D9F0C5            1706                =CL8'EKYEPR0E'
000A58 40E2D8D340C5D9D9            1707                =CL16' SQL ERROR CODE='
000A68 C5D2E8C5D7D9F1C5            1708                =CL8'EKYEPR1E'
000A70 C5D2E8C5D7D9F2C5            1709                =CL8'EKYEPR2E'
000A78 D2C5E840D6C640E2            1710                =C'KEY OF SEG2 NOT PROVIDED BY DL/I CAPTURE'
000AA0 C5D2E8C5D7D9F3C5            1711                =CL8'EKYEPR3E'
000AA8 C5D2E8C5D7D9F4C5            1712                =CL8'EKYEPR4E'
000AB0 C5D2E8C5D7D9F5C5            1713                =CL8'EKYEPR5E'
000AB8 C4C2C4D5C1D4C57E            1714                =CL08'DBDNAME='
000AC0 C5D2E8C5D7D9F1C1            1715                =CL8'EKYEPR1A'
000AC8 C9E2D9E3                    1716                =CL4'ISRT'
000ACC D9C5D7D3                    1717                =CL4'REPL'
000AD0 D9C5C9D5                    1718                =CL4'REIN'
000AD4 C4D3C5E3                    1719                =CL4'DLET'
000AD8 C4D3D7D7                    1720                =CL4'DLPP'
000ADC 0000004C                    1721                =F'76'
000AE0 00000000                    1722                =XL4'00000000'
000AE4 00000000                    1723                =V(DSNHLI)
000AE8 C4D3D3D7                    1724                =C'DLLP'
000AEC 00000028                    1725                =F'40'
000AF0 00000064                    1726                =F'100'
000AF4 C3C1E2C3                    1727                =C'CASC'
000AF8 E3D9C240                    1728                =CL4'TRB '
000AFC E3C5C440                    1729                =CL4'TED'
000B00 00000035                    1730                =A(L'TRHEADER)
000B04 00000013                    1731                =A(L'TXTSQLC)
000B08 00000017                    1732                =A(L'TXTWH)
```

Figure 52 (Part 27 of 40). First Sample Propagation Exit Routine (Assembler)

```
000B0C 00000008                1733               =A(L'TXTCOL1)
000B10 00000005                1734               =A(L'FCK_SEG1KEY1)
000B14 00000008                1735               =A(L'TXTCOL2)
000B18 00000002                1736               =A(L'FCK_SEG2KEY1)
000B1C 00000008                1737               =A(L'TXTCOL3)
000B20 00000006                1738               =A(L'FCK_SEG2KEY2)
000B24 00000012                1739               =A(L'TXTPRC)
000B28 00000008                1740               =A(L'TXTCOL4)
000B2C 00000008                1741               =A(L'SEG2DAT1)
000B30 00000008                1742               =A(L'TXTCOL5)
000B34 00000008                1743               =A(L'SEG2DAT2)
000B38 00000008                1744               =A(L'TXTCOL6)
000B3C 00000007                1745               =A(L'SEG1DAT1)
000B40 00000000                1746               =V(EKYR410X)
000B44 E3C1C2F240404040        1747               =CL18'TAB2'
000B56 01C4                    1748               =X'01C4'
000B58 0008                    1749               =H'8'
000B5A 0007                    1750               =H'7'
000B5C 0005                    1751               =H'5'
000B5E 0002                    1752               =H'2'
000B60 0006                    1753               =H'6'
000B62 0004                    1754               =H'4'
000B64 D7D9D6D7C1C7C1E3        1755               =CL30'PROPAGATION FAILURE FOR TABLE='
000B82 C6C1C9D3C9D5C740        1756               =CL22'FAILING SQL STATEMENT='
000B98 D7C1E3C840C4C1E3        1757               =C'PATH DATA NOT PROVIDED BY DL/I CAPTURE'
000BBE 40C6E4D5C37E            1758               =CL06' FUNC='
000BC4 0014                    1759               =H'20'
000BC6 E4D5C5E7D7C5C3E3        1760               =C'UNEXPECTED DBD- OR SEGNAME FOR EKYEPR1A'
000BED C4C1E3C140D6C640        1761               =C'DATA OF SEG2 NOT PROVIDED BY DL/I CAPTURE'
000C16 E4D5C5E7D7C5C3E3        1762               =C'UNEXPECTED CALL FUNCTION IN DL/I XPCB'
000C3B 40E2C5C7D5C1D4C5        1763               =CL09' SEGNAME='
000C44 40D7D9D6D7C1C7C1        1764               =C' PROPAGATING SQL-'
000C55 40C6D6D940E3C1C2        1765               =C' FOR TABLE='
000C60 E2D8D340C5D9D9D6        1766               =C'SQL ERROR CODE=-NNN'
                               1768 ***********************************************************************
                               1769 *        DESCRIPTION OF GETMAINED AREA CONTAINING AMONG OTHER:        *
                               1770 *             - SAVEAREA                                              *
                               1771 *             - EXIT WORKSPACE                                        *
                               1772 *             - AN SQL WORKAREA (SQLDSECT)                            *
                               1773 *             - A CALL PARAMETER LIST USED FOR CALLS TO THE           *
                               1774 *               THE DPROP TRACER                                      *
                               1775 *             - A TRACE REQUEST BLOCK (TRB)                           *
                               1776 *             - 10 TRACE ELEMENT DESCRIPTORS (TED'S)                  *
                               1777 ***********************************************************************

000000                         1779 GETM     DSECT
                               1780 *---------------------------------------------------------------------*
                               1781 *        REGISTER SAVEAREA                                            *
                               1782 *---------------------------------------------------------------------*
000000                         1783 SAVE     DS    18F'0'              REGISTER SAVEAREA

                               1785 *---------------------------------------------------------------------*
                               1786 *        WORK SPACE FOR EXIT                                          *
                               1787 *---------------------------------------------------------------------*

000048 4040404040404040        1789 OPER     DC    CL8' '              TYPE OF SQL OPERATION
000050 0000000000000000        1790 DBLW     DC    D'0'                DOUBLE WORD USED AS WORK

                               1792 *---------------------------------------------------------------------*
                               1793 *        SPACE FOR THE SQL WORK AREA                                  *
                               1794 *---------------------------------------------------------------------*
000058                         1795          DS    0D
000058                         1796 WORKSQL  DS    CL(SQLDLEN)         RESERVE LENGTH OF SQL DSECT

                               1798 *---------------------------------------------------------------------*
                               1799 *        PARAMETER LIST TO CALL THE DPROP TRACER                      *
                               1800 *---------------------------------------------------------------------*
```

Figure 52 (Part 28 of 40). First Sample Propagation Exit Routine (Assembler)

```
0000E8                        1802          DS    0F
                       000E8  1803 TRAPARML EQU   *                TRACE PARAMETER LIST
0000E8 00000000               1804 TRATRB   DC    A(0)             A(TRACE REQUEST BLOCK)
0000EC 00000000               1805 TRATED1  DC    A(0)             A(1ST TRACE ELEMENT DESCRIPTOR)
0000F0 00000000               1806 TRATED2  DC    A(0)             A(2ND TRACE ELEMENT DESCRIPTOR)
0000F4 00000000               1807 TRATED3  DC    A(0)             A(3RD TRACE ELEMENT DESCRIPTOR)
0000F8 00000000               1808 TRATED4  DC    A(0)             A(4TH TRACE ELEMENT DESCRIPTOR)
0000FC 00000000               1809 TRATED5  DC    A(0)             A(5TH TRACE ELEMENT DESCRIPTOR)
000100 00000000               1810 TRATED6  DC    A(0)             A(6TH TRACE ELEMENT DESCRIPTOR)
000104 00000000               1811 TRATED7  DC    A(0)             A(7TH TRACE ELEMENT DESCRIPTOR)
000108 00000000               1812 TRATED8  DC    A(0)             A(8TH TRACE ELEMENT DESCRIPTOR)
00010C 00000000               1813 TRATED9  DC    A(0)             A(9NT TRACE ELEMENT DESCRIPTOR)
000110 00000000               1814 TRATED10 DC    A(0)             A(10TH TRACE ELEMENT DESCRIPTOR)

                              1816 *-----------------------------------------------------------------*
                              1817 *        SPACE FOR ONE TRACE REQUEST BLOCK (TRB)                   *
                              1818 *-----------------------------------------------------------------*

000118                        1820          DS    0D
000118                        1821 WORKTRB  DS    CL(TRBLEN)

                              1823 *-----------------------------------------------------------------*
                              1824 *        SPACE FOR 10 DIFFERENT TRACE ELEMENT DESCRIPTORS (TED'S)    *
                              1825 *-----------------------------------------------------------------*

000160                        1827          DS    0D
000160                        1828 WORKTED  DS    10CL(TEDLEN)     SPACE FOR 10 TED'S

                              1830 *-----------------------------------------------------------------*
                              1831 *        SPACE FOR A TRACE HEADER                                  *
                              1832 *-----------------------------------------------------------------*

0002C8                        1834 TRHEADER DS    0CL53            TRACE HEADER
0002C8                        1835 TRHP0    DS    CL8' '           NAME OF MODULE CREATING TRACE
0002D0                        1836 TRHP1    DS    CL17             =C' PROPAGATING SQL-'
0002E1                        1837 TRHP2    DS    CL8' '           SQL OPERATION
0002E9                        1838 TRHP3    DS    CL11             =C' FOR TABLE='
0002F4                        1839 TRHP4    DS    CL18' '          TABLE NAME

                              1841 *-----------------------------------------------------------------*
                              1842 *        SPACE FOR A TRACE SUBHEADER FOR SQL CODE                  *
                              1843 *-----------------------------------------------------------------*

000306 E2D8D340C5D9D9D6       1845 TXTSQLC  DC    C'SQL ERROR CODE=-NNN' TEXT OF TRACE SUBHEADER
000319                 00315  1846          ORG   *-4
000315 40                     1847 TXTSQLCS DC    C' '             SIGN OF SQL CODE
000316 404040                 1848 TXTSQLCC DC    CL3' '           SQL CODE

                       00319  1850 GETML    EQU   *-GETM           LENGTH OF GETMAINED AREA
                              1852 *******************************************************************
                              1853 *        DESCRIPTION OF DL/I SEGMENTS AND OF FULLY CONCATENATED KEY *
                              1854 *******************************************************************

                              1856 *-----------------------------------------------------------------*
                              1857 *        DESCRIPTION OF THE DL/I SEGMENT 'SEG2', WHICH IS          *
                              1858 *        PROPAGATED BY THE EXIT TO THE TARGET TABLE 'TAB2'         *
                              1859 *-----------------------------------------------------------------*

000000                        1861 SEG2     DSECT
000000 4040                   1862 SEG2KEY1 DC    CL2' '           1ST KEY SUBFIELD OF SEG2
000002 404040404040           1863 SEG2KEY2 DC    CL6' '           2ND KEY SUBFIELD OF SEG2
000008 4040404040404040       1864 SEG2DAT1 DC    CL8' '           A DATA FIELD OF SEG2
000010 4040404040404040       1865 SEG2DAT2 DC    CL8' '           A DATA FIELD OF SEG2
```

Figure 52 (Part 29 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                  1867 *-------------------------------------------------------------------*
                                  1868 *        DESCRIPTION OF THE PARENT SEGMENT 'SEG1' OF 'SEG2'.         *
                                  1869 *        THE FIELD SEG1DAT1 IS 'PATH DATA' WHICH NEEDS TO            *
                                  1870 *        BE PROPAGATED TOGETHER WITH SEG2 DATA TO THE TARGET         *
                                  1871 *        DB2 TABLE 'TAB2'.                                            *
                                  1872 *-------------------------------------------------------------------*

000000                            1874 SEG1     DSECT ,
000000 4040404040                 1875 SEG1KEY1 DC    CL5' '             KEY FIELD OF SEG1
000005 40404040404040             1876 SEG1DAT1 DC    CL7' '             A DATA FIELD OF SEG1
00000C 40404040                   1877 SEG1DAT2 DC    CL4' '             A DATA FIELD OF SEG1
000010 4040404040404040           1878 SEG1DAT3 DC    CL8' '             A DATA FIELD OF SEG1

                                  1880 *-------------------------------------------------------------------*
                                  1881 *        DESCRIPTION OF THE FULLY CONCATENATED KEY                    *
                                  1882 *        OF THE DL/I SEGMENT 'SEG2'.                                  *
                                  1883 *-------------------------------------------------------------------*

000000                            1885 FCKEY    DSECT
000000 4040404040                 1886 FCK_SEG1KEY1 DC CL5' '            KEY FIELD OF SEG1
000005 4040                       1887 FCK_SEG2KEY1 DC CL2' '            1ST KEY SUBFIELD OF SEG2
000007 404040404040               1888 FCK_SEG2KEY2 DC CL6' '            2ND KEY SUBFIELD OF SEG2


                                  1890 ********************************************************************
                                  1891 *        DESCRIPTION/DECLARATION OF THE 'TAB2' TABLE               *
                                  1892 ********************************************************************

                                  1894 ***$$$
                                  1895 *        EXEC  SQL DECLARE TAB2 TABLE                                 C
                                                     (TAB2COL1  CHAR(5)  NOT NULL  ,                         C
                                                      TAB2COL2  CHAR(2)  NOT NULL  ,                         C
                                                      TAB2COL3  CHAR(6)  NOT NULL  ,                         C
                                                      TAB2COL4  CHAR(8)  NOT NULL  ,                         C
                                                      TAB2COL5  CHAR(8)  NOT NULL  ,                         C
                                                      TAB2COL6  CHAR(7)  NOT NULL  )
                                  1896 ***$$$
                                  1898          EKYRCPIC ,               EXIT INTERFACE CONTROL BLOCK
                                  1899+****************** START OF CONTROL BLOCK SPECIFICATION **************/
                                  1900+*                                                                   */
                                  1901+*        CONTROL BLOCK NAME:                                        */
                                  1902+*          EKYRCPIC (PIC)                                           */
                                  1903+*                                                                   */
                                  1904+*        DESCRIPTIVE NAME:                                          */
                                  1905+*          DPROP PROPAGATION EXIT INTERFACE BLOCK                   */
                                  1906+*                                                                   */
                                  1907+*                                                                   */
                                  1908+*********************************************************************
                                  1909+*                                                                   *
                                  1910+*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".     *
                                  1911+*                                                                   *
                                  1912+*          5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.             *
                                  1913+*          ALL RIGHTS RESERVED.                                     *
                                  1914+*                                                                   *
                                  1915+*          U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                *
                                  1916+*          USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY            *
                                  1917+*          GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                 *
                                  1918+*                                                                   *
                                  1919+*          LICENSED MATERIALS - PROPERTY OF IBM.                    *
                                  1920+*                                                                   *
                                  1921+*********************************************************************
                                  1922+*                                                                   */
                                  1923+*        STATUS: V1 R2 M0                                           */
                                  1924+*                                                                   */
```

Figure 52 (Part 30 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                  1925+*       FUNCTION:                                                   */
                                  1926+*         THIS IS THE CONTROL BLOCK USED TO INTERFACE BETWEEN       */
                                  1927+*              - DPROP                                              */
                                  1928+*              AND                                                  */
                                  1929+*              - A USER'S PROPAGATION  EXIT ROUTINE                 */
                                  1930+*                                                                   */
                                  1931+*         THERE IS ONE PIC CB FOR EACH EXIT PROPAGATION             */
                                  1932+*         EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT        */
                                  1933+*         IN VIRTUAL STORAGE.                                       */
                                  1934+*         FOR SYNCH PROPAGATION IN MPP REGIONS:                     */
                                  1935+*         - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER      */
                                  1936+*           SUBTASK.                                                */
                                  1937+*         FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR           */
                                  1938+*         ASYNCH PROPAGATION, AND FOR CCU PROCESSING:               */
                                  1939+*         - THIS IS THE DURATION OF THE JOBSTEP.                    */
                                  1940+*                                                                   */
                                  1941+*       MODULE TYPE= MACRO                                          */
                                  1942+*          PROCESSOR= ASSEMBLER H                                   */
                                  1943+*                                                                   */
                                  1944+*       INNER CONTROL BLOCKS: NONE                                  */
                                  1945+*                                                                   */
                                  1946+*       MACROS USED FROM MACRO LIBRARY: NONE                        */
                                  1947+*                                                                   */
                                  1948+*       CHANGE ACTIVITY:                                            */
                                  1949+*                     KMP0057  12/13/90                             */
                                  1950+*                     KMP0060  02/08/91 COPYRIGHT INFORMATION       */
                                  1951+*                                                                   */
                                  1952+******************** END OF CONTROL BLOCK SPECIFICATION *************/

000000                            1954+PIC      DSECT
                                  1955+*-------------------------------------------------------------*
                                  1956+*             THIS SECTION CONTAINS INFORMATION PROVIDED BY     *
                                  1957+*             DPROP TO THE INVOKED EXIT AT ENTRY TO CALL. THIS  *
                                  1958+*             SECTION MUST NOT BE MODIFIED BY THE EXIT.         *
                                  1959+*-------------------------------------------------------------*

000000 C5D2E8D9C3D7C9C3           1961+PICEYE   DC    CL8'EKYRCPIC'      EYE CATCHER
000008 4040404040404040           1962+PICEXIT  DC    CL8' '             NAME OF THE EXIT ROUTINE
000010 4040                       1963+PICCALL  DC    CL2' '             TYPE OF CALL TO EXIT
                                  1964+*                                 ...'HR': HIERARCH TO RELATIONAL PROP
                                  1965+*                                 ...'RH': REL. TO HIERARCH
000012 00                         1966+PICDBLEV DC    X'00'              DEBUG LEVEL IN EFFECT
                            00002 1967+PICDBLV2 EQU   X'02'                 2 : EXTERNAL TRACE OF PROPAGATING
                                  1968+*                                        SQL STATEMENTS AND DL/I CALLS
000013 00                         1969+         DC    X'00'              RESERVED
000014 00000000                   1970+PICPTD   DC    A(0)               A(DPROP PTD)
000018 4040404040404040           1971+PICPRID  DC    CL8' '             PR-ID
000020 4040404040404040           1972+PICPRSET DC    CL8' '             PRSET-ID
000028 4040404040404040           1973+PICPRTST DC    CL26' '            PR TIMESTAMP
000042 0000                       1974+         DC    XL2'00'            RESERVED
000044 4040404040404040           1975+PICPCBLA DC    CL8' '             PCB LABEL AS SPECIFIED ON PR
00004C 0000000000000000           1976+         DC    XL56'00'           RESERVED
000084 40404040                   1977+PICOPSYS DC    CL4' '             OPERATING SYSTEM
                                  1978+*                                 ...'ESA ': MVS/ESA
000088 40404040                   1979+PICTRANS DC    CL4' '             IMS REGION TYPE
                                  1980+*                                 ...'MPP ': MPP REGION
                                  1981+*                                 ...'IFP ': IMS FAST PATH REGION
                                  1982+*                                 ...'BMP ': IMS BMP REGION
                                  1983+*                                 ...'BAT ': IMS BATCH REGION
                                  1984+*                                 ...'    ': IF NONE OF ABOVE
00008C 40404040                   1985+PICPROGM DC    CL4' '             CALLING PROGRAM
                                  1986+*                                 ...'DPRS': DPROP SYNCH PROPAGATION
                                  1987+*                                 ...'DPRA': DPROP ASYNCH PROPAGATION
000090 0000000000000000           1988+         DC    XL12'00'           RESERVED FOR DPROP
```

Figure 52 (Part 31 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                 1990+*-----------------------------------------------------------------*
                                 1991+*             THIS SECTION IS USED BY THE EXIT TO PROVIDE       *
                                 1992+*             INFORMATION TO DPROP                              *
                                 1993+*-----------------------------------------------------------------*

00009C 40                        1995+PICENTRD DC    CL1' '                 SET BY EXIT ROUTINE TO
                                 1996+*                                     C'X', INDICATES
                                 1997+*                                     THAT EXIT HAS BEEN ENTERED
                                 1998+*
00009D 40                        1999+PICINCTL DC    CL1' '                 SET BY EXIT ROUTINE TO
                                 2000+*                                     C'X', INDICATES
                                 2001+*                                     THAT EXIT IS IN CONTROL

                                 2003+*********
                                 2004+*********     RETURN CODE AND ERROR MESSAGE
                                 2005+*********

00009E 0000                      2007+PICXRETC DC    H'0'               RETURN CODE
                                 2008+*                                 ...4: SQL ERROR
                                 2009+*                                       SQL ERROR CODE IS IN THE FIELD
                                 2010+*                                       SQLCODE OF THE SQLCA
                                 2011+*                                 ...8: DLI ERROR
                                 2012+*                                       AIBRETRN, AIBREASN AND
                                 2013+*                                       DLI STATUS CODE IN PCB
                                 2014+*                                       POINTED BY AIBRSA1
                                 2015+*                                 ..12: ERROR OTHER THAN SQL ERROR:
                                 2016+*                                       SOME RESOURCES NOT AVAILABLE
                                 2017+*                                 ..16: ERROR OTHER THAN SQL ERROR:
                                 2018+*                                       NOT A RESOURCE AVAILABILITY
                                 2019+*                                       PROBLEM.
                                 2020+*                                 ..20: SHOULD NOT OCCUR/SHOULD ABEND
                                 2021+*
0000A0                           2022+PICXMESG DS    0CL280             USER EXIT ERROR/WARNING MESSAGE
                                 2023+*                                     DPROP WILL WRITE THE MESSAGE
                                 2024+*                                     TO VARIOUS DESTINATIONS ACCORDING
                                 2025+*                                     TO USUAL DPROP/RUP ERROR HANDLING
                                 2026+*                                     LOGIC.
0000A0                           2027+PICXML1  DS    0CL70' '           1ST MESSAGE LINE
0000A0                           2028+PICXMSGI DS    CL8' '             ...8 BYTES MESSAGE ID
0000A8                           2029+PICXMSGB DS    C' '               ...ONE BLANK
0000A9                           2030+PICXMTXT DS    CL61' '            ...61 TEXT BYTES IN 1ST MESSAGE LINE
0000E6                           2031+PICXML2  DS    CL70' '            2ND MESSAGE LINE
00012C                           2032+PICXML3  DS    CL70' '            3RD MESSAGE LINE
000172                           2033+PICXML4  DS    CL70' '            4TH MESSAGE LINE
                                 2034+*
0001B8 0000000000000000          2035+         DC    XL12'00'           RESERVED FOR DPROP

                                 2037+*********
                                 2038+*********     NAME OF OBJECTS ASSOCIATED WITH ERROR
                                 2039+*********

0001C4 4040404040404040          2041+PICDBN   DC    CL8' '             DBDNAME ASSOCIATED WITH THE ERROR
0001CC 4040404040404040          2042+PICSEGN  DC    CL8' '             SEG NAME ASSOCIATED WITH THE ERROR
0001D4 4040404040404040          2043+PICTABQ  DC    CL8' '             TABLE NAME QUALIFIER ASSOC. W. ERROR
0001DC 4040404040404040          2044+PICTABN  DC    CL18' '            TABLE NAME ASSOCIATED WITH THE ERROR
0001EE 0000000000000000          2045+         DC    XL14'00'           RESERVED FOR DPROP
```

Figure 52 (Part 32 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                   2047+*-----------------------------------------------------------------*
                                   2048+*            EXIT WORK AREA                                        *
                                   2049+*                                                                  *
                                   2050+*            THE EXIT WORK AREA CAN BE USED TO SAVE                *
                                   2051+*            INFORMATION ACROSS CALLS TO THE EXIT (E.G.            *
                                   2052+*            TO SAVE THE ADDRESSES OF GETMAINED AREAS ACROSS       *
                                   2053+*            CALLS TO THE EXIT.                                    *
                                   2054+*-----------------------------------------------------------------*
000200                             2056+         DS    0D
000200 0000000000000000            2057+PICSWORK DC    XL256'00'          WORK AREA FOR THE EXIT
000300 0000000000000000            2058+         DC    XL16'00'           RESERVED FOR DPROP

                                   2060+*-----------------------------------------------------------------*
                                   2061+*            SQL COMMUNICATION AREA (SQLCA).                       *
                                   2062+*                                                                  *
                                   2063+*            THE EXIT SHOULD USE THIS SQLCA FOR ITS SQL            *
                                   2064+*            STATEMENTS.                                           *
                                   2065+*-----------------------------------------------------------------*
000310                             2067+SQLCA           DS 0D
000310                             2068+SQLCID          DS CL8            ID
000318                             2069+SQLCABC         DS F              BYTE COUNT
00031C                             2070+SQLCODE         DS F              RETURN CODE
000320                             2071+SQLERRM         DS H,CL70         ERROR MSG PARMS
000368                             2072+SQLERRP         DS CL8            IMPL DEPENDENT
000370                             2073+SQLERRD         DS 6F
000388                             2074+SQLWARN         DS 0C             WARNING FLAGS
000388                             2075+SQLWARN0        DS C'W'           IF ANY
000389                             2076+SQLWARN1        DS C'W'           = WARNING
00038A                             2077+SQLWARN2        DS C'W'           = WARNING
00038B                             2078+SQLWARN3        DS C'W'           = WARNING
00038C                             2079+SQLWARN4        DS C'W'           = WARNING
00038D                             2080+SQLWARN5        DS C'W'           = WARNING
00038E                             2081+SQLWARN6        DS C'W'           = WARNING
00038F                             2082+SQLWARN7        DS C'W'           = WARNING
000390                             2083+SQLEXT          DS CL8
000398                             2084+                DS 4F             RESERVED

                                   2086+*--------------------------------------------------------------------*
                                   2087+*            DLI APPLICATION INTERFACE BLOCK (AIB)                    *
                                   2088+*                                                                     *
                                   2089+*            THE EXIT SHOULD USE THIS AIB FOR ITS DLI                 *
                                   2090+*            CALL. BEFORE FIRST CALL, DPROP INITS                     *
                                   2091+*            AIBID, AIBLEN, AIBRSNM1 AND AIBSFUNC FIELDS.             *
                                   2092+*                                                                     *
                                   2093+*--------------------------------------------------------------------*
0003A8                             2095+PICAIB   DS    0D                 AIB INITIALIZED BY DPROP
0003A8                             2096+PIC_AIBID      DS    CL8'DFSAIB'  EYECATCHER
0003B0                             2097+PIC_AIBLEN     DS    F            DFSAIB ALLOCATED LENGTH
0003B4                             2098+PIC_AIBSFUNC   DS    CL8          SUBFUNCTION CODE
0003BC                             2099+PIC_AIBRSNM1   DS    CL8          RESOURCE NAME 1
0003C4                             2100+PIC_AIBRSNM2   DS    CL8          RESOURCE NAME 2
0003CC                             2101+               DS    2F           RESERVED
0003D4                             2102+PIC_AIBOALEN   DS    F            OUTPUT AREA LENGTH (MAX)
0003D8                             2103+PIC_AIBOAUSE   DS    F            OUTPUT AREA LENGTH (USED)
0003DC                             2104+               DS    2F           RESERVED
0003E4                             2105+               DS    H            RESERVED
0003E6                             2106+               DS    H            RESERVED
```

Figure 52 (Part 33 of 40). First Sample Propagation Exit Routine (Assembler)

```
0003E8                            2107+PIC_AIBRETRN   DS    F           RETURN CODE
0003EC                            2108+PIC_AIBREASN   DS    F           REASON CODE
0003F0                            2109+               DS    F           RESERVED
0003F4                            2110+PIC_AIBRSA1    DS    A           RESOURCE ADDRESS 1
0003F8                            2111+PIC_AIBRSA2    DS    A           RESOURCE ADDRESS 2
0003FC                            2112+PIC_AIBRSA3    DS    A           RESOURCE ADDRESS 3
000400                            2113+               DS    10F         RESERVED
                           00080  2114+PIC_AIBLL      EQU   *-PICAIB    DFSAIB LENGTH
000428                            2115+                  DS 4F          RESERVED

                           00438  2117+PICEND   EQU   *                 END OF PIC
                           00438  2118+PICLEN   EQU   *-PIC             LENGTH OF PIC


                                  2120 ***********************************************************************
                                  2121 *       REDEFINITIONS OF THE MESSAGE AREA LOCATED IN THE PIC        *
                                  2122 ***********************************************************************

000438                     000A0  2124          ORG   PICXML1
0000A0 4040404040404040           2125 MSGSID   DC    CL8' '
0000A8 40                         2126 MSGSBL1  DC    C' '              ONE BLANK
0000A9 4040404040404040           2127 MSGSTXT  DC    CL30' '           =C'PROPAGATION FAILURE FOR TABLE='
0000C7 4040404040404040           2128 MSGSTABLE DC   CL18' '           TABLE NAME

0000D9                     000E6  2130          ORG   PICXML2
0000E6 4040404040404040           2131 MSGSTXT2 DC    CL22' '           =C'FAILING SQL STATEMENT='
0000FC 4040404040404040           2132 MSGSTXTO DC    CL8' '            TYPE OF SQL STATEMENT
000104 4040404040404040           2133 MSGSTXT3 DC    CL16' '           =C' SQL ERROR CODE='
000114 40                         2134 MSGSSQLCS DC   CL1' '            SIGN OF SQL ERROR CODE
000115 404040                     2135 MSGSSQLC DC    CL3' '            SQL ERROR CODE


000118                     000A0  2137          ORG   PICXML1
0000A0 4040404040404040           2138 MSGOID   DC    CL8' '
0000A8 40                         2139 MSGOBL1  DC    C' '              ONE BLANK
0000A9 4040404040404040           2140 MSGOTXT  DC    CL61' '           TEXT

0000E6                     000E6  2142          ORG   PICXML2
0000E6 4040404040404040           2143 MSGOTXT2 DC    CL08' '           =C'DBDNAME='
0000EE 4040404040404040           2144 MSGODBD  DC    CL8' '            DBDNAME
0000F6 4040404040404040           2145 MSGOTXT3 DC    CL09' '           =C' SEGNAME='
0000FF 4040404040404040           2146 MSGOSEG  DC    CL8' '            SEGNAME
000107 404040404040               2147 MSGOTXT4 DC    CL06' '           =C' FUNC='
00010D 40404040                   2148 MSGOFUNC DC    CL04' '           CALL FUNCTION
                                  2150          EKYRCDL1 ,              DL/I CAPTURE INTERFACE CB'S

                                  2152+**********************************************************************
                                  2153+*                                                                    *
                                  2154+*        E X T E N D E D   D A T A  B A S E  P C B  -- X P C B        *
                                  2155+*                                                                    *
                                  2156+**********************************************************************

000000                            2158+XPCB      DSECT
000000                            2159+XPCBEYE   DS    CL4     "XPCB" EYECATCHER
000004                            2160+XPCBVER   DS    CL2     XPCB VERSION INDICATOR
000006                            2161+XPCBREL   DS    CL2     XPCB RELEASE INDICATOR
000008                            2162+XPCBEXIT  DS    CL8     SEGMENT USER EXIT NAME
000010                            2163+XPCBRC    DS    H       RETURN-CODE
000012                            2164+XPCBRSNC  DS    H       REASON-CODE
000014                            2165+XPCBDBD   DS    CL8     PHYSICAL DATA BASE NAME
00001C                            2166+XPCBVERA  DS    A       ADDRESS OF DBD VERSION ID
000020                            2167+XPCBSEG   DS    CL8     PHYSICAL SEGMENT NAME
000028                            2168+XPCBCALL  DS    CL4     'CALL FUNCTION' DEFINED BY IMS/ESA
```

Figure 52 (Part 34 of 40). First Sample Propagation Exit Routine (Assembler)

```
                         2169+*                    ISRT: INSERT
                         2170+*                    REPL: REPLACE
                         2171+*                    DLET: DELETE
                         2172+*                    CASC: CASCADING DELETE
                         2173+*                    DLLP: NOW ALSO DELETED FROM LOGICAL PATH
00002C                   2174+XPCBPCALL DS    CL4     'PHYSICAL UPDATE TYPE' DEFINED BY IMS
                         2175+*                    ISRT: INSERT
                         2176+*                    REIN: RE-INSERT VIA LOGICAL PATH
                         2177+*                    REPL: REPLACE
                         2178+*                    DLET: DELETE
                         2179+*                    DLPP: DELETED ONLY FROM PHYSICAL PATH
000030                   2180+          DS    CL4     RESERVED
000034                   2181+XPCBPCBA  DS    A       ADDRESS OF DB PCB
000038                   2182+XPCBPCBN  DS    CL8     NAME OF DB PCB
000040                   2183+XPCBINQA  DS    A       ADDRESS OF "INQY" OUTPUT
000044                   2184+XPCBIOPA  DS    A       ADDRESS OF I/O PCB
000048                   2185+          DS    H       RESERVED
00004A                   2186+XPCBCKEYL DS    H       LENGTH OF CONCATENATED KEY
00004C                   2187+XPCBCKEYA DS    A       ADDRESS OF CONCATENATED KEY
000050                   2188+XPCBXSDBD DS    A       ADDRESS OF XSDB FOR DATA
000054                   2189+XPCBXSDBB DS    A       ADDRESS OF XSDB FOR REPL DATA
000058                   2190+XPCBXSDBP DS    A       ADDRESS OF XSDB FOR PATH DATA
00005C                   2191+          DS    F       RESERVED
000060                   2192+          DS    F       RESERVED
000064                   2193+          DS    F       RESERVED
000068                   2194+XPCBEXIWP DS    A       ADDRESS OF 256-BYTE AREA RESERVED FOR EXIT
00006C                   2195+          DS    F       RESERVED
000070                   2196+          DS    F       RESERVED
000074                   2197+XPCBTIMST DS    CL8     TIMESTAMP OF CALL
00007C                   2198+          DS    F       RESERVED
                  00080  2199+XPCBLEN   EQU   *-XPCB  LENGTH OF XPCB


                         2201+*********************************************************************
                         2202+*                                                                   *
                         2203+*         E X T E N D E D   S E G M E N T   D A T A   -- X S D B    *
                         2204+*                                                                   *
                         2205+*********************************************************************

000000                   2207+XSDB      DSECT
000000                   2208+XSDBEYE   DS    CL4     "XSDB" EYECATCHER
000004                   2209+XSDBVER   DS    CL2     XSDB VERSION INDICATOR
000006                   2210+XSDBREL   DS    CL2     XSDB RELEASE INDICATOR
000008                   2211+XSDBNXSDB DS    A       NEXT XSDB POINTER
00000C                   2212+XSDBDBD   DS    CL8     PHYSICAL DATA BASE NAME
000014                   2213+XSDBSEG   DS    CL8     PHYSICAL SEGMENT NAME
00001C                   2214+XSDBPHP   DS    CL1     PHYSICAL PATH ACCESSIBILITY
                  000E8  2215+XSDBPHPY  EQU   C'Y'    ...SEGM ACCESSIBLE VIA PHYSICAL PATH
                  000D5  2216+XSDBPHPN  EQU   C'N'    ...SEGM NOT ACCESSIBLE VIA PH. PATH
00001D                   2217+          DS    CL3     RESERVED
000020                   2218+XSDBSEGLV DS    H       SEGMENT DATA BASE LEVEL
000022                   2219+XSDBKEYL  DS    H       LENGTH OF PHYSICAL KEY
000024                   2220+XSDBKEYA  DS    A       ADDRESS OF PHYSICAL KEY
000028                   2221+XSDBFIL1  DS    H       RESERVED
00002A                   2222+XSDBSEGL  DS    H       LENGTH OF SEGMENT DATA
00002C                   2223+XSDBSEGA  DS    A       ADDRESS OF SEGMENT DATA
000030                   2224+XSDBFIL2  DS    F       RESERVED
000034                   2225+XSDBFIL3  DS    F       RESERVED
000038                   2226+XSDBFIL4  DS    F       RESERVED
                  0003C  2227+XSDBLEN   EQU   *-XSDB  LENGTH OF XSDB


                         2229+*********************************************************************
                         2230+*                                                                   *
                         2231+*        D A T A   B A S E   P C B                                  *
                         2232+*                                                                   *
                         2233+*********************************************************************
```

Figure 52 (Part 35 of 40). First Sample Propagation Exit Routine (Assembler)

```
000000                    2235+DBPCB    DSECT
000000                    2236+DBPCBDBD DS    CL8           DBD NAME
000008                    2237+DBPCBLEV DS    CL2           LEVEL FEEDBACK
00000A                    2238+DBPCBSTC DS    CL2           STATUS CODES (RETURNED TO USER)
00000C                    2239+DBPCBPRO DS    CL4           PROCESSING OPTIONS

000010                    2241+DBPCBPFX DS    F             PREFIX ADDRESS
000014                    2242+DBPCBSFD DS    CL8           SEGMENT NAME FEEDBACK
00001C                    2243+DBPCBMKL DS    F             CURRENT LENGTH OF KEY FEEDBACK AREA
                          2244+*                            OR GSAM FEEDBACK AREA
000020                    2245+DBPCBNSS DS    F             NO OF SENSITIVE SEGMENTS IN PCB
                    00024 2246+DBPCBSZ2 EQU   *-DBPCB       SIZE OF PCB WITHOUT KEY FEEDBACK AREA
000024                    2247+DBPCBKFD DS    0CL256        KEY FEEDBACK AREA


                          2249+***********************************************************************
                          2250+*                                                                     *
                          2251+*      INQUIRY (INQY) CALL OUTPUT                                     *
                          2252+*                                                                     *
                          2253+*      THE INQY CALL RETURNS DATA TO THE USER'S I/O AREA BASED        *
                          2254+*      ON THE SUBFUNCTION SPECIFIED IN THE AIB.                       *
                          2255+*                                                                     *
                          2256+*      THE FOLLOWING SUBFUNCTIONS RETURN DATA TO THE APPLICATION:    *
                          2257+*         'ENVIRON'  -  SYSTEM ENVIRONMENT DATA                       *
                          2258+*         'NULL'     -  DATA ASSOCIATED WITH THE PCB NAME             *
                          2259+*                       THAT WAS PASSED IN THE AIB                    *
                          2260+*                                                                     *
                          2261+***********************************************************************

                          2263+*---------------------------------------------------------------------*
                          2264+*                                                                     *
                          2265+*      -----------------------                                        *
                          2266+*      SUBFUNCTION = 'ENVIRON'                                        *
                          2267+*      -----------------------                                        *
                          2268+*                                                                     *
                          2269+*---------------------------------------------------------------------*

000000                    2271+INQENVRN DSECT
000000                    2272+INQEIMID DS    CL8                IMS IDENTIFIER
000008                    2273+INQEIMRL DS    F                  IMS RELEASE LEVEL
                          2274+*
                          2275+***    CONTROL REGION TYPES:
                          2276+***       'BATCH   ' - BATCH DATABASE MANAGER
                          2277+***       'DB      ' - ONLINE DATABASE MANAGER SUBSYSTEM
                          2278+***       'DB/DC   ' - ONLINE DB AND DC MANAGER SUBSYSTEM
                          2279+*
00000C                    2280+INQECRT  DS    CL8                CONTROL REGION TYPE
                          2281+*
                          2282+***    APPLICATION REGION TYPES:
                          2283+***       'BATCH   ' - BATCH REGION
                          2284+***       'BMP     ' - BATCH MESSAGE PROCESSING REGION
                          2285+***       'DRA     ' - DATABASE RESOURCE ADAPTER THREAD
                          2286+***       'IFP     ' - FAST PATH REGION
                          2287+***       'MPP     ' - MESSAGE PROCESSING REGION
                          2288+*
000014                    2289+INQEART  DS    CL8                APPLICATION REGION TYPE
00001C                    2290+INQEARID DS    F                  APPLICATION RGN IDENTIFIER
000020                    2291+INQEPGM  DS    CL8                APPLICATION PROGRAM NAME
000028                    2292+INQEPSB  DS    CL8                ALLOCATED PSB NAME
000030                    2293+INQETRAN DS    CL8                TRANSACTION NAME
000038                    2294+INQEUSER DS    CL8                USER IDENTIFIER
000040                    2295+INQEGPNM DS    CL8                GROUP NAME
                          2296+*
```

Figure 52 (Part 36 of 40). First Sample Propagation Exit Routine (Assembler)

```
                                  2297+***     STATUS GROUP INDICATOR:
                                  2298+***        '    ' - NO STATUS GROUP WAS INITIALIZED
                                  2299+***        'A   ' - INIT STATUS GROUPA CALL WAS ISSUED
                                  2300+***        'B   ' - INIT STATUS GROUPB CALL WAS ISSUED
                                  2301+*
000048                            2302+INQESGID DS    CL4               HIGHEST STATUS GROUP ID
00004C                            2303+INQERTA  DS    A                 ADDRESS OF RECOVERY TOKEN
                                  2304+*                                STRING MAPPED BY INQERTS
000050                            2305+INQEAPA  DS    A                 ADDRESS OF APPLICATION PARM
                                  2306+*                                STRING MAPPED BY INQEAPS
                           00054  2307+INQELEN  EQU   *-INQENVRN        ENVIRON OUTPUT LENGTH
000054                            2308+         DS    CL120             SPACE FOR UOW-ID AND APPL-PARMS
                           000CC  2309+INQELEN2 EQU   *-INQENVRN        IOAREA_LENGTH FOR INQY DL1 CALL

                                  2311+*-------------------------------------------------------------*
                                  2312+*                                                             *
                                  2313+*       RECOVERY TOKEN STRING DSECT                           *
                                  2314+*                                                             *
                                  2315+*-------------------------------------------------------------*
000000                            2316+INQERTS  DSECT
000000                            2317+INQERTLL DS    H                 RECOVERY TOKEN LENGTH
                           00002  2318+INQERTKN EQU   *                 START OF RECOVERY TOKEN

                                  2320+*-------------------------------------------------------------*
                                  2321+*                                                             *
                                  2322+*       APPLICATION PARAMETER STRING DSECT                    *
                                  2323+*                                                             *
                                  2324+*-------------------------------------------------------------*
000000                            2325+INQEAPS  DSECT
000000                            2326+INQEAPLL DS    H                 APPL PARM STRING LENGTH
                           00002  2327+INQEAPRM EQU   *                 START OF APPL PARM STRING


                                  2329+***************************************************************
                                  2330+*                                                             *
                                  2331+*       DFSAIB DSECT - APPLICATION INTERFACE BLOCK            *
                                  2332+*                                                             *
                                  2333+*       THE DFSAIB IS THE APPLICATION INTERFACE BLOCK PASSED  *
                                  2334+*       TO IMS ON APPLICATION CALLS WHICH USE THE DFSAIBLI    *
                                  2335+*       LANGUAGE INTERFACE ENTRY POINT.  APPLICATIONS WHICH   *
                                  2336+*       USE THIS ENTRY POINT ARE EITHER ISSUING CALLS USING   *
                                  2337+*       A PCB NAME INSTEAD OF A PCB ADDRESS, OR ARE ISSUING   *
                                  2338+*       CALLS WHICH ARE NOT ASSOCIATED WITH A PCB.            *
                                  2339+*       THE DFSAIB PROVIDES A STANDARD MECHANISM FOR IMS AND  *
                                  2340+*       AND THE APPLICATION TO EXCHANGE INFORMATION.          *
                                  2341+*                                                             *
                                  2342+*       THE DFSAIB IS ALLOCATED AND INITIALIZED BY THE        *
                                  2343+*       APPLICATION PROGRAM.  INDIVIDUAL DL/I CALLS MAY HAVE  *
                                  2344+*       DIFFERENT REQUIREMENTS FOR REQUIRED INPUT FIELDS.     *
                                  2345+*       AT A MINIMUM, THE FOLLOWING FIELDS MUT BE INITIALIZED *
                                  2346+*       PRIOR TO ISSUING ANY DL/I CALL.                       *
                                  2347+*         AIBID    = CHARACTER STRING 'DFSAIB  '             *
                                  2348+*         AIBLEN   = LENGTH USED BY THE APPLICATION TO ALLOCATE *
                                  2349+*                    THE STORAGE AREA.                        *
                                  2350+*         AIBOALEN = LENGTH OF APPLICATION I/O AREA           *
                                  2351+*                    (ONLY REQUIRED ON DL/I CALLS IN WHICH IMS *
                                  2352+*                    WILL RETURN DATA IN THE I/O AREA)        *
                                  2353+*                                                             *
                                  2354+*       IMS WILL RETURN A RETURN CODE TO THE APPLICATION      *
                                  2355+*       IN THE DFSAIB.  ADDITIONALLY, OTHER INFORMATION SUCH  *
                                  2356+*       AS A REASON CODE, MAY BE RETURNED AS REQUIRED BY      *
                                  2357+*       SPECIFIC CALLS.                                       *
                                  2358+*                                                             *
                                  2359+***************************************************************
```

Figure 52 (Part 37 of 40). First Sample Propagation Exit Routine (Assembler)

```
000000                       2361+DFSAIB   DSECT
000000                       2362+AIBID    DS    CL8'DFSAIB'         EYECATCHER
000008                       2363+AIBLEN   DS    F                   DFSAIB ALLOCATED LENGTH
00000C                       2364+AIBSFUNC DS    CL8                 SUBFUNCTION CODE
000014                       2365+AIBRSNM1 DS    CL8                 RESOURCE NAME 1
00001C                       2366+AIBRSNM2 DS    CL8                 RESOURCE NAME 2
000024                       2367+         DS    2F                  RESERVED
00002C                       2368+AIBOALEN DS    F                   OUTPUT AREA LENGTH (MAX)
000030                       2369+AIBOAUSE DS    F                   OUTPUT AREA LENGTH (USED)
000034                       2370+         DS    2F                  RESERVED
00003C                       2371+         DS    H                   RESERVED
00003E                       2372+         DS    H                   RESERVED
000040                       2373+AIBRETRN DS    F                   RETURN CODE
000044                       2374+AIBREASN DS    F                   REASON CODE
000048                       2375+         DS    F                   RESERVED
00004C                       2376+AIBRSA1  DS    A                   RESOURCE ADDRESS 1
000050                       2377+AIBRSA2  DS    A                   RESOURCE ADDRESS 2
000054                       2378+AIBRSA3  DS    A                   RESOURCE ADDRESS 3
000058                       2379+         DS    10F                 RESERVED
                       00080 2380+AIBLL    EQU   *-DFSAIB            DFSAIB LENGTH
                             2382          EKYTRB   ,                TRACE REQUEST BLOCK
                             2383+****************** START OF CONTROL BLOCK SPECIFICATION ***************
                             2384++*                                                                    *
                             2385++*       CONTROL BLOCK NAME:                                          *
                             2386++*         EKYTRB (TRB)                                               *
                             2387++*                                                                    *
                             2388++*       DESCRIPTIVE NAME:                                            *
                             2389++*         DPROP TRACE REQUEST BLOCK (TRB)                            *
                             2390++*               =     =       =                                      *
                             2391++*                                                                    *
                             2392+**********************************************************************
                             2393++*                                                                    *
                             2394++*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".      *
                             2395++*                                                                    *
                             2396++*          5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.              *
                             2397++*          ALL RIGHTS RESERVED.                                      *
                             2398++*                                                                    *
                             2399++*          U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                 *
                             2400++*          USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY             *
                             2401++*          GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                  *
                             2402++*                                                                    *
                             2403++*          LICENSED MATERIALS - PROPERTY OF IBM.                     *
                             2404++*                                                                    *
                             2405+**********************************************************************
                             2406++*                                                                    *
                             2407++*       STATUS: V1 R2 M0                                             *
                             2408++*                                                                    *
                             2409++*       FUNCTION:                                                    *
                             2410++*         A TRB IS USED FOR THE COMMUNICATION BETWEEN A              *
                             2411++*         'PROPAGATION USER EXIT ROUTINE' AND THE DPROP TRACE        *
                             2412++*         FUNCTION.                                                  *
                             2413++*                                                                    *
                             2414++*         WHEN INVOKING THE DPROP-TRACE FUNCTION, THE CALLING        *
                             2415++*         USER EXIT MUST PROVIDE THE TRB AS FIRST CALL-PARAMETER.    *
                             2416++*                                                                    *
                             2417++*         THE TRB PROVIDES INFORMATION ABOUT THE TRACE REQUEST.      *
                             2418++*                                                                    *
                             2419++*       MODULE TYPE= MACRO                                           *
                             2420++*          PROCESSOR= ASSEMBLER H                                    *
                             2421++*                                                                    *
                             2422++*       ACQUIRED BY MODULE INVOKING THE TRACE                        *
                             2423++*                                                                    *
                             2424++*       INNER CONTROL BLOCKS: NONE                                   *
                             2425++*                                                                    *
                             2426++*       MACROS USED FROM MACRO LIBRARY: NONE                         *
                             2427++*                                                                    *
```

Figure 52 (Part 38 of 40). First Sample Propagation Exit Routine (Assembler)

```
                              2428+*        CHANGE ACTIVITY:                                           *
                              2429+*                       KMP0057  12/13/90                           *
                              2430+*                                                                   *
                              2431+******************** END OF CONTROL BLOCK SPECIFICATION ***************


000000                        2435+TRB      DSECT
000000 E3D9C240               2436+TRBEYE   DC   C'TRB '       EYE-CATCHER
000004 00000000               2437+TRBPTD   DC   A(0)          ADDRESS OF THE DPROP-PTD CONTROL BLOCK

                              2439+********
                              2440+********                    NAME OF OBJECTS ASSOCIATED WITH THE TRACE
                              2441+********

000008 4040404040404040       2443+TRBTABQ  DC    CL8' '       TABLE-NAME QUALIFIER ASSOC. W. TRACE
000010 4040404040404040       2444+TRBTABN  DC    CL18' '      TABLE-NAME ASSOCIATED WITH THE TRACE
000022 4040                   2445+         DC    CL2' '
000024 4040404040404040       2446+TRBDBN   DC    CL8' '       DBD-NAME ASSOCIATED WITH THE TRACE
00002C 4040404040404040       2447+TRBSEGN  DC    CL8' '       SEG-NAME ASSOCIATED WITH THE TRACE

                              2449+********
                              2450+********                    SOLICITED/UNSOLICITED INDICATION
                              2451+********

000034 40                     2453+TRBSOLI  DC   CL1' '        SOLICITED TRACE
                        000E8 2454+TRBSOLY  EQU  C'Y'          ...Y: TRACE SOLICITED BY THE USER
                        000D5 2455+TRBSOLN  EQU  C'N'          ...N: TRACE NOT SOLICITED BY THE USER

000035 0000000000000000       2457+         DC   13X'00'       RESERVED/MUST BE ZERO
                        00042 2458+TRBEND   EQU  *
                        00042 2459+TRBLEN   EQU  *-TRB         LENGTH OF ONE TRB
                              2461          EKYTED   ,                TRACE ELEMENT DESCRIPTION
                              2462+****************** START OF CONTROL BLOCK SPECIFICATION ***************
                              2463+*                                                                   *
                              2464+*        CONTROL BLOCK NAME:                                        *
                              2465+*          EKYTED (TED)                                             *
                              2466+*                                                                   *
                              2467+*        DESCRIPTIVE NAME:                                          *
                              2468+*          DPROP TRACE ELEMENT DESCRIPTOR (TED)                     *
                              2469+*                =     =       =                                    *
                              2470+*                                                                   *
                              2471+*********************************************************************
                              2472+*                                                                   *
                              2473+*           THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".    *
                              2474+*                                                                   *
                              2475+*           5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.            *
                              2476+*           ALL RIGHTS RESERVED.                                    *
                              2477+*                                                                   *
                              2478+*           U.S. GOVERNMENT USERS RESTRICTED RIGHTS -               *
                              2479+*           USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY           *
                              2480+*           GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                *
                              2481+*                                                                   *
                              2482+*           LICENSED MATERIALS - PROPERTY OF IBM.                   *
                              2483+*                                                                   *
                              2484+*********************************************************************
                              2485+*                                                                   *
                              2486+*        STATUS: V1 R2 M0                                           *
                              2487+*                                                                   *
                              2488+*        FUNCTION:                                                  *
                              2489+*          WHEN INVOKING THE DPROP TRACE FUNCTION, THE CALLING      *
                              2490+*          MODULE MUST PROVIDE ONE TED FOR EACH:                    *
                              2491+*           - TRACE-HEADER                                          *
                              2492+*           - TRACE-SUBHEADER                                       *
                              2493+*           - DATA-AREA                                             *
                              2494+*        WHICH SHOULD BE TRACED/SNAPPED.                            *
                              2495+*                                                                   *
```

Figure 52 (Part 39 of 40). First Sample Propagation Exit Routine (Assembler)

```
                              2496+*        MODULE TYPE= MACRO                                        *
                              2497+*           PROCESSOR= ASSEMBLER H                                 *
                              2498+*                                                                  *
                              2499+*        ACQUIRED BY MODULE INVOKING THE TRACE                     *
                              2500+*                                                                  *
                              2501+*        INNER CONTROL BLOCKS: NONE                                *
                              2502+*                                                                  *
                              2503+*        MACROS USED FROM MACRO LIBRARY: NONE                      *
                              2504+*                                                                  *
                              2505+*        CHANGE ACTIVITY:                                          *
                              2506+*                       KMP0057  12/13/90                          *
                              2507+*                                                                  *
                              2508+******************** END OF CONTROL BLOCK SPECIFICATION ***************


000000                        2512+TED      DSECT
000000 E3C5C440               2513+TEDEYE   DC   C'TED '       EYE-CATCHER
000004 40                     2514+TEDTYPE  DC   C' '          TYPE OF TRACE ITEM
                       000C8  2515+TEDTYPH  EQU  C'H'          ... HEADER
                       000E2  2516+TEDTYPS  EQU  C'S'          ... SUB-HEADER
                       000C4  2517+TEDTYPD  EQU  C'D'          ... DATA
000005 40                     2518+TEDALIGN DC   C' '          ALIGNMENT FOR SNAP-FORMATTING
                       000D3  2519+TEDALIGL EQU  C'L'          ...L    = LEFT ALIGNMENT
                       00040  2520+TEDALIGB EQU  C' '          ...BLANK= NO LEFT ALIGNMENT
000006 0000                   2521+         DC   XL2'00'       RESERVED
000008 00000000               2522+TEDTXTA  DC   A(0)          PTR TO TEXT-STRING
00000C 00000000               2523+TEDTXTL  DC   F'0'          LENGTH OF TEXT-STRING
000010 00000000               2524+TEDMA    DC   A(0)          VIRTUAL STORAGE ADDR OF AREA TO BE SNAPPED
000014 00000000               2525+TEDALEN  DC   F'0'          LENGTH OF AREA TO BE SNAPPED
000018 00000000               2526+TEDALET  DC   F'0'          ALET OF DATA (MUST BE ZERO IN THIS RELEASE)
00001C 0000000000000000       2527+         DC   2F'0'         RESERVED/MUST BE ZERO
                       00024  2528+TEDEND   EQU  *
                       00024  2529+TEDLEN   EQU  *-TED         LENGTH OF ONE TED
000C73                        2531 EKYEPR1A CSECT ,                   REQUIRED BECAUSE OF DB2 PRECOMPILER
                              2532 ***$$$ SQL WORKING STORAGE
000C73 00
000C74 00000090               2533 SQLDSIZ  DC    A(SQLDLEN) SQLDSECT SIZE
000000                        2534 SQLDSECT DSECT
000000                        2535 SQLPLIST DS    F
000004                        2536 SQLPLLEN DS    H        PLIST LENGTH
000006                        2537 SQLFLAGS DS    XL2      FLAGS
000008                        2538 SQLCTYPE DS    H        CALL-TYPE
00000A                        2539 SQLPROGN DS    CL8      PROGRAM NAME
000012                        2540 SQLTIMES DS    CL8      TIMESTAMP
00001A                        2541 SQLSECTN DS    H        SECTION
00001C                        2542 SQLCODEP DS    A        CODE POINTER
000020                        2543 SQLVPARM DS    A        VPARAM POINTER
000024                        2544 SQLAPARM DS    A        AUX PARAM PTR
000028                        2545 SQLSTNUM DS    H        STATEMENT NUMBER
00002A                        2546 SQLSTYPE DS    H        STATEMENT TYPE
00002C                        2547 SQLPVARS DS    F,6CL12
000078                        2548 SQLAVARS DS    F,0CL12
00007C                        2549 SQLTEMP  DS    CL18     TEMPLATE
000090                        2550          DS    0D
                       00090  2551 SQLDLEN  EQU   *-SQLDSECT
000000                        2552          END   EKYEPR1A
```

Figure 52 (Part 40 of 40). First Sample Propagation Exit Routine (Assembler)

Definitions For First Sample Propagation Exit

This section contains definitions associated with the first sample Propagation exit routine. It includes the following types of definitions:

- IMS DBDGEN and PSBGEN definitions
- DB2 CREATE TABLE definitions
- DataRefresher definitions required to define the PR DataRefresher and to extract the IMS data with DataRefresher
- SQL statements defining the PR without DataRefresher in the MVG input tables

DBDGEN Definitions

Figure 53 shows a DBDGEN definition for the sample Propagation exit routine in Figure 52 on page 190.

```
         DBD NAME=DB1,VERSION=V123456789,                              C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),               C
               EXIT=(EKYRUP00,KEY,PATH,DATA)
         DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=SEG1,PARENT=0,BYTES=24
         FIELD  NAME=(SEG1KEY,SEQ,U),BYTES=5,START=1
*
         SEGM  NAME=SEG2,PARENT=SEG1,BYTES=24
         FIELD  NAME=(SEG2KEY,SEQ,U),BYTES=8,START=1
*
         DBDGEN
         FINISH
         END
```

Figure 53. DBDGEN Definition

Notes:

1. The EXIT= keyword of the DBD macro specifies that EKYRUP00 (the RUP) be called when a segment of this DBD is changed. This is required for synchronous data propagation with DPROP.
2. The EXIT= keyword of the DBD statement requests the PATH data option. This is required for the mapping performed by this sample Propagation exit routine (because the Propagation exit routine maps nonkey, path data, from the parent segment).

CREATE TABLE Statement

Figure 54 on page 231 shows a CREATE TABLE statement for the sample Propagation exit routine in Figure 52 on page 190.

```
  CREATE TABLE T096606.TAB2
    (TAB2COL1    CHAR(5)     NOT NULL,
     TAB2COL2    CHAR(2)     NOT NULL,
     TAB2COL3    CHAR(6)     NOT NULL,
     TAB2COL4    CHAR(8)             ,
     TAB2COL5    CHAR(8)             ,
     TAB2COL6    CHAR(8)             ,
     PRIMARY KEY (TAB2COL1, TAB2COL2, TAB2COL3))
   IN DU096606.PROPTS;


CREATE UNIQUE INDEX XN01 ON TAB2 (TAB2COL1, TAB2COL2, TAB2COL3)
       USING VCAT KOE ;
```

Figure 54. CREATE TABLE Statement

Using DataRefresher to Define the PR

This section describes how you can use DataRefresher to define the PR for the sample Propagation exit routine in Figure 52 on page 190.

CREATE DXTPSB

Figure 55 shows a CREATE DXTPSB statement for the sample Propagation exit routine in Figure 52 on page 190.

```
 CREATE DXTPSB   NAME=KOEPSB

   DXTPCB   NAME=DB1, DBNAME=DB1, DBACCESS=HDAM

   SEGMENT  NAME=SEG1, PARENT=0, BYTES=24

       FIELD    NAME=SEG1KEY1, START=1 , BYTES=5, SEQFLD=R
       FIELD    NAME=SEG1DAT1, START=6 , BYTES=7, TYPE=C
       FIELD    NAME=SEG1DAT2, START=13, BYTES=4, TYPE=C
       FIELD    NAME=SEG1DAT3, START=17, BYTES=8, TYPE=C

   SEGMENT  NAME=SEG2, PARENT=SEG1, BYTES=24

       FIELD    NAME=SEG2KEY , START=1 , BYTES=8, SEQFLD=R
       FIELD    NAME=SEG2KEY1, START=1 , BYTES=2, TYPE=C
       FIELD    NAME=SEG2KEY2, START=3 , BYTES=6, TYPE=C
       FIELD    NAME=SEG2DAT1, START=9 , BYTES=8, TYPE=C
       FIELD    NAME=SEG2DAT2, START=17, BYTES=8, TYPE=C ;
```

Figure 55. CREATE DXTPSB Statement

The Propagation exit routine does not map the key field of segment SEG2 to one DB2 column. Instead, the key field of SEG2 is mapped as two key subfields to two columns of the DB2 primary key. Therefore, the key field SEG2KEY is redefined by the two key subfields SEG2KEY1 and SEG2KEY2 that overlay SEG2KEY.

CREATE DXTVIEW

Figure 56 on page 232 shows a CREATE DXTVIEW statement for the sample Propagation exit routine in Figure 52 on page 190.

```
CREATE   DXTVIEW NAME     = VIEW011,
                 DXTPSB   = KOEPSB,
                 DXTPCB   = DB1,
                 SEGMENT  = SEG2,
                 MINSEGM  = SEG2,
                 FIELDS   = *     ;
```

Figure 56. CREATE DXTVIEW Statement

DataRefresher UIM SUBMIT Command and EXTRACT Statement

Figure 57 shows a DataRefresher UIM SUBMIT command and EXTRACT statement for the Propagation exit routine in Figure 52 on page 190.

```
SUBMIT   EXTID=PR001,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=U,
                         MAPDIR=HR,
                         ACTION=REPL,
                         ERROPT=BACKOUT,
                         EXITNAME=EKYEPR1A,
                         PROPSEGM=(DB1/SEG2)'

EXTRACT
    INTO T096606.TAB2    (TAB2COL1   NOT NULL,
                          TAB2COL2   NOT NULL,
                          TAB2COL3   NOT NULL,
                          TAB2COL4           ,
                          TAB2COL5           ,
                          TAB2COL6           )
                  SELECT  SEG1KEY1,
                          SEG2KEY1,
                          SEG2KEY2,
                          SEG2DAT1,
                          SEG2DAT2,
                          SEG1DAT1
                FROM VIEW011 ;
```

Figure 57. DataRefresher UIM SUBMIT Command and EXTRACT Statement

Notes:

1. The MAPEXIT= keyword of the SUBMIT command specifies EKYMCE00. This causes DataRefresher UIM to call the DPROP-provided Map Capture Exit EKYMCE00 during the processing of the SUBMIT or EXTRACT. This is required to allow DPROP to create the PR.
2. PRTYPE=U (user mapping) must be specified, because the PR must be processed by a Propagation exit routine.
3. EXITNAME=EKYEPR1A specifies the name of the Propagation exit routine that performs the propagation for this PR.
4. PROPSEGM=DB1 or SEG2 identifies the segment types being propagated by this PR. As explained in the commentary for the source code of the EKYEPR1A, the sample exit routine propagates changes to the data of SEG2 (together with path data of SEG1). However, the sample exit routine does not propagate changes to the data of SEG1. Therefore, the PROPSEGM= keyword identifies only SEG2 as the segment being propagated.

5. The EXTRACT statement describes to DataRefresher which fields must be mapped to which columns during the data extract. These definitions are important for the extract but are not important for DPROP (because the mapping and propagation is not done by the generalized mapping logic of DPROP).

Using DataRefresher For the Extract

This section covers INITDEM and USE DXTPSB Control Statements. Figure 58 shows INITDEM and USE DXTPSB control statements for the Propagation exit routine in Figure 52 on page 190.

```
INITDEM NAME=BASILEUS;
USE DXTPSB=KOEPSB;
```

Figure 58. Using DataRefresher For the Extract: INITDEM and USE DXTPSB Control Statements

Defining the PR in the MVG Input Tables

Figure 59 on page 234 describes DSNTEP2 SQL statements required to define the PR in the MVG input tables.

The following rows are inserted into the MVG input tables:

- One row is inserted into the DPRIPR table (the PR table).

  This row identifies the PRID, indicates that the PRTYPE is U (user mapping), and provides in the EXITNAME column the name of the Propagation exit routine EKYEPR1A that performs the propagation for this PR.

- One row for each segment type being propagated by the PR and the Propagation exit routine is inserted into the DPRISEG table (the SEG table).

  As explained in the commentary of the source code of EKYEPR1A, the sample exit routine propagates changes to the data of SEG2 (together with path data of SEG1). However, the sample exit routine does not propagate changes to the data of SEG1. Therefore, only one row is inserted into the DPRISEG table, a row indicating that the PR is propagating SEG2.

- One row is inserted into the DPRITAB table (the TAB table).

  This row indicates that the target table is T096606.TAB2.

For PRTYPE=U, DPROP does not require that you insert any rows in the DPRIFLD table; this is why the example below does not insert any row in the DPRITAB table.

```
DELETE FROM T096606.DPRIPR WHERE PRID = 'PR001'            ;


INSERT INTO T096606.DPRIPR
             ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR,
               ERROPT,   ACTION,   EXITNAME)
      VALUES ('PR001',  'T096606','U',    ' ',      'HR',
              'BACKOUT','REPL',   'EKYEPR1A')             ;

INSERT INTO T096606.DPRISEG
             ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR001','DB1',  'SEG2',    ' ' )           ;

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL,   TABNAME )
      VALUES ('PR001','T096606', 'TAB2')                   ;


COMMIT;
```

Figure 59. DSNTEP2 SQL Statements Required to Define the PR in the MVG Input Tables

Second Sample Propagation Exit Routine

A second example of a Propagation exit routine written in an HLL is shown in Figure 62 on page 236.

This is a key range splitting example: the mapping is provided from two different segment types of two different databases. Both segments have the same structure and the same key construction, but each key is unique over both databases.

The first database contains the lower key range (000000 to 499999), and the second one contains the higher key range (500000 to 999999).

Each segment occurrence is mapped to a specific row of the propagated table.

Mapping Performed by the Sample Exit Routine

Figure 60 illustrates an overview of the propagation done by the sample Propagation exit routine.

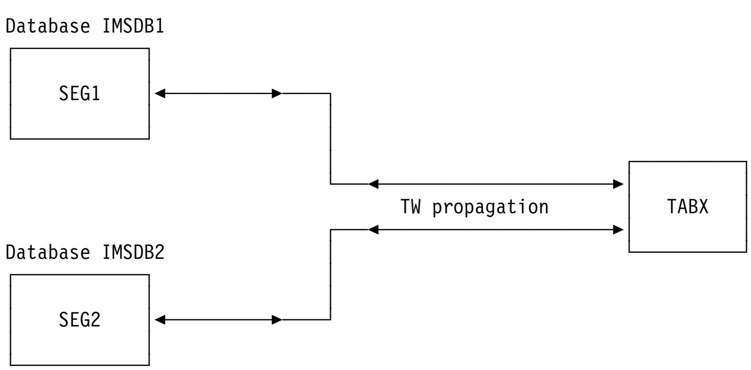

Figure 60. Overview of the Propagation Performed By the Exit Routine

Figure 61 shows the mapping of individual IMS source fields to the DB2 target columns and vice versa.

Figure 61. Mapping IMS Source Fields to DB2 Target Columns

Segment Name	Field Name	Key attribute	Column Name	Column Type
SEG1	SEG1KEY1	Key field	TABXCOL1	DB2 Primary key
SEG1	SEG1DAT1	-	TABXCOL2	-
SEG1	SEG1DAT2	-	TABXCOL3	-
SEG1	SEG1DAT3	-	TABXCOL4	-
SEG2	SEG2KEY	Key field	TABXCOL1	DB2 Primary Key
SEG2	SEG2DAT1	-	TABXCOL2	-
SEG2	SEG2DAT2	-	TABXCOL3	-
SEG2	SEG2DAT3	-	TABXCOL4	-

Sample Exit Routine Source Code

The example in Figure 62 on page 236 is intentionally simplified to emphasize the fundamental logic involved. Your Propagation exit routine will likely be more complex to meet your propagation requirements.

The source code below is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYEPR2K. The following source code shows sample module EKYEPR2K after the DB2 precompiler processed it.

Following the source code are definitions related to the sample Propagation exit routine.

```
/*******************************************************************
 *                                                                 *
 *       Licensed Materials - Property of IBM                      *
 *                                                                 *
 *       5685-124 (C) Copyright IBM Corp. 1989, 1992.              *
 *                                                                 *
 *       See Copyright Instructions                                *
 *                                                                 *
 *******************************************************************

********************************************************************
*                                                                  *
* Module name: EKYEPR2K                                            *
*                                                                  *
* Descriptive name: Sample C Language Propagation User Exit Routine. *
*                                                                  *
*                                                                  *
* Function:                                                        *
*                                                                  *
*    The purpose of this program is to provide a sample propagation  *
*    exit routine.  This is a key range splitting example, e.g. the  *
*    mapping is provided from two different segment types of two     *
*    different databases.  Both segments have the same structure and *
*    the same key construction, but each key content is unique over  *
*    both databases.                                               *
*                                                                  *
*    The first database contains the lower key range i.e.          *
*       - "000000" to "499999".                                    *
*                                                                  *
*    The second one contains the higher key range i.e.             *
*       - "500000" to "999999".                                    *
*                                                                  *
*    Each segment occurrence is mapped to one row of the propagated  *
*    table.                                                        *
*                                                                  *
********************************************************************
#pragma page(1)
********************************************************************
*                                                                  *
*                                                                  *
* The figure below provides an overview of the IMS-to-DB2 mapping  *
* performed by this sample propagation exit.                      *
*                                                                  *
*    *---------------------*         *-----------------------*       *
*    |     IMS world       |         |      DB2 world        |       *
*    *---------------------          *-----------------------*       *
*                                                                  *
*    *---------------------*                                       *
*    |  database "IMSDB1"  |                                       *
*    |  segment  "SEG1"    |                                       *
*    *---------------------*                                       *
*    | seg1key  key field  |<--+                                   *
*    | seg1dat1            |<---+                                  *
*    | seg1dat2            |<----+    *-----------------------*      *
*    | seg1dat3            |<-----+   |   table "TABX"        |      *
*    *---------------------*   ||||   *-----------------------*      *
*                              +----->| TABXCOL1 primary key |       *
*    *---------------------*   |+---->| TABXCOL2             |       *
*    |  database "IMSDB2"  |   ||+--->| TABXCOL3             |       *
*    |  segment  "SEG2"    |   |||+-->| TABXCOL4             |       *
*    *---------------------*   ||||   |                      |       *
```

Figure 62 (Part 1 of 18). Second Sample Propagation Exit Routine (C)

```
*    | SEG2KEY  key field |<--+|||   |                     |       *
*    | SEG2DAT1           |<---+||   *---------------------*       *
*    | SEG2DAT2           |<----+|                                 *
*    | SEG2DAT3           |<-----+                                 *
*    *--------------------*                                        *
*                                                                  *
*                                                                  *
*                                                                  *
********************************************************************
#pragma page(1)
********************************************************************
*                                                                  *
* Return code =  0  processing successful - no message set.        *
*                                                                  *
* Return code =  4  SQL error: error while propagating from IMS to *
*                   DB2.                                           *
*                                                                  *
* Return code =  8  IMS error: error while propagating from DB2 to *
*                   IMS.                                           *
*                                                                  *
* Return code = 12  error other than SQL and IMS, unavailable      *
*                   problem.                                       *
*                                                                  *
* Return code = 16  error other than SQL and IMS, not an unavailable *
*                   resource problem.                              *
*                                                                  *
* Return code = 20  severe error: abend is required.               *
*                                                                  *
*                                                                  *
*                                                                  *
* Error messages issued by EKYEPR2P:                               *
*                                                                  *
*    EKYEPR1E propagation failure for table=@ failing SQL          *
*             statement=@  SQL code=@                              *
*    EKYEPR2E propagation failure for segment=@ failing IMS        *
*             segment=@                                            *
*    EKYEPR3E invalid propagation direction in PICCALL             *
*    EKYEPR4E IMS-to-DB2: unexpected DBD or segname                *
*    EKYEPR5E IMS-to-DB2: data is missing for a REPL or an ISRT call *
*    EKYEPR6E IMS-to-DB2: unexpected call function in the IMS XPCB *
*    EKYEPR7E IMS-to-DB2: KFBA is missing for a REPL call          *
*    EKYEPR8E DB2-to-IMS: invalid call function in the HEC         *
*    EKYEPR9E DB2-to-IMS: PCB label not found                      *
*                                                                  *
******************** End of Specifications ************************
#pragma page(1)
******************** Logic of EKYEPR2P *****************************
*                                                                  *
* Processing:                                                      *
*                                                                  *
*    - Set "module entered" and "module in control" flags into PIC. *
*                                                                  *
*    - Check function code to see if the module is called to perform *
*      IMS-to-DB2 propagation (HR) or to perform DB2-to-IMS        *
*      propagation (RH).                                           *
*                                                                  *
* Processing for IMS-to-DB2 propagation:                           *
*                                                                  *
*    - Provide addressing for "EKYRCDLP".                          *
*                                                                  *
```

Figure 62 (Part 2 of 18). Second Sample Propagation Exit Routine (C)

```
*    - Set table qualifier and table name into PIC.                  *
*                                                                    *
*    - Verify information provided by DL/I capture and/or DPROP.     *
*                                                                    *
*    - Verify that the exit is invoked to propagate the right        *
*      DBD/segname.                                                  *
*                                                                    *
*    - For ISRT and REPL operations:                                 *
*         Verify that DL/I capture provides the segment data.        *
*                                                                    *
*    - For DLET operation:                                           *
*         Verify that DL/I capture provides the KFBA.                *
*                                                                    *
*    - Branch according to type of IMS update operation:             *
*                                                                    *
*    - For an IMS REPL:                                              *
*         Issue a SQL update statement for a row with columns        *
*         originating from SEG1 or SEG2.                             *
*                                                                    *
*         If the SQL update results in an error or warning execute   *
*         the SQL error logic.                                       *
*                                                                    *
*    - For an IMS ISRT:                                              *
*         Issue a SQL insert statement to insert a row with columns  *
*         originating from SEG1 or SEG2.                             *
*                                                                    *
*         If the SQL insert results in an error or warning execute   *
*         the SQL error logic.                                       *
*                                                                    *
*    - For an IMS DLET:                                              *
*         Issue a SQL delete statement to delete the target row.     *
*                                                                    *
*         If the SQL delete results in an error or a warning execute *
*         the SQL error logic.                                       *
**********************************************************************
#pragma page(1)
**********************************************************************
*                                                                    *
*  SQL error logic:                                                  *
*                                                                    *
*    - set return code of 4                                          *
*    - copy the SQLCA used in this module to the "PIC" SQLCA         *
*    - format an error message                                       *
*    - return to the caller.                                         *
*                                                                    *
*                                                                    *
*  Processing for DB2-to-IMS propagation:                            *
*                                                                    *
*  - provide addressing for "EKYHCHCP" and other appropriate control *
*    blocks                                                          *
*                                                                    *
*  - get the column data and move it to the IMS segment work area    *
*                                                                    *
*  - build the SSA, init the AIB and set the correct function code   *
*    for the DL/I call                                               *
*                                                                    *
*                                                                    *
*  - perform the following depending on the DB2 operation:           *
*                                                                    *
```

Figure 62 (Part 3 of 18). Second Sample Propagation Exit Routine (C)

```
 *    - for an INSERT call:                                             *
 *                                                                      *
 *        - issue an IMS insert with the IMS segment work area          *
 *                                                                      *
 *        - if the IMS insert results in an error or warning            *
 *             build the error message and set an 8 - return code       *
 *                                                                      *
 *        - return to the caller                                        *
 *                                                                      *
 *                                                                      *
 *    - for an UPDATE or DELETE call:                                   *
 *                                                                      *
 *        - issue an IMS get hold unique                                *
 *                                                                      *
 *        - if the GHU results in an error or warning                   *
 *             build the error message and set an 8 - return code       *
 *                                                                      *
 *        - else issue an IMS REPL or DLET depending on the             *
 *          SQL operation                                               *
 *                                                                      *
 *        - if the IMS call results in an error or warning              *
 *             build the error message and set an 8 - return code       *
 *                                                                      *
 *        - return to the caller                                        *
 ***********************************************************************
#pragma page(1)
 ***********************************************************************
 *                                                                      *
 *  - Errors other than SQL errors:                                     *
 *                                                                      *
 *    - set return code of 4                                            *
 *    - build an error message in the PIC                               *
 *    - return to caller of the exit                                    *
 *                                                                      *
 ******************** End of Logic Description ************************/
#pragma page(1)
#include <leawi.h>
#include <ims.h>
#pragma linkage(ekyepr2k,fetchable)
#include <stdlib.h>
#include <string.h>
#pragma page(1)
/**********************************************************************
 *                   Propagated DB2 table                               *
 **********************************************************************/

#pragma linkage (DSNHLI,OS)
typedef struct
              { short   SQLPLLEN;
                short   SQLFLAGS;
                short   SQLCTYPE;
                char    SQLPROGN[8];
                short   SQLTIMES[4];
                short   SQLSECTN;
                char   *SQLCODEP;
                char   *SQLVPARM;
                char   *SQLAPARM;
                short   SQLSTNUM;
                short   SQLSTYPE;
              } SQLPLIST;
```

Figure 62 (Part 4 of 18). Second Sample Propagation Exit Routine (C)

```
typedef struct
              { short    SQLTYPE;
                short    SQLLEN;
                char    *SQLADDR;
                char    *SQLIND;
              } SQLELTS;
typedef SQLELTS     *SQLELTS_PTR;
char    SQLTEMP[ 19 ] ;

/***$$$
 EXEC SQL DECLARE TABX TABLE
             (TABXCOL1   CHAR(6) NOT NULL,
              TABXCOL2   CHAR(7) NOT NULL,
              TABXCOL3   CHAR(4) NOT NULL,
              TABXCOL4   CHAR(8) NOT NULL)
$$$***/



/***$$$
 EXEC SQL INCLUDE SQLCA
$$$***/
#ifndef SQLCODE
struct sqlca
  { unsigned  char     sqlcaid[8];
              long     sqlcabc;
              long     sqlcode;
              short    sqlerrml;
    unsigned  char     sqlerrmc[70];
    unsigned  char     sqlerrp[8];
              long     sqlerrd[6];
    unsigned  char     sqlwarn[11];
    unsigned  char     sqlstate[5];
            } ;
#define SQLCODE  sqlca.sqlcode
#define SQLWARN0 sqlca.sqlwarn[0]
#define SQLWARN1 sqlca.sqlwarn[1]
#define SQLWARN2 sqlca.sqlwarn[2]
#define SQLWARN3 sqlca.sqlwarn[3]
#define SQLWARN4 sqlca.sqlwarn[4]
#define SQLWARN5 sqlca.sqlwarn[5]
#define SQLWARN6 sqlca.sqlwarn[6]
#define SQLWARN7 sqlca.sqlwarn[7]
#define SQLWARN8 sqlca.sqlwarn[8]
#define SQLWARN9 sqlca.sqlwarn[9]
#define SQLWARNA sqlca.sqlwarn[10]
#define SQLSTATE sqlca.sqlstate
#endif
struct sqlca sqlca;

#pragma page(1)
/***********************************************************************
 *                   Declare Host variables                           *
 ***********************************************************************/


/***$$$
 EXEC SQL BEGIN DECLARE SECTION
$$$***/
```

Figure 62 (Part 5 of 18). Second Sample Propagation Exit Routine (C)

```
 char SEGIKEY[7];
 char SEGIDAT1[8];
 char SEGIDAT2[5];
 char SEGIDAT3[9];


/***$$$
 EXEC SQL END DECLARE SECTION
$$$***/


#pragma page(1)

/**********************************************************************
 *                Include control block structures                    *
 **********************************************************************/

#include "ekyrcpck.h"
#include "ekyrcdlk.h"
#include "ekyhcq2k.h"
#include "ekyhchck.h"

/**********************************************************************
 *                      Prototypes                                    *
 **********************************************************************/

void imstodb2 (EKYRCPIC *,  XPCB *);

void db2toims (EKYRCPIC *,  HEC *);

void segok    (EKYRCPIC *,  XPCB *);

void db2repl  (EKYRCPIC *,  XPCB *,  SEGI *);

void db2isrt  (EKYRCPIC *,  SEGI *);

void db2dlet  (EKYRCPIC *,  XPCB *);

void db2check (EKYRCPIC *);

void sqlerr   (EKYRCPIC *);

void imserr   (EKYRCPIC *);

void invdir   (EKYRCPIC *);

void invseg   (EKYRCPIC *, XPCB *);

void datmis   (EKYRCPIC *);

void invcal   (EKYRCPIC *, XPCB *);

void errcom    (EKYRCPIC *, XPCB *);

void invkfb   (EKYRCPIC *);

void invfun   (EKYRCPIC *);
```

Figure 62 (Part 6 of 18). Second Sample Propagation Exit Routine (C)

```
void lablnf   (EKYRCPIC *);

#pragma page(1)
/*********************************************************************
 *               Declare global variables                            *
 *********************************************************************/

long int  x1, x2, x3, x4, ncount;
char      opcode[6], wsqlcode[6], funccode[4],
          w77ckey[6];

PICXML1 msg11 = {{""},' ',{"Propagation failure for table=\0"}},
        msg21 = {{""},' ',{""}},
        msg31 = {{""},' ',{"Propagation failure for segment=\0"}};

struct {
   char msgstxt2[22];
   char msgstxto[6];
   char msgstxt3[16];
   char msgssqlc[4];
       } msg12 = {{"Failing SQL statement="},
                  {""},{" SQL error code="},{""}};

struct {
   char msgotxt2[8];
   char msgodbd[8];
   char msgotxt3[9];
   char msgoseg[8];
   char msgotxt4[6];
   char msgofunc[4];
       } msg22 = {{"DBDNAME="},
                  {""},{" SEGNAME="},{""},{" FUNC="},{""}};

struct {
   char msgitxt2[22];
   char msgitxto[4];
       } msg32 = {{"Failing IMS statement="},{""}};

char ssaisrt[9];

struct {
   char ssasegnm[8],
        filler_1,
        ssakeynm[8],
        filler_2,
        ssavalue[6],
        filler_3;
       } ssa = {{""},'(',{""},'=',{""},')'};

char segiarea[25];
char segoarea[25];

#pragma page(1)
/*********************************************************************
 *                 Main function - ekyepr2k                          *
 *********************************************************************/
void ekyepr2k ( EKYRCPIC *ekyrcpic, void *secondcb)
{
/* XPCB *xpcb; */
```

Figure 62 (Part 7 of 18). Second Sample Propagation Exit Routine (C)

```
   /* Set control flags - exit entered and in control */
   ekyrcpic->picentrd = 'X';
   ekyrcpic->picinctl = 'X';

/*********************************************************************
 * Check function code to determine if module is called              *
 *    to perform IMS-to-DB2 propagation (HR) or                      *
 *    to perform DB2-to-IMS propagation (RH).                        *
 *********************************************************************/

   if (strncmp(ekyrcpic->piccall, "HR", 2) == 0)
      imstodb2(ekyrcpic, secondcb);
   else

   if (strncmp(ekyrcpic->piccall, "RH", 2) == 0)
      db2toims(ekyrcpic, secondcb);

   else
      invdir(ekyrcpic); /* invalid propagation direction */

   return;

} /* end of ekyepr2k */

#pragma page(1)
/*********************************************************************
 *                R U P   i s   t h e   c a l l e r                  *
 *           M a i n   IMS_to_DB2   p r o c e s s i n g             *
 *********************************************************************/
void imstodb2 (EKYRCPIC *ekyrcpic, XPCB *xpcb)
{
   strncpy(ekyrcpic->pictabq, "        ", 8);
   strncpy(ekyrcpic->pictabn, "TABX              ", 18);

   if (strncmp(xpcb->xpcbdbd, "DIVNTZ02", 8) == 0)
      if (strncmp(xpcb->xpcbseg, "SEG1    ", 8) == 0)
         segok(ekyrcpic, xpcb);
      else
         invseg(ekyrcpic, xpcb); /* unexpected segment name */

   else
   if (strncmp(xpcb->xpcbdbd, "DHVNTZ02", 8) == 0)
      if (strncmp(xpcb->xpcbseg, "SEG2    ", 8) == 0)
         segok(ekyrcpic, xpcb);
      else
         invseg(ekyrcpic, xpcb); /* unexpected segment name */

   else
      invseg(ekyrcpic, xpcb);    /* unexpected DBD */

   return;
} /* end of imstodb2 */

#pragma page(1)
void segok(EKYRCPIC *ekyrcpic, XPCB *xpcb)
{
   XSDB *xsdb = NULL;
   SEGI *segi = NULL;
```

Figure 62 (Part 8 of 18). Second Sample Propagation Exit Routine (C)

```
   if (strncmp(xpcb->xpcbcall, "DLET", 4) == 0)
   {
      strncpy(opcode, "DELETE", 6);
      db2dlet(ekyrcpic,xpcb);
   }

   else
  /*******************************************************************
   * Verify that the segment data is provided for "REPL" and "ISRT".  *
   * Process according to the type of IMS update.                     *
   *******************************************************************/
   {
      if (xpcb->xpcbxsdbd == NULL)
         datmis(ekyrcpic); /* data is missing (EKYEPR5E) */
      else xsdb = xpcb->xpcbxsdbd;

      if (xsdb->xsdbsega == NULL)
         datmis(ekyrcpic); /* data is missing (EKYEPR5E) */
      else segi = xsdb->xsdbsega;

      if (strncmp(xpcb->xpcbcall, "REPL", 4) == 0)
      {
         strncpy(opcode, "UPDATE", 6);
         db2repl(ekyrcpic,xpcb,segi);
      }
      else
      if (strncmp(xpcb->xpcbcall, "ISRT", 4) == 0)
      {
         strncpy(opcode, "INSERT", 6);
         db2isrt(ekyrcpic,segi);
      }
      else
         invcal(ekyrcpic, xpcb);
   }
   return;
} /* end of segok */

#pragma page(1)
/*********************************************************************
 * IMS segment has been replaced.  This results in a propagating SQL  *
 * UPDATE of the target DB2 row.                                      *
 *********************************************************************/
void db2repl(EKYRCPIC *ekyrcpic, XPCB *xpcb, SEGI *segi)
{
   strncpy( SEGIKEY, segi->segikey, 6);
   strncpy( SEGIDAT1, segi->segidat1, 7);
   strncpy( SEGIDAT2, segi->segidat2, 4);
   strncpy( SEGIDAT3, segi->segidat3, 8);


/***$$$
   EXEC SQL UPDATE TABX
      SET   TABXCOL2 = :SEGIDAT1,
            TABXCOL3 = :SEGIDAT2,
            TABXCOL4 = :SEGIDAT3
      WHERE TABXCOL1 = :SEGIKEY
$$$***/
```

Figure 62 (Part 9 of 18). Second Sample Propagation Exit Routine (C)

```
  {
  SQLPLIST SQLPLIST2 =
  {40, -32768, 30, "EKYEPR2K", 0, 0, 0 ,0,
   1, 0, 0, 0, 463, 234};
  SQLELTS_PTR SQLELTS_PTR2;
  struct
     { long  SQLPVARS;
       char  SQLPVELT[(sizeof(SQLELTS) * 4)];
     } SQLPVARS2;
  SQLELTS_PTR2 = (SQLELTS *) &SQLPVARS2.SQLPVELT;
  SQLELTS_PTR2->SQLTYPE = 460;
  SQLELTS_PTR2->SQLLEN  = 8;
  SQLELTS_PTR2->SQLADDR = (char *)
  &(SEGIDAT1);
  SQLELTS_PTR2->SQLIND  = NULL;
  SQLELTS_PTR2 = SQLELTS_PTR2 + 1;
  SQLELTS_PTR2->SQLTYPE = 460;
  SQLELTS_PTR2->SQLLEN  = 5;
  SQLELTS_PTR2->SQLADDR = (char *)
  &(SEGIDAT2);
  SQLELTS_PTR2->SQLIND  = NULL;
  SQLELTS_PTR2 = SQLELTS_PTR2 + 1;
  SQLELTS_PTR2->SQLTYPE = 460;
  SQLELTS_PTR2->SQLLEN  = 9;
  SQLELTS_PTR2->SQLADDR = (char *)
  &(SEGIDAT3);
  SQLELTS_PTR2->SQLIND  = NULL;
  SQLELTS_PTR2 = SQLELTS_PTR2 + 1;
  SQLELTS_PTR2->SQLTYPE = 460;
  SQLELTS_PTR2->SQLLEN  = 7;
  SQLELTS_PTR2->SQLADDR = (char *)
  &(SEGIKEY);
  SQLELTS_PTR2->SQLIND  = NULL;
  SQLPVARS2.SQLPVARS = 52;
  SQLPLIST2.SQLVPARM = (char *) &SQLPVARS2.SQLPVARS;
  SQLPLIST2.SQLCODEP = (char *) &sqlca;
  SQLPLIST2.SQLTIMES[0] = 0x14EA;
  SQLPLIST2.SQLTIMES[1] = 0x9521;
  SQLPLIST2.SQLTIMES[2] = 0x0570;
  SQLPLIST2.SQLTIMES[3] = 0x64A0;
  DSNHLI ( (unsigned int * ) &SQLPLIST2);
  }


   db2check(ekyrcpic);
   return;

} /* end of db2repl */

#pragma page(1)
/**********************************************************************
 * IMS segment has been inserted.  This results in a propagating SQL  *
 * INSERT of the target DB2 row.                                      *
 **********************************************************************/
void db2isrt(EKYRCPIC *ekyrcpic, SEGI *segi)
{
   strncpy( SEGIKEY, segi->segikey, 6);
   strncpy( SEGIDAT1, segi->segidat1, 7);
   strncpy( SEGIDAT2, segi->segidat2, 4);
```

Figure 62 (Part 10 of 18). Second Sample Propagation Exit Routine (C)

```
   strncpy( SEGIDAT3, segi->segidat3, 8);


/***$$$
   EXEC SQL INSERT INTO TABX
           (TABXCOL1,
            TABXCOL2,
            TABXCOL3,
            TABXCOL4)
      VALUES
           (:SEGIKEY,
            :SEGIDAT1,
            :SEGIDAT2,
            :SEGIDAT3)
$$$***/
  {
  SQLPLIST SQLPLIST3 =
  {40, -32768, 30, "EKYEPR2K", 0, 0, 0 ,0,
   2, 0, 0, 0, 486, 232};
  SQLELTS_PTR SQLELTS_PTR3;
  struct
     { long  SQLPVARS;
       char  SQLPVELT[(sizeof(SQLELTS) * 4)];
     } SQLPVARS3;
  SQLELTS_PTR3 = (SQLELTS *) &SQLPVARS3.SQLPVELT;
  SQLELTS_PTR3->SQLTYPE = 460;
  SQLELTS_PTR3->SQLLEN  = 7;
  SQLELTS_PTR3->SQLADDR = (char *)
  &(SEGIKEY);
  SQLELTS_PTR3->SQLIND  = NULL;
  SQLELTS_PTR3 = SQLELTS_PTR3 + 1;
  SQLELTS_PTR3->SQLTYPE = 460;
  SQLELTS_PTR3->SQLLEN  = 8;
  SQLELTS_PTR3->SQLADDR = (char *)
  &(SEGIDAT1);
  SQLELTS_PTR3->SQLIND  = NULL;
  SQLELTS_PTR3 = SQLELTS_PTR3 + 1;
  SQLELTS_PTR3->SQLTYPE = 460;
  SQLELTS_PTR3->SQLLEN  = 5;
  SQLELTS_PTR3->SQLADDR = (char *)
  &(SEGIDAT2);
  SQLELTS_PTR3->SQLIND  = NULL;
  SQLELTS_PTR3 = SQLELTS_PTR3 + 1;
  SQLELTS_PTR3->SQLTYPE = 460;
  SQLELTS_PTR3->SQLLEN  = 9;
  SQLELTS_PTR3->SQLADDR = (char *)
  &(SEGIDAT3);
  SQLELTS_PTR3->SQLIND  = NULL;
  SQLPVARS3.SQLPVARS = 52;
  SQLPLIST3.SQLVPARM = (char *) &SQLPVARS3.SQLPVARS;
  SQLPLIST3.SQLCODEP = (char *) &sqlca;
  SQLPLIST3.SQLTIMES[0] = 0x14EA;
  SQLPLIST3.SQLTIMES[1] = 0x9521;
  SQLPLIST3.SQLTIMES[2] = 0x0570;
  SQLPLIST3.SQLTIMES[3] = 0x64A0;
  DSNHLI ( (unsigned int * ) &SQLPLIST3);
  }


   db2check(ekyrcpic);
   return;
} /* end of db2isrt */
```

Figure 62 (Part 11 of 18). Second Sample Propagation Exit Routine (C)

```
#pragma page(1)
/*********************************************************************
 * IMS segment has been deleted.  This results in a propagating SQL   *
 * DELETE of the target DB2 row.                                      *
 *********************************************************************/
void db2dlet(EKYRCPIC *ekyrcpic, XPCB *xpcb)
{
   if (xpcb->xpcbckeya == NULL)
      invkfb(ekyrcpic); /* IMS-to-DB2: KFBA is missing (EKYEPR7E) */
   else
   {
      strncpy( SEGIKEY, xpcb->xpcbckeya, 6);


/***$$$
      EXEC SQL DELETE FROM TABX
         WHERE TABXCOL1 = :SEGIKEY
$$$***/
  {
  SQLPLIST SQLPLIST4 =
  {40, -32768, 30, "EKYEPR2K", 0, 0, 0 ,0,
   3, 0, 0, 0, 514, 233};
  SQLELTS_PTR SQLELTS_PTR4;
  struct
     { long  SQLPVARS;
       char  SQLPVELT[(sizeof(SQLELTS) * 1)];
     } SQLPVARS4;
  SQLELTS_PTR4 = (SQLELTS *) &SQLPVARS4.SQLPVELT;
  SQLELTS_PTR4->SQLTYPE = 460;
  SQLELTS_PTR4->SQLLEN  = 7;
  SQLELTS_PTR4->SQLADDR = (char *)
  &(SEGIKEY);
  SQLELTS_PTR4->SQLIND  = NULL;
  SQLPVARS4.SQLPVARS = 16;
  SQLPLIST4.SQLVPARM = (char *) &SQLPVARS4.SQLPVARS;
  SQLPLIST4.SQLCODEP = (char *) &sqlca;
  SQLPLIST4.SQLTIMES[0] = 0x14EA;
  SQLPLIST4.SQLTIMES[1] = 0x9521;
  SQLPLIST4.SQLTIMES[2] = 0x0570;
  SQLPLIST4.SQLTIMES[3] = 0x64A0;
  DSNHLI ( (unsigned int * ) &SQLPLIST4);
  }

   }

   db2check(ekyrcpic);
   return;

} /* end of db2dlet */

#pragma page(1)
/*********************************************************************
 * Check SQL error code and SQL warnings.                             *
 *********************************************************************/
void db2check( EKYRCPIC *ekyrcpic)
{
   strncpy (ekyrcpic->picsqlca.sqlcaid, sqlca.sqlcaid, 136);
   if ((SQLCODE != 0) || (SQLWARN0 == 'W')) sqlerr(ekyrcpic);
   return;
} /* end of db2check */
```

Figure 62 (Part 12 of 18). Second Sample Propagation Exit Routine (C)

```
#pragma page(1)
/*********************************************************************
 * Propagation failure for table = TABX.                             *
 *********************************************************************/
void sqlerr(EKYRCPIC *ekyrcpic)
{
   unsigned int i, j;

   ekyrcpic->picxretc = 4;
   strncpy(msg11.picxmsgi, "EKYEPR1E", 8);
   i = strlen(msg11.picxmtxt) + 4;

   strncat(msg11.picxmtxt, "TABX", 4);
   memset(&(msg11.picxmtxt[i]), ' ', 61-i);
   strncpy(msg12.msgstxto, opcode, 6);

   i = abs(SQLCODE);
   for (j = 3; j > 0; j--,i/=10)
      wsqlcode[j] = i%10 + '0';
   wsqlcode[0] = (SQLCODE > 0)? '+':'-';
   strncpy(msg12.msgssqlc, wsqlcode, 4);

   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, msg11.picxmsgi, 70);
   memset(&(ekyrcpic->picxmesg.picxml2), ' ', 70);
   strncpy(ekyrcpic->picxmesg.picxml2, msg12.msgstxt2, 48);

   return;

} /* end of sqlerr */

#pragma page(1)
/*********************************************************************
 *                H U P   i s   t h e   c a l l e r                  *
 *           M a i n   DB2_to_IMS   p r o c e s s i n g             *
 *********************************************************************/
void db2toims(EKYRCPIC *ekyrcpic, HEC *hec)
{
   long THREE = 3;
   long FOUR  = 4;
   int i, j;
   QW0185A *qw0185a;
   QW0185B *qw0185b;

   qw0185a = hec->heccdcdd;
   qw0185b = hec->heccdcda;

  /*******************************************************************
   * Move the contents of the DB2 columns, one by one, to the IMS     *
   * segment work area.                                               *
   *******************************************************************/
   for(x1 = 0; x1 < qw0185a->qw0185ld; x1++)
   {
     /****************************************************************
      * Set x2 to the offset of the column in the dat row.           *
      ****************************************************************/
      x2 = qw0185a->qw0185vr[x1].qw0185sx;
```

Figure 62 (Part 13 of 18). Second Sample Propagation Exit Routine (C)

```
    /*****************************************************************
     * Set output offset depending on the column name.              *
     *****************************************************************/
     if (strncmp(&(qw0185a->qw0185vr[x1].qw0185cn[0]),
                  "TABXCOL1", 8) == 0 )
        x3 = 0;
     else
     if (strncmp(&(qw0185a->qw0185vr[x1].qw0185cn[0]),
                  "TABXCOL2", 8) == 0 )
        x3 = 6;
     else
     if (strncmp(&(qw0185a->qw0185vr[x1].qw0185cn[0]),
                  "TABXCOL3", 8) == 0 )
        x3 = 13;
     else
        x3 = 17;

    /*****************************************************************
     * Move the content of the column, byte by byte.                *
     *****************************************************************/
     for(i = 0; i < qw0185a->qw0185vr[x1].qw0185le; i++)
         segoarea[x3+i] = qw0185b->qw0185dr[x2+i];

  } /* end x1 loop */

#pragma page(1)
 /*******************************************************************
  *          Initialize the AIB and the SSAs                        *
  *******************************************************************/

  memset(&(ekyrcpic->picaib.aibsfunc), ' ', 8);
  memset(&(ekyrcpic->picaib.aibrsnm2), ' ', 8);
  ekyrcpic->picaib.aiboalen =25;
  strncpy(ssa.ssavalue, segoarea, 6);

  if (strncmp(segoarea, "500000", 6) < 0)
  {
     strncpy(ssa.ssasegnm, "SEG1    ", 8);
     strncpy(ssa.ssakeynm, "SEG1KEY ", 8);
  }
  else
  {
     strncpy(ssa.ssasegnm, "SEG2    ", 8);
     strncpy(ssa.ssakeynm, "SEG2KEY ", 8);
  }

#pragma page(1)
 /*******************************************************************
  *          Search the PCB label                                   *
  *******************************************************************/

  for (x4 = 0; x4 < hec->hecdbsln; x4++)
  {
     if (strncmp(ssa.ssasegnm,
                 hec->hecdbsla->hecdslds[x4].hecsegnm,
                 4) == 0)
     {
        strncpy(ekyrcpic->picaib.aibrsnm1,
                hec->hecdbsla->hecdslds[x4].hecpcbnm, 8);
```

Figure 62 (Part 14 of 18). Second Sample Propagation Exit Routine (C)

```
        if (strncmp(qw0185b->qw0185pc, "IN", 2) == 0)
       /*************************************************************
        * IMS segment to be inserted.                               *
        *************************************************************/
        {
           strncpy(funccode, "ISRT", 4);
           strncpy(ssaisrt, ssa.ssasegnm, 8);
           ssaisrt[8] = ' ';
           CEETDLI (&FOUR, funccode,
                    ekyrcpic->picaib, segoarea, ssaisrt);
           if (ekyrcpic->picaib.aibretrn != 0) imserr(ekyrcpic);
        } /* end INSERT call */

#pragma page(1)
        else
        if (strncmp(qw0185b->qw0185pc, "UA", 2) == 0)
       /*************************************************************
        * IMS segment is to be replaced.                            *
        *************************************************************/
        {
           strncpy(funccode, "GHU ", 4);
           CEETDLI (&FOUR, funccode, ekyrcpic->picaib, segiarea, ssa);
           if (ekyrcpic->picaib.aibretrn != 0)
           {
              imserr(ekyrcpic);
              return;
           }
           else
           {
              strncpy(funccode, "REPL", 4);
              CEETDLI (&THREE, funccode, ekyrcpic->picaib, segoarea);
              if (ekyrcpic->picaib.aibretrn != 0)
              {
                 imserr(ekyrcpic);
                 return;
              }
           }
        } /* end UPDATE call */

#pragma page(1)
        else
        if (strncmp(qw0185b->qw0185pc, "DE", 2) == 0)
       /*************************************************************
        * IMS segment is to be deleted.                             *
        *************************************************************/
        {
           strncpy(funccode, "GHU ", 4);
           CEETDLI (&FOUR, funccode, ekyrcpic->picaib, segiarea, ssa);

           if (ekyrcpic->picaib.aibretrn != 0)
           { /* Propagation failure for segment (EKYEPR2E) */
              imserr(ekyrcpic);
              return;
           } /* end AIB return code not equal to zero for GHU call */

           else
           {
              strncpy(funccode, "DLET", 4);
```

Figure 62 (Part 15 of 18). Second Sample Propagation Exit Routine (C)

```
               CEETDLI (&THREE, funccode, ekyrcpic->picaib, segiarea);
               if (ekyrcpic->picaib.aibretrn != 0)
               { /* Propagation failure for segment (EKYEPR2E) */
                  imserr(ekyrcpic);
                  return;
               } /* end AIB return code not zero for DLET call */
            } /* end AIB return code equal to zero for GHU call */

         } /* end DELETE call */

        /************************************************************
         *     Invalid function call                                *
         ************************************************************/
         else invfun(ekyrcpic); /* not INSERT, UPDATE or DELETE call */
         return;

      } /* SSASEGNM matches HECSEGNM */
   } /* end for x4 loop */
   lablnf(ekyrcpic); /* PCB label not found */
   return;

} /* end of db2toims */

#pragma page(1)
/*********************************************************************
 * IMS error.                                                        *
 *********************************************************************/
void imserr(EKYRCPIC *ekyrcpic)
{
   int i;

   ekyrcpic->picxretc = 8;
   strncpy(msg31.picxmsgi, "EKYEPR2E", 8);
   i = strlen(msg31.picxmtxt) + 8;
   strncat( msg31.picxmtxt, ssa.ssasegnm, 8);
   memset( &(msg31.picxmtxt[i]), ' ', 61-i);

   strncpy(msg32.msgitxto, funccode, 4);
   memset(&(ekyrcpic->picxmesg.picxml2), ' ', 70);
   strncpy(ekyrcpic->picxmesg.picxml2, msg32.msgitxt2, 26);

   return;

} /* end of imserr */

#pragma page(1)
/*********************************************************************
 * Invalid propagation direction (found in PICCALL)                  *
 *********************************************************************/
void invdir(EKYRCPIC *ekyrcpic)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR3E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "Invalid propagation direction in PICCALL                     ",
      61);

   return;

} /* end of invdir */
```

Figure 62 (Part 16 of 18). Second Sample Propagation Exit Routine (C)

```
#pragma page(1)
/***********************************************************************
 * IMS to DB2 - unexpected DBD or segment name.                        *
 ***********************************************************************/
void invseg(EKYRCPIC *ekyrcpic, XPCB *xpcb)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR4E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "IMS-to-DB2: Unexpected DBD or SEGNAME for EKYEPR2K           ",
      61);
   errcom(ekyrcpic, xpcb);

   return;

} /* end of invseg */

#pragma page(1)
/***********************************************************************
 * IMS to DB2 - data missing for a REPL or ISRT call.                  *
 ***********************************************************************/
void datmis(EKYRCPIC *ekyrcpic)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR5E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "IMS-to-DB2: Data is missing for a REPL or ISRT call          ",
      61);

   return;

} /* end of datmis */

#pragma page(1)
/***********************************************************************
 * IMS to DB2 - unexpected call function in IMS XPCB.                  *
 ***********************************************************************/
void invcal(EKYRCPIC *ekyrcpic, XPCB *xpcb)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR6E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "IMS-to-DB2: Unexpected call function in IMS XPCB             ",
      61);
   errcom(ekyrcpic, xpcb);

   return;

} /* end of invcal */

#pragma page(1)
/***********************************************************************
 * Additional processing when unexpected DBD or segment encountered    *
 * OR when unexpected call function found in IMS XPCB.                 *
 ***********************************************************************/
```

Figure 62 (Part 17 of 18). Second Sample Propagation Exit Routine (C)

```
void errcom(EKYRCPIC *ekyrcpic, XPCB *xpcb)
{
   strncmp(msg22.msgodbd,  xpcb->xpcbdbd,  8);
   strncmp(msg22.msgoseg,  xpcb->xpcbseg,  8);
   strncmp(msg22.msgofunc, xpcb->xpcbcall, 4);
   memset(&(ekyrcpic->picxmesg.picxml2), ' ', 70);
   strncpy(ekyrcpic->picxmesg.picxml2, msg22.msgotxt2, 43);

   return;

} /* end of errcom */

#pragma page(1)
/*********************************************************************
 * IMS to DB2 - KFBA is missing.                                     *
 *********************************************************************/
void invkfb(EKYRCPIC *ekyrcpic)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR7E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "IMS-to-DB2: KFBA is missing for REPL call                    ",
      61);

   return;

} /* end of invkfb */

#pragma page(1)
/*********************************************************************
 * DB2 to IMS - Invalid call function in the HEC.                    *
 *********************************************************************/
void invfun(EKYRCPIC *ekyrcpic)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR8E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "DB2-to-IMS: Invalid call function in the HEC                 ",
      61);

   return;

} /* end of invfun */

#pragma page(1)
/*********************************************************************
 * DB2 to IMS - PCB label not found.                                 *
 *********************************************************************/
void lablnf(EKYRCPIC *ekyrcpic)
{
   ekyrcpic->picxretc = 16;
   strncpy(ekyrcpic->picxmesg.picxml1.picxmsgi, "EKYEPR9E", 8);
   ekyrcpic->picxmesg.picxml1.picxmsgb = ' ';
   strncpy(ekyrcpic->picxmesg.picxml1.picxmtxt,
      "DB2-to-IMS: PCBLABEL not found                               ",
      61);

   return;

} /* end of lablnf */
/* end of program */
```

Figure 62 (Part 18 of 18). Second Sample Propagation Exit Routine (C)

Definitions for Second Sample Propagation Exit

This section contains definitions associated with the second sample Propagation exit routine. The following types of definitions are provided:

- IMS DBDGEN and PSBGEN definitions
- DB2 CREATE TABLE definitions
- DataRefresher definitions required to define the PR DataRefresher and to extract the IMS data with DataRefresher
- SQL statements defining the PR without DataRefresher in the MVG input tables

DBDGEN Definitions

Figure 63 shows a DBDGEN definition for the Propagation exit routine in Figure 62 on page 236.

```
*
***         DESCRIPTION OF THE FIRST DBD
*
         DBD NAME=IMSDB1,VERSION=V123456789,                           C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),               C
               EXIT=(EKYRUP00)
         DATASET DD1=IMSDB1,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=SEG1,PARENT=0,BYTES=25
         FIELD  NAME=(SEG1KEY,SEQ,U),BYTES=6,START=1
*
***         DESCRIPTION OF THE SECOND DBD
*
         DBD NAME=IMSDB2,VERSION=V123456789,                           C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),               C
               EXIT=(EKYRUP00)
         DATASET DD1=IMSDB2,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=SEG2,PARENT=0,BYTES=25
         FIELD  NAME=(SEG2KEY,SEQ,U),BYTES=6,START=1
*
         DBDGEN
         FINISH
         END
```

Figure 63. DBDGEN Definition

Note: The EXIT= keyword of the DBD macros specify that EKYRUP00 (the RUP) be called when a segment of these DBDs is changed. This is required for synchronous data propagation with DPROP.

CREATE TABLE Statement

Figure 64 on page 255 shows a CREATE TABLE statement for the Propagation exit routine in Figure 62 on page 236.

```
   CREATE TABLE T096606.TABX
     (TABXCOL1    CHAR(6)     NOT NULL,
      TABXCOL2    CHAR(7)             ,
      TABXCOL3    CHAR(4)             ,
      TABXCOL4    CHAR(8)             ,
      PRIMARY KEY (TABXCOL1))
      DATA CAPTURE CHANGES
    IN DU096606.PROPTS;

CREATE UNIQUE INDEX XN01 ON TABX (TABXCOL1)
       USING VCAT KOE ;
```

Figure 64. CREATE TABLE Statement

Note: The DATA CAPTURE CHANGES option of the CREATE TABLE command specifies that the DB2 Changed Data Capture exit (the HUP) be called when a row of this table is changed under IMS attach.

Using DataRefresher to Define the PR

This section shows how can use DataRefresher to define the PR for the Propagation exit routine in Figure 62 on page 236.

CREATE DXTPSB

Figure 65 shows a CREATE DXTPSB statement for the Propagation exit routine in Figure 62 on page 236.

```
   CREATE DXTPSB   NAME=KOEPSB

     DXTPCB   NAME=DECADIX1, DBNAME=IMSDB1, DBACCESS=HDAM

     SEGMENT  NAME=SEG1, PARENT=0, BYTES=25

         FIELD    NAME=SEG1KEY,  START=1 , BYTES=6, SEQFLD=R
         FIELD    NAME=SEG1DAT1, START=7 , BYTES=7, TYPE=C
         FIELD    NAME=SEG1DAT2, START=14, BYTES=4, TYPE=C
         FIELD    NAME=SEG1DAT3, START=18, BYTES=8, TYPE=C

     DXTPCB   NAME=DECADIX2, DBNAME=IMSDB2, DBACCESS=HDAM

     SEGMENT  NAME=SEG2, PARENT=0, BYTES=25

         FIELD    NAME=SEG2KEY,  START=1 , BYTES=6, SEQFLD=R
         FIELD    NAME=SEG2DAT1, START=7 , BYTES=7, TYPE=C
         FIELD    NAME=SEG2DAT2, START=14, BYTES=4, TYPE=C
         FIELD    NAME=SEG2DAT3, START=18, BYTES=8, TYPE=C   ;
```

Figure 65. CREATE DXTPSB

The DXTPXB contains two DXTPCBs, each referring to a particular database.

CREATE DXTVIEW

Figure 66 on page 256 shows a CREATE DXTVIEW statement for the Propagation exit routine in Figure 62 on page 236.

```
CREATE   DXTVIEW NAME     = VIEW01,
                 DXTPSB   = KOEPSB,
                 DXTPCB   = DECADIX1,
                 SEGMENT  = SEG1,
                 MINSEGM  = SEG1,
                 FIELDS   = *     ;

CREATE   DXTVIEW NAME     = VIEW02,
                 DXTPSB   = KOEPSB,
                 DXTPCB   = DECADIX2,
                 SEGMENT  = SEG2,
                 MINSEGM  = SEG2,
                 FIELDS   = *     ;
```

Figure 66. CREATE DXTVIEW Statement

DataRefresher UIM SUBMIT Command and EXTRACT Statement

Figure 67 shows a DataRefresher UIM SUBMIT command and EXTRACT statement for the Propagation exit routine in Figure 62 on page 236.

```
SUBMIT   EXTID=PROP01,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=U,
                         MAPDIR=TW,
                         ACTION=REPL,
                         ERROPT=BACKOUT,
                         EXITNAME=EKYEPR2K,
                         PROPSEGM=(IMSDB1/SEG1,IMSDB2/SEG2)'

EXTRACT
    INTO T096606.TABX    (TABXCOL1   NOT NULL,
                          TABXCOL2           ,
                          TABXCOL3           ,
                          TABXCOL4           )
                 SELECT  SEG1KEY,
                          SEG1DAT1,
                          SEG1DAT2,
                          SEG1DAT3
               FROM VIEW01, VIEW02      ;

SUBMIT   EXTID=EXTR02,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,
               JCS=DDJCS01,
               USERDECK='RESUME(YES)',
               FORMAT=SOURCE

EXTRACT
    INTO T096606.TABX    (TABXCOL1   NOT NULL,
                          TABXCOL2           ,
                          TABXCOL3           ,
                          TABXCOL4           )
                 SELECT  SEG2KEY,
                          SEG2DAT1,
                          SEG2DAT2,
                          SEG2DAT3
               FROM VIEW02              ;
```

Figure 67. DataRefresher UIM SUBMIT Command and EXTRACT Statement

Notes:

1. It is necessary to provide two DataRefresher extract requests (ER) to extract the complete data by the DEM, but only the first extract request becomes a propagation request (PR) for DPROP.

In the figure above, the first ER is the PR used by DPROP, and the second ER is required only to extract the data from the second database.

The following description refers only to the propagation request (PROP01).

2. The MAPEXIT= keyword of the SUBMIT command specifies EKYMCE00. This causes DataRefresher UIM to call the DPROP-provided Map Capture Exit EKYMCE00 during the processing of the SUBMIT/EXTRACT. This is required to allow DPROP to create the PR.
3. PRTYPE=U (user mapping) must be specified, because the PR should be processed by a Propagation exit routine.
4. EXITNAME=EKYEPR2K specifies the name of the Propagation exit routine which will perform the propagation for this PR.
5. PROPSEGM=(IMSDB1/SEG1,IMSDB2/SEG2) identifies the segment types and their respective databases being propagated by this PR.
6. FROM VIEW01,VIEW02 identifies the views for the two databases that this PR propagates.
7. The EXTRACT statement describes to DataRefresher which fields should be mapped to which columns during the data extract. These definitions are important for the extract but are not important for DPROP because the mapping and propagation is not done by the generalized mapping logic of DPROP.
8. There is no PCBLABEL provided in the MAPUPARM operand. DPROP needs two different PCB labels (two different databases). The two PCB labels needed by the HUP to perform DB2-to-IMS propagation are the names of the DXTPCBs provided at DXTPSB coding (DECADIX1 and DECADIX2). These names are picked up by DPROP and are passed in the HEC to the Propagation exit routine.

Using DataRefresher for the Extract

This section covers INITDEM and USE DXTPSB Control Statements. Figure 68 shows a INITDEM and USE DXTPSB control statements for the Propagation exit routine in Figure 62 on page 236.

```
INITDEM NAME=BASILEUS;
USE DXTPSB=KOEPSB;
```

Figure 68. Using DataRefresher for the Extract: INITDEM and USE DXTPSB Control Statements

Defining the PR in the MVG Input Tables

Figure 69 on page 258 describes DSNTEP2 SQL statements required to define the PR in the MVG input tables.

The following rows are inserted into the MVG input tables:

- One row is inserted into the DPRIPR table (the PR table).

This row identifies the PRID, indicates that the PRTYPE is U (user mapping), and provides in the EXITNAME column the name of the Propagation exit routine, EKYEPR2K, which performs the propagation for this PR.

- One row for each segment type being propagated by the PR and the Propagation exit routine is inserted into the DPRISEG table (the SEG table).

  As explained in the commentary of the source code of EKYEPR2K, the sample exit routine propagates changes to segment SEG1 of database IMSDB1 or to segment SEG2 of database IMSDB2 depending on the key content.

- One row is inserted into the DPRITAB table (the TAB table).

  This row indicates that the target table is T096606.TABX.

For PRTYPE=U, DPROP does not require that you insert any rows in the DPRIFLD table; this is why the example below does not insert any rows in the DPRIFLD table.

```
DELETE FROM T096606.DPRIPR WHERE PRID = 'PROP01'           ;


INSERT INTO T096606.DPRIPR
            ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR,
              ERROPT,   ACTION,   EXITNAME)
     VALUES ('PROP01','T096606','U',     ' ',     'TW',
             'BACKOUT','REPL',   'EKYEPR2K')             ;

INSERT INTO T096606.DPRISEG
            ( PRID,    DBNAME,    SEGNAME, ROLE, PCBLABEL)
     VALUES ('PROP01','IMSDB1', 'SEG1', ' ', 'DECADIX1')  ;

INSERT INTO T096606.DPRISEG
            ( PRID,    DBNAME,    SEGNAME, ROLE, PCBLABEL)
     VALUES ('PROP01','IMSDB2', 'SEG2', ' ', 'DECADIX2')  ;

INSERT INTO T096606.DPRITAB
            ( PRID,    TABQUAL,   TABNAME)
     VALUES ('PROP01','T096606', 'TABX')                    ;

COMMIT;
```

Figure 69. DSNTEP2 SQL Statements

Chapter 5. DB2 Data Capture Subexit Routine

You will need to write a DB2 Data Capture subexit routine if your installation needs the HUP to coexist with another DB2 Data Capture exit routine. Instead of having two DB2 Data Capture exit routines (which is not supported by DB2), you will:

- Use the HUP as a DB2 Data Capture exit routine, and
- Define to DPROP the other generalized exit routine as a DB2 Data Capture subexit routine (definition is done during DPROP installation).

The purpose of the subexit routine is usually not DB2-to-IMS propagation. Instead, its purpose is usually to:

- Propagate changed DB2 rows to other tables, or
- Perform other generalized functions, such as auditing changed DB2 rows.

DPROP calls your subexit routine when the HUP is invoked by the DB2 Data Capture function. DPROP calls the subexit routine even if you have not defined a PR and even if propagation has been emergency stopped.

However, your subexit will **not** be invoked when the HUP issues a rollback of the unit of work or an abend. This is not a problem since, in this case, the SQL update can be considered nonexistent.

When your subexit routine is invoked, the HUP provides it with both the data and the description of the changed row.

Although DPROP calls the DB2 Data Capture subexit routine, it is not part of its propagation procedure. Its call occurs regardless of whether:

- Propagation requests exist.
- Propagation is suspended.
- Propagation is deactivated.
- Propagation is emergency stopped.

Therefore, your DB2 Data Capture subexit routine **cannot** benefit from DPROP support functions.

Your exit routine can be written in Assembler, COBOL, PL/I, or C. DPROP support for exit routines written in HLL requires LE/370 Version 1 Release 2.

The DB2 Data Capture subexit routine is called when the HUP receives captured DB2 data. This applies to IMS batch and online dependent regions accessing DB2. Your DB2 Data Capture subexit routine runs within the same unit of work (UOW) as the updating application program and propagation request. Avoid using functions affecting PR processing, including:

- Execution of SQL COMMIT and ROLLBACK
- IMS CHKP, SETS, ROLS, ROLL, and ROLB calls
- IMS INIT STATUS GROUPA and GROUPB calls
- Execution of IFI calls requesting captured data
- ABENDs of your exit

How To Write a DB2 Data Capture Subexit Routine

Because the DB2 Data Capture subexit routine is not considered to be part of propagation, DPROP does not have special requirements for it.

DB2 Data Capture Subexit Routine Interface

When DPROP receives the changed data, it performs normal propagation. After processing a PR for the table for which data was captured, it calls your DB2 Data Capture subexit routine.

The HUP calls your subexit routine with the following parameters:

- A 64-byte anchor area.
- The HEC. The HEC is a DPROP control block that contains pointers to areas passed by the DB2 Data Capture exit.

Upon entry to your subexit routine, Register 1 contains the address of the list. This list is two fullwords long and contains the addresses of the parameters in the order listed above.

64-Byte Anchor Area

DPROP gives you 64 bytes as a general-purpose storage area. You can use it for whatever you want. Initially, the area is set to all binary zeros, and DPROP never changes it again.

The anchor area exists in virtual storage, and remains yours for the duration of the exit.

- For IMS Batch and BMP regions, the anchor area lasts for the duration of the application program.
- For MPP regions, the anchor area lasts for the duration of the IMS Program Controller Subtask. This spans multiple MPP executions.

HEC Interface

The HEC is the second parameter passed to your DB2 Data Capture subexit routine when the HUP calls it. It is used to provide the pointers to the areas received from the DB2 Data Capture (DB2CDC) and passed to your exit. These areas describe and contain the captured changed data, and are listed below:

QWHC Is the DB2 Instrumentation Facility standard header mapped by DSNDQWHC

QWHS Is the DB2 Instrumentation Facility correlation data mapped by DSNDQWHS

CDCDD Contains the Data Capture table description and is mapped by the QW0185 DSECT within DSNDQW02

CDCDA Contains the Data Capture data row and is also mapped by the QW0185 DSECT within DSNDQW02

For inserts and deletes there is one data row with the data of the inserted or deleted row. For updates there is one data row containing the after-image and one data row with the before-image of the updated row.

Your exit routine must not modify the HEC or the data pointed to by this control block.

Figure 70 provides an overview of the interface defined through the HEC.

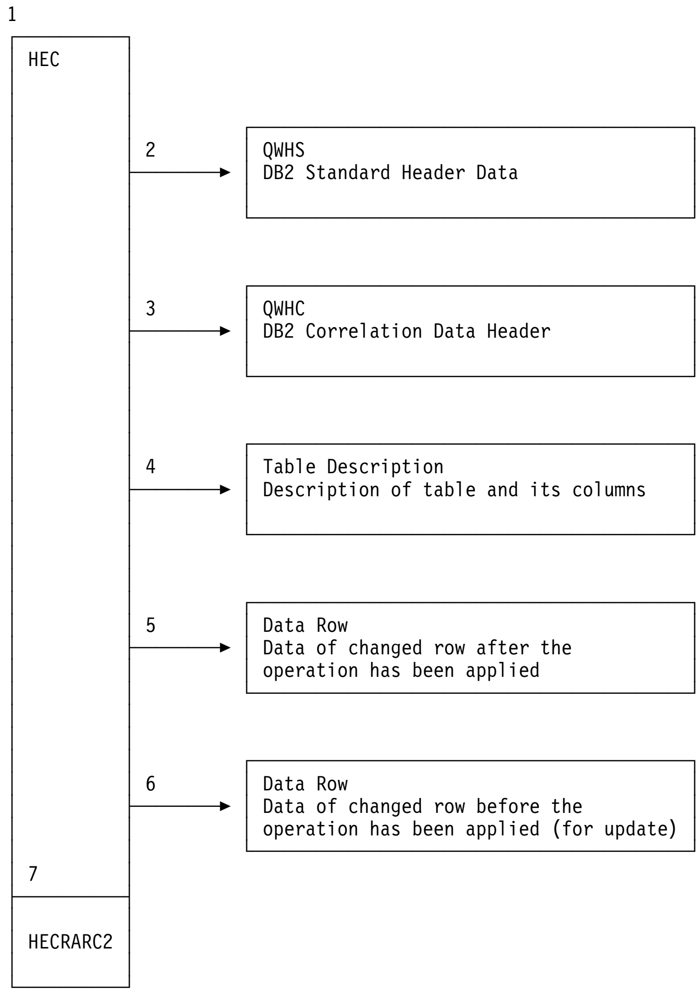

Figure 70. HEC, QWHS, QWHC, Table Description and Data Row Control Block Structures

As shown in the numbered sections of the figure, the interface consists of:

1. One HEC control block that provides various pointers.
2. A pointer to the DB2 Instrumentation Facility standard header data that contains specific DB2 information based on the active trace.
3. A pointer to the DB2 Instrumentation Facility correlation data header containing information about correlation and authorization.
4. A pointer to the Data Capture table description of the changed table and its columns.
5. A pointer to the Data Capture Data (data row) record containing the **after** image of the captured row. For SQL INSERT and DELETE, this is the only data row passed to your exit routine.

6. A pointer to the Data Capture Data (data row) record containing the **before** image of the captured row. This data row is only present for update operations.

7. A field containing the reason code returned by DB2 for the generated IFI call to retrieve the captured data. See *DB2 Messages and Codes* for a description of IFI reason codes.

HEC Control Block DSECT

You can generate the following DSECT in your assembler exit routine by coding the EKYHCHEC macro statement. For HLL exit routines, you can include or copy one of the following members to map the HUP Exit Communication Block:

EKYHCHCC Exit routines written in COBOL
EKYHCHCP Exit routines written in PL/I
EKYHCHCK Exit routines written in C

```
                                      1          EKYHCHEC
                                      2+****************** START OF CONTROL BLOCK SPECIFICATION **************
                                      3+*                                                                     *
                                      4+*       CONTROL BLOCK NAME:                                           *
                                      5+*         EKYHCHEC (HEC)                                              *
                                      6+*                                                                     *
                                      7+*       DESCRIPTIVE NAME:                                             *
                                      8+*         DPROP HUP EXIT COMMUNICATION BLOCK                          *
                                      9+*               =   =    =                                            *
                                     10+*                                                                     *
                                     11+***********************************************************************
                                     12+*                                                                     *
                                     13+*         THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".        *
                                     14+*                                                                     *
                                     15+*         5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                *
                                     16+*         ALL RIGHTS RESERVED.                                        *
                                     17+*                                                                     *
                                     18+*         U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                   *
                                     19+*         USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY               *
                                     20+*         GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                    *
                                     21+*                                                                     *
                                     22+*         LICENSED MATERIALS - PROPERTY OF IBM.                       *
                                     23+*                                                                     *
                                     24+***********************************************************************
                                     25+*                                                                     *
                                     26+*       STATUS: V1 R2 M0                                              *
                                     27+*                                                                     *
                                     28+*       FUNCTION:                                                     *
                                     29+*         THIS IS THE CONTROL BLOCK USED TO PASS INFORMATION          *
                                     30+*         GOT BY DPROP FROM THE DB2 CHANGED DATA CAPTURE EXIT         *
                                     31+*         (USING IFI CALLS) TO THE PROPAGATION EXIT ROUTINE           *
                                     32+*         AND / OR THE DB2 CHANGED DATA CAPTURE SUBEXIT ROUTINE.      *
                                     33+*                                                                     *
                                     34+*         THE HEC IS BUILD FOR EACH EXIT CALL NEW AND DOES            *
                                     35+*         CONTAIN DATA TO BE RETAINED BEETWEEN EXIT CALLS.            *
                                     36+*                                                                     *
                                     37+*       MODULE TYPE= MACRO                                            *
                                     38+*          PROCESSOR= ASSEMBLER H                                     *
                                     39+*                                                                     *
                                     40+*       INNER CONTROL BLOCKS: NONE                                    *
                                     41+*                                                                     *
                                     42+*       MACROS USED FROM MACRO LIBRARY: NONE                          *
                                     43+*                                                                     *
                                     44+*       CHANGE ACTIVITY:                                              *
                                     45+*                                                                     *
                                     46+******************** END OF CONTROL BLOCK SPECIFICATION **************

000000                               48+HEC      DSECT ,                       START OF CONTROL BLOCK

                                     50+*------- EYE CATCHTERS
000000                               51+HECEYE   DS    0CL8                    EYE-CATCHER AREA
000000 C5D2E840                      52+HECEYE1  DC    CL4'EKY '               EYE-CATCHER DPROP
000004 C8C5C340                      53+HECEYE2  DC    CL4'HEC '               EYE-CATCHER CONTROL BLOCK
000008 0000000000000000              54+HECRESV1 DC    2F'0'                   RESERVED

                                     56+*------- POINTERS TO IFI HEADER AREAS
000010 00000000                      57+HECQWHS  DC    A(*-*)                  ADDRESS OF THE DB2 IFI
                                     58+*                                      STANDARD HEADER AREA
000014 00000000                      59+HECQWHC  DC    A(*-*)                  ADDRESS OF THE DB2 IFI
                                     60+*                                      CORRELATION DATA AREA

                                     62+*------- POINTERS TO CDC DATA AREAS
000018 00000000                      63+HECCDCDD DC    A(*-*)                  ADDRESS OF CDC DATA DESCRIPT.
                                     64+*                                      ALWAYS PASSED TO EXIT
00001C 00000000                      65+HECCDCDA DC    A(*-*)                  ADDRESS OF CDC DATA ROW
                                     66+*                                      ALWAYS PASSED TO EXIT.
```

Figure 71 (Part 1 of 2). HUP Exit Communication Block

```
                                      67+*                                   ONLY DATA FOR INSERT/DELETE
                                      68+*                                   OR CONTAINS THE AFTER
                                      69+*                                   IMAGE FOR UPDATE OPERATIONS
000020 00000000                       70+HECCDCDB DC    A(*-*)               ADDRESS OF CDC DATA ROW.
                                      71+*                                   ZERO FOR INSERT AND DELETE
                                      72+*                                   OR BEFORE IMAGE OF ROW FOR
                                      73+*                                   UPDATE OPERATIONS

                                      75+*------- RETURN CODE FROM IFI CALL
000024 00000000                       76+HECRARC2 DC    F'0'                 IFCRC2 REASON CODE

                                      78+*------- DBDNAME/SEGNAME/PCBLABEL AREA (MAPPED BY HECDSLDS BELOW)
000028 00000000                       79+HECDBSLA DC    A(*-*)               ADDR. OF DBD/SEG/PCBLABEL AREA
00002C 00000000                       80+HECDBSLN DC    F'0'                 NUMBER OF ENTRIES IN THIS AREA

                                      82+*------- RESERVED SPACE AND CB SIZE
000030 0000000000000000               83+HECRESV2 DC    4F'0'                RESERVED
000040                                84+HECEND   DS    0D                   END OF CONTROL BLOCK
                             00040    85+HECLEN   EQU   *-HEC                LENGTH OF CONTROL BLOCK

                                      87+*-------------------------------------------------------------------*
                                      88+*        FOR PROPAGATION EXIT ROUTINES ONLY, THE HECDBSLA FIELD      *
                                      89+*        POINTS TO AN AREA (FOR DB2 SUBEXIT ROUTINES THIS FIELD IS   *
                                      90+*        ZERO). THIS AREA CONTAINS 24 BYTE ENTRIES (IN TOP TO BOTTOM *
                                      91+*        HIERARCHY) WHICH WAS DEFINED TO DPROP FOR THE PR IN PROCESS. *
                                      92+*        THE NUMBER OF ENTRIES IN THIS LIST IS CONTAINED IN THE      *
                                      93+*        HECDBSLN FIELD.                                             *
                                      94+*-------------------------------------------------------------------*
000040                                95+HECDSLDS DS    0D                   ENTRY FOR DBD/SEG/PCBLABEL
000040                                96+HECDBDNM DS    CL8                  - DBD NAME
000048                                97+HECSEGNM DS    CL8                  - SEGMENT NAME
000050                                98+HECPCBNM DS    CL8                  - PCB LABEL NAME
                             00018    99+HECDSLDL EQU   *-HECDSLDS           LENGTH OF ONE ENTRY
                                     100          END
```

Figure 71 (Part 2 of 2). HUP Exit Communication Block

The QWHS and QWHC Control Blocks

The IFI standard header data and IFI correlation data are passed as received from the DB2 Instrumentation Facility.

DSNDQWHS Is the DB2 provided macro which maps the standard header data
DSNDQWHC Is the DB2 provided macro which maps the correlation data

Refer to *DB2 Administration Guide* for information about these control blocks.

The Table Description and Data Row Control Blocks

The Data Capture table description contains a description of the captured data. It is always present when the HUP calls your DB2 Data Capture subexit routine.

The Data Capture Data (data row) contains a row's data. When the HUP calls your DB2 Data Capture subexit routine, it passes one or two data row areas, depending on the type of SQL operation that caused the data to be captured:

- For INSERT and DELETE, there is only one data row that contains either the inserted or deleted row.
- For UPDATE, there are two data rows, one containing the image of the row before the update, and one containing the image after the update operation.

  Both data rows have the same format and are described by the same Data Capture table description that is passed to your exit routine.

The table description and data row are composed of a header common to both, and a data part, which is different for each control block type:

- The header part describes the table, using its qualified table name and the time stamp of the table description. For the data row, it also contains the RBAs of log records, the operation code, and the operation code qualifier.
- The data part of the table description contains a description of the columns of the table. The description is similar to the SQLDA.
- The data part of the data row contains the row data, as described in the table description data part.

You can generate the following DSECT (provided by DB2) in your assembler exit routine by coding the DSNDQW02 macro statement. This macro contains the QW0185 DSECT that represents the mapping of the table description and data row control blocks that the DB2 Data Capture uses.

For HLL exit routines, you can include or copy one of the following members to map the table description and data row control blocks:

EKYHCQ2C Exit routines written in COBOL
EKYHCQ2P Exit routines written in PL/I
EKYHCQ2K Exit routines written in C

```
                                      DSNDQW02

                                     +*****************************************************************
                                     +*  QW00185 IS WRITTEN FOR READS REQUESTS FOR IFCID 185.          *
                                     +*  FOR IFCID 185, THE PRODUCT SECTION WILL PRECEDE THE DATA      *
                                     +*  SECTION.  A SINGLE READS REQUEST FOR IFCID 185 MAY RESULT IN  *
                                     +*  A SERIES OF 185 RECORDS.  ONLY THE FIRST 185 RECORD IN SUCH A *
                                     +*  A SERIES WILL CONTAIN A PRODUCT SECTION.  IFCID 185 RECORDS   *
                                     +*  MAY BE BROKEN AT ANY POINT IN THE DATA.  IT IS UP TO THE      *
                                     +*  READER OF THE RECORD TO INTERPRET SPANNED IFCID 185 RECORDS.  *
                                     +*                                                                *
                                     +*  QW0185 CONTAINS A HEADER SECTION WHICH IS FOLLOWED BY A DATA  *
                                     +*  SECTION.  THE DATA PORTION OF QW0185 BEGINS WITH FIELD        *
                                     +*       - QW0185ID IF QW0185TP=S                                 *
                                     +*      OR                                                        *
                                     +*       - QW0185DR IF QW0185TP=D                                 *
                                     +*****************************************************************
000000                               +QW0185    DSECT                    READS IFCID FOR DATA OF DB2CDC
000000                               +QW0185LN  DS     F                 LENGTH OF TOTAL DB2CDC DATA
000004                               +QW0185TP  DS     CL1               TYPE: S = DB2CDC TABLE
                                     +*                                            DESCRIPTION
                                     +*                                      D = DB2CDC DATA ROW
000005                               +          DS     CL3               RESERVED
000008                               +QW0185RC  DS     CL4               REASON CODE DESCRIBING ERROR
                                     +*                                  FOR THIS DATA PORTION
00000C                               +QW0185QT  DS    0CL26              QUALIFIED TABLE NAME
00000C                               +QW0185CR  DS     CL8               CREATOR OF TABLE (AUTH ID)
000014                               +QW0185TB  DS     CL18              TABLE NAME
000026                               +QW0185TS  DS     CL10              TIMESTAMP (INTERNAL FORMAT) OF
                                     +*                                  TABLE DESCRIPTION FROM CATALOG
000030                               +QW0185TL  DS     CL10              TIMESTAMP (INTERNAL FORMAT) OF
                                     +*                                  LOG BUFFER CI WHEN IT IS EXTERNAL-
                                     +*                                  IZED OR WHEN THE BUFFER IS
                                     +*                                  INITIALIZED
00003A                               +QW0185UR  DS     CL8               RBA OF THE FIRST LOG RECORD FOR
                                     +*                                  THIS UNIT OF WORK.
000042                               +QW0185LR  DS     CL8               RBA OF LOG RECORD THAT THIS
                                     +*                                 DB2CDC DATA ROW WAS DERIVED FROM
00004A                               +QW0185PC  DS     CL2               OPERATION CODE.
                                     +*                                  USED ONLY IF QW0185TP=D, IN
                                     +*                                  WHICH CASE, QW0185PC MAY HAVE
                                     +*                                  ANY OF THE FOLLOWING VALUES:
                                     +*                                    IN - INSERT
                                     +*                                    UB - UPDATE BEFORE IMAGE
                                     +*                                    UA - UPDATE AFTER IMAGE
                                     +*                                    DE - DELETE
                                     +*                                    '0000'X IF QW0185TP = 'S'.
00004C                               +QW0185RI  DS     CL2               OPERATION CODE QUALIFIER.
                                     +*                                    '0000'X IF QW0185TP = 'S'.
                                     +*                                    'RI' IF THE OPERATION IS THE
                                     +*                                      RESULT OF A REFERENTIAL
                                     +*                                      CONSTRAINT ENFORCEMENT OF
                                     +*                                      A DELETE SET NULL OR
                                     +*                                      CASCADE OPERATION AND
                                     +*                                      IF QW0185TP = 'D'.
00004E                               +          DS     CL6               RESERVED
                            00054    +QW0185HL  EQU    84                TOTAL LENGTH OF HEADER PORTION
000054                               +QW0185DA  DS    0C                 BEGIN OF DATA PORTION
                                     +*****************************************************************
                                     +*                                                                *
                                     +*  IFCID 185 DATA PORTION FOLLOWS                                *
                                     +*                                                                *
                                     +*  IF QW0185TP = S, THEN                                         *
                                     +*    THE DATA PORTION CONSISTS OF FOUR VARIABLES FOLLOWED BY AN  *
                                     +*    ARBITRARY NUMBER OF OCCURRENCES OF THE QW0185VR STRUCTURE.  *
                                     +*                                                                *
                                     +*****************************************************************
```

Figure 72 (Part 1 of 2). Table Description and Data Row Control Blocks

```
000054                00054   68+         ORG    QW0185DA
000054                        69+QW0185ID  DS     CL8             EYE CATCHER = 'CDCDD   '
00005C                        70+QW0185BC  DS     F               LENGTH OF THE CDCDD =
                              71+*                                 (QW0185NO * 44) +16
000060                        72+QW0185NO  DS     H               TOTAL NUMBER OF OCCURRENCES OF
                              73+*                                QW0185VR
000062                        74+QW0185LD  DS     H               NUMBER OF COLUMNS DESCRIBED BY
                              75+*                                OCCURRENCES OF QW0185VR
000064                        76+QW0185VR  DS    0CL44            DESCRIBES A COLUMN IN A
                              77+*                                CAPTURED TABLE
000064                        78+QW0185ST  DS     H               TELLS THE DATA TYPE OF THE
                              79+*                                COLUMN AND WHETHER IT HAS AN
                              80+*                                ASSOCIATED INDICATOR VARIABLE
000066                        81+QW0185LE  DS     H               DEFINES THE EXTERNAL LENGTH OF
                              82+*                                A VALUE FROM THE COLUMN
000068                        83+QW0185SD  DS     F               CONTAINS THE CCSID (CODED CHAR
                              84+*                                SET ID IN BYTES 3 AND 4.
00006C                        85+QW0185SI  DS     F               OFFSET OF THIS COLUMN INTO THE
                              86+*                                DATA ROW
000070                        87+QW0185SN  DS    0C               LENGTH OF NAME AND NAME OF THE
                              88+*                                COLUMN
000070                        89+QW0185NL  DS     H               LENGTH OF COLUMN NAME OR LABEL
000072                        90+QW0185CN  DS     CL30            NAME OR LABEL OF COLUMN
                              91+*
                              92+*************************************************************
                              93+*                                                           *
                              94+* IF QW0185TP = D, THEN                                     *
                              95+*    THE DATA PORTION CONSISTS OF                           *
                              96+*      - THE DATA ROW IF QW0185RC EQUAL 0.                  *
                              97+*      OR                                                   *
                              98+*      - AN ERROR MESSAGE IF QW0185RC NOT EQUAL 0.          *
                              99+*                                                           *
                             100+*  IN THIS CASE, LENGTH OF DATA PORTION IS QW0185LN - QW0185HL.  *
                             101+*                                                           *
                             102+*************************************************************
000090                00054  103+         ORG    QW0185DA
000054                       104+QW0185DR  DS    0C              DATA ROW OR ERROR MESSAGE
                             105          END
```

Figure 72 (Part 2 of 2). Table Description and Data Row Control Blocks

The Table Description and Data Row Header

The following describes the fields of the table description and data row header part in more detail:

QW0185LN Length of the total table description or data row (header and data).

QW0185TP Contains the CDC control block type:

- **S** For the DB2CDC table description
- **D** For the DB2CDC data row

QW0185RC Reason code describing errors for this table and used only for the data row. If a severe error was detected for this table, the HUP calls your DB2 Data Capture subexit routine only if there is no PR defined for the captured data. In the other case, to keep propagated data consistent, the HUP enforces the rollback of the changes. The reason codes that your DB2 Data Capture subexit routine must handle are:

X'00E60A01' The following message is returned in the data portion:

```
VIOLATION OF INSTALLATION DEFINED EDIT PROCEDURE proc_name,
REASON CODE: reason_code
```

X'00E60A08' The following message is returned in the data portion:

```
COLUMN column_name ON TABLE table_name IN VIOLATION OF
INSTALLATION DEFINED FIELD PROCEDURE RT: return_code,
RS: reason_code, MSG: message_token
```

X'00E60A09' The following message is returned in the data portion:

```
INCORRECT DATA RETURNED FORM FIELD PROCEDURE fieldproc_name
FOR TABLE table_name AND COLUMN column_name, MSG: message_token
```

X'00E60A0A' The following message is returned in the data portion:

```
AN INSTALLATION FIELD PROCEDURE HAS RETURNED A RETURN CODE
IN REGISTER 15 OTHER THAN AN EXPECTED 0 OR 4
```

X'00E60A0B' This code indicates that although the date or time install option was specified as LOCAL, a date or time column value of the row has been returned in ISO format. The DB2 Data Capture never calls date and time exits.

QW0185QT The qualified table name, which is composed of the table creator (QW0185CR) and table name (QW0185TB).

QW0185CR The creator name (authorization ID), which is 8 bytes long and padded with blanks.

QW0185TB The table name, which is 18 bytes long and padded on the right with blanks.

QW0185TS The time stamp (internal format) of the table description from the catalog.

QW0185TL The time stamp (internal format) of the log record within the log buffer CI. This field is present only in the data row (QW0185TP=D).

QW0185UR RBA of the first log record for this unit of work. This field is present only in the data row (QW0185TP=D).

QW0185LR RBA of log record of this data row. This field is present only in the data row (QW0185TP=D).

QW0185PC Operation code describing the type of row image and the SQL operation that performed the data change. This field is present only in the data row (QW0185TP=D). The possible values of QW0185PC are:

Code	Description
IN	Insert
UB	Update before-image
UA	Update after-image
DE	Delete

QW0185RI Operation code qualifier present only in the data row (QW0185TP=D). This field is either blanks, or **RI** if the operation is a result of a referential constraint enforcement of a DELETE SET NULL or CASCADE operation.

The Table Description Data

The table description data portion contains a similar form of an **SQLDA** that describes the table. It is like the standard SQLDA external format, except for the field where you usually specify the address of the data area for a particular column. In the CDC table description, this field is already set and contains the **offset to the column** within the data row data section, which is optionally prefixed by a null indicator variable.

The data portion of the table description consists of four variables followed by an arbitrary number of occurrences of a sequence of five variables, collectively called QW0185VR.

QW0185ID An eye catcher for storage dumps containing **CDCDD**

QW0185BC Length of the table description data portion, which is (QW0185NO * 44) + 16

QW0185NO Total number of occurrences of QW0185VR

QW0185LD The number of columns described by occurrences of QW0185VR

The following five variables are collectively called QW0185VR and occur QW0185NO times in the table description. Each occurrence of QW0185VR describes a column in the captured table.

QW0185ST Tells the data type of the column and whether it has an associated indicator variable. For a description of the type codes, see Figure 73 on page 270.

QW0185LE Defines the external length of a value of the column, as follows:

Data Type	Content
Character	Length attribute in bytes
Graphic	Length attribute in bytes
Decimal	byte 1 = precision byte 2 = scale
Float	4 (bytes) for single precision 8 (bytes) for double precision
Smallint	2 (bytes)
Integer	4 (bytes)
Date	10 (bytes) or LOCAL value
Time	8 (bytes) or LOCAL value
Time stamp	26 (bytes).

QW0185SD Contains the CCSID (Coded Character Set Identifier) in bytes 3 and 4. It is a two-byte (unsigned) binary number that uniquely identifies an encoding scheme and one or more pairs of character sets and code pages.

QW0185SI Contains a flag byte and the offset of this column into the data row. The flag byte indicates if the column can be nullable or not. If the column can be NULL, then the column data in the data row is prefixed by an indicator variable (2 bytes). The offset points to the null indicator variable instead of the data for the column; the data immediately follows the indicator and starts at offset + 2. The indicator variable is a two-byte field in the data row containing X'FFFF' (value -1) if the field is null, or X'0000' if the field contains data.

The format of the QW0185SI field is:

Bytes	Content
1	Flag byte. If the highest bit (bit 0) is on, then the column is prefixed with a null indicator variable, and the real data starts at offset + 2. The rest of the bits are reserved.
2-4	Offset into the data, or indicator variable for this column. This offset must be added to the data row data portion address (QW0185DR) to compute the virtual storage address of the column data or indicator variable.

QW0185SN Length of name (QW0185NL) and name of the column (QW0185CN).

QW0185NL Contains the length of the column name.

QW0185CN Contains the name of the column.

Figure 73 lists values of the QW0185ST field of the table description and their meanings. There are two values for each data type. The first value means that the column does not have a null indicator and does not allow nulls; the second means that the column has a null indicator and allows nulls. For more information about data types refer to the *DB2 SQL Reference*.

Figure 73. Values of QW0185ST and Their Meanings

Values	Data Type
384/385	Date
388/389	Time
392/393	Time stamp
448/449	Variable-length character string
452/453	Fixed-length character string
456/457	Long character string
460/461	Variable-length, optionally null terminated character string (C)
464/465	Variable-length graphic string
468/469	Fixed-length graphic string
472/473	Long graphic string
480/481	Floating point
484/485	Decimal
496/497	Large Integer
500/501	Small Integer

The Data Row Data

The data row data portion starts at label QW0185DR. It contains actual data mapped according to the table description, with DB2 calculated **offsets** into the data for each column.

SQL inserts (IN) and SQL deletes (DE) are passed as one row pointed to by HECCDCDA, a single image that contains **all** the columns in the table.

SQL updates are passed as two rows, an after-image (UA) pointed to by HECCDCDA, and a before-image (UB) pointed to by HECCDCDB. Both images contain **all** the columns of the table.

As applicable, the rules of the external form of a table description dictate how the following data items are handled:

- A string of fields, ordered as they were specified in the external form of a table description of the table, and in standard SQL external format.
- EDITPROCs and FIELDPROCs are called as in standard SQL. The returned data is as decoded by an EDITPROC or any FIELDPROCs that apply, the same as in standard SQL.
- DBCS data is supported as in standard SQL.
- VARCHARs are padded to maximum length, but they contain the actual length in the first two bytes of the data.
- Nulls are represented by an indicator variable (two bytes), which precedes the field, but this field is not included in the length.

Exit Routine Processing

Using the information in the control blocks described above (HEC, table description, and data row), you can do your processing in any way you choose. This section describes some of the things you must consider when developing your DB2 Data Capture subexit routine.

Calling Your Exit Routine

DPROP loads your DB2 Data Capture subexit routine before its first call, and keeps it in virtual storage until the OS/VS task terminates. In MPP regions, this spans multiple MPP executions. Before calling your exit routine, the HUP determines if there is a PR for the captured data, and performs propagation, using generalized or user mapping cases, if applicable. If standard propagation must be aborted, then the HUP does **not** call your DB2 Data Capture subexit routine. This is because the whole unit of work is rolled back, and changes that the application program performs are made nonexistent.

DPROP uses standard OS/VS conventions when calling your exit routine.

Register 1 Points to the parameter list described above.
Register 13 Contains the address of a register save area.
Register 14 Contains the return address.
Register 15 Contains the entry point address of the exit routine

Upon entering the exit routine, the register contents must be saved into the caller's save area. If your exit routine calls other routines that use standard MVS linkage conventions, it must also provide a save area of its own. The exit routine must return to its caller using normal OS/VS conventions after restoring the registers.

DPROP does not analyze the return code that your DB2 Data Capture subexit routine returns in register 15. Also, like the other DPROP exit routines, your DB2 Data Capture subexit routine gains control in AMODE 31, and must return control in AMODE 31.

The DB2 Data Capture subexit routine can be called multiple times during the processing of an SQL statement, if the statement updates or deletes more than one row. The number of calls, and the order in which they are made, depends on the DB2 process sequence of the rows, and is unpredictable for DPROP and the DB2 Data Capture subexit routine.

Exit Routine Logic

Your exit routine can do any processing with the supplied captured data. For performance reasons, it is recommended that your exit routine generate static SQL calls. Avoid using functions that have a detrimental effect on the performance of the application program (such as performing an OPEN and CLOSE on an MVS file each time the exit routine is called). It is also recommended that the DBRMs of your DB2 Data Capture subexit routine be package bound. The DB2 plans created for the propagating application programs must then list the packages.

Because the exit routine executes in the same environment as the propagating application program, it can generate the same type of IMS calls and SQL statements as the application program can.

The DBRM of your DB2 Data Capture subexit routine must be included in the DB2 plans of those application programs that synchronously propagate the changed data. If your exit generates IMS calls, use the AIB interface described in *IMS/ESA Application Programming: DL/I Calls*, which allows your exit routine to generate calls without the address of the IMS PCBs.

Any changes you make to propagated data from within your DB2 Data Capture subexit routine are not captured and cannot be propagated.

A DB2 Data Capture subexit routine must not perform functions that are not supported by the environment in which it is running. For example, an exit routine running in an MPP region must not write to OS files, and the exit routine must not generate STIMER macros in an IMS environment.

It is recommended that you code and link-edit your program as reusable.

Return Codes

This section discusses how to return from your exit routine to DPROP. Remember that you must return control to the caller in AMODE 31, using the normal MVS conventions described in the previous section.

DPROP does not accept return codes from your DB2 Data Capture subexit routine, because this exit is not intended for propagation. Therefore, the DB2 Data Capture subexit routine cannot use the DPROP error handling techniques.

Saving Information Across Calls

You can save information across calls to the exit routine. Save the information in the 64-byte anchor area passed at entry to your DB2 Data Capture subexit routine. If this area is not large enough, generate a GETMAIN and save the address of the storage in the 64-byte anchor area.

Updating Your DB2 Data Capture Subexit Routine

DPROP does not provide any online change logic to replace an existing load module copy of your exit routine with a new version of the load module. If you need to change your exit routine, stop the affected IMS regions before performing the change. A change of the exit routine without stopping the IMS regions causes unpredictable results. For example, some MPP regions use the new version of the exit routine, while other regions use the old version. After the change, you can restart the IMS regions.

Telling DPROP About Your Subexit Routine

This section discusses how you can inform DPROP that you want to use a DB2 Data Capture subexit routine. To do this, during DPROP generation, specify which DB2 Data Capture subexit routine must be called when changes are captured for DB2 tables. The exit name you define applies to a whole DPROP system. The DB2 Data Capture subexit routine is called for all captured data, whether or not propagation for it exists.

Sample DB2 Data Capture Subexit Routine

The sample DB2 Data Capture subexit routine in Figure 74 on page 274 is an example of DB2-to-DB2 propagation. In this case, the subexit routine intercepts updates to the table TABLE02 and propagates the same changes to a mirror table TABLE0M.

The source code in Figure 74 on page 274 is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYEDB2A. Following the source code are definitions related to the sample DB2 Data Capture subexit routine.

```
         MACRO
         SQLSECT &TYPE
         GBLC  &SQLSECT
         AIF ('&TYPE' EQ 'RESTORE').REST
&SQLSECT SETC  '&SYSECT'
         MEXIT
.REST    ANOP
&SQLSECT CSECT
         MEND
         PRINT NOGEN
****************** START OF SPECIFICATIONS ****************************
*                                                                     *
*    MODULE NAME = EKYEDB2A                                           *
*                                                                     *
*    DESCRIPTIVE NAME = SAMPLE 'DB2 CDC SUBEXIT ROUTINE'              *
*                                                                     *
*    STATUS: V1 R2 M0                                                 *
*                                                                     *
*    FUNCTION = EKYEDB2A IS A SAMPLE DPROP 'DB2 CDC SUBEXIT ROUTINE'  *
*               WHICH ILLUSTRATES HOW TO USE THE INFORMATION PASSED   *
*               BY HUP TO SUCH A USER EXIT ROUTINE.                   *
*                                                                     *
*               BECAUSE PROPAGATION TO IMS DATABASES IS PERFORMED BY  *
*               DPROP, THE SCOPE OF A DB2 SUBEXIT ROUTINE SHOULD NOT  *
*               BE SUCH A PROPAGATION.                                *
*                                                                     *
*                 - FOR PROPAGATION OF CHANGES OF THE DB2 DATA YOU    *
*                   SHOULD USE DPROP'S:                               *
*                                                                     *
*                   -- GENERALIZED MAPPING CASES FOR PROPAGATION      *
*                      TO IMS DATABASES WHERE THE PRESCRIBED RULES    *
*                      CAN BE APPLIED                                 *
*                                                                     *
*                   -- USER MAPPING CASE FOR PROPAGATION TO IMS       *
*                      DATABASES WHERE THE RULES OF THE GENERALIZED   *
*                      MAPPING CASES ARE NOT FLEXIBLE ENOUGH          *
*                                                                     *
*                 - AND USE A DB2 SUBEXIT FOR ANY OTHER PURPOSE,      *
*                   EXCEPT PROPAGATION TO IMS DATABASES, SUCH AS:     *
*                                                                     *
*                   -- MONITORING CHANGES OF THE DB2 DATA             *
*                                                                     *
*                   -- SECURITY CHECKING                              *
*                                                                     *
*                   -- PROPAGATION TO OTHER ENVIROMENTS AS IMS DB     *
*                                                                     *
*                                                                     *
*               BECAUSE THE SCOPE OF SUCH A DB2 CDC SUBEXIT ROUTINE   *
*               SHOULD NOT BE IMS PROPAGATION, DPROP WILL INVOKE IT   *
*               REGARDLESS:                                           *
*                                                                     *
*                 - IF THERE EXIST A DPROP PROPAGATION REQUEST OR NOT *
*                 - IF DPROP PROPAGATION HAS BEEN SUSPENDED           *
*                 - IF DPROP PROPAGATION HAS BEEN DEACTIVATED         *
*                 - IF DPROP PROPAGATION HAS BEEN EMERGENCY STOPPED   *
*                                                                     *
*               HOWEVER, IT IS NOT INVOKED:                           *
*                                                                     *
*                 - IF THERE IS A PROPAGATION REQUEST FOR THE CHANGED *
*                   DATA WHICH CANNOT BE SUCCESSFULLY APPLIED AND     *
*                                                          ---        *
*                 - IF THE DROP ERROR LOGIC REQUESTS A ROLLBACK OF    *
*                   THE CHANGES MADE TO THE DB2 DATA                  *
*                                                                     *
```

Figure 74 (Part 1 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
*             THE DB2 CDC SUBEXIT ROUTINE IS INVOKED ONCE FOR EACH  *
*             RETRIEVED UPDATE EVENT IN THE IFI DATA STREAM. IF     *
*             THE ORIGINATING APPLICATION SQL STATEMENT AFFECTED    *
*             MULTIPLE ROWS, THEN THE DB2 CDC SUBEXIT ROUTINE WILL  *
*             BE INVOKED BY DPROP MULTIPLE TIMES, UNTIL ALL UPDATE  *
*             EVENTS HAVE BEEN PROCESSED BY IT. FOR EACH SINGLE     *
*             INVOCATION, THE CAPTURED DATA IS PASSED AS FOLLOWS:   *
*                                                                   *
*               - A CHANGED DATA CAPTURE DATA DEFINITION (CDCDD)    *
*                 IS ALWAYS PASSED TO YOUR EXIT. THIS AREA CONTAINS *
*                 A DEFINITION OF THE ROW DATA IN A SIMILAR FORM    *
*                 AS IN THE SQLDA.                                  *
*                                                                   *
*               - A CHANGED DATA CAPTURE DATA ROW (CDCDA) WHICH     *
*                 CONTAINS THE COLUMN VALUES OF THE AFFECTED ROW.   *
*                 THIS AREA IS ALWAYS PASSED TO YOUR EXIT AND       *
*                 REPRESENTS EITHER THE ONLY DATA ROW FOR INSERT    *
*                 AND UPDATE OPERATIONS, OR CONTAINS THE AFTER      *
*                 IMAGE OF THE ROW IN CASE OF UPDATE OPERATIONS.    *
*                                                                   *
*               - FOR UPDATE OPERATIONS, YOUR DB2 CDC SUBEXIT       *
*                 ROUTINE, WILL RECEIVE AN ADDITIONAL CHANGED       *
*                 DATA CAPTURE DATA ROW (CDCDA). THIS AREA          *
*                 CONTAINS THE BEFORE IMAGE OF THE AFFECTED ROW.    *
*                                                                   *
*             DISCLAIMERS:                                          *
*                                                                   *
*               - THIS SAMPLE EXIT IS BY PURPOSE VERY SIMPLE,       *
*                 IN ORDER TO AVOID TO OBSCURE THE MOST ESSENTIAL   *
*                 ASPECTS OF THE LOGIC OF A DB2 CHANGED DATA        *
*                 CAPTURE SUBEXIT ROUTINE.                          *
*                                                                   *
*               - THE SCOPE OF THIS SAMPLE EXIT IS THE DB2 TO       *
*                 DB2 PROPAGATION. ANY DATA UPDATE (MADE UNDER      *
*                 IMS ATTACH) TO THE TABLE 'TABLE02' IS PROPAGATED  *
*                 BY THIS DB2 SUBEXIT ROUTINE TO ITS MIRROR TABLE   *
*                 'TABLE0M'. BOTH TABLES ARE IDENTICAL AND HAVE     *
*                 THE FOLLOWING COLUMNS:                            *
*                                                                   *
*                 -- KEYFLD1      CHAR(2)       NOT NULL            *
*                 -- KEYFLD2      CHAR(6)       NOT NULL            *
*                 -- FAMILY       VARCHAR(30)                       *
*                 -- FIRST        VARCHAR(20)                       *
*                 -- CITY         VARCHAR(35)                       *
*                                                                   *
*                 EACH TABLE CONTAINS AN UNIQUE INDEX WITH THE      *
*                 COLUMNS KEYFLD1 AND KEYFLD2.                      *
*                                                                   *
*                                                                   *
*    NOTES:                                                         *
*       DEPENDENCIES = NONE                                         *
*                                                                   *
*       RESTRICTIONS = THE DB2 CHANGED DATA CAPTURE SUBEXIT RUNS    *
*                      WITHIN THE SAME UNIT-OF-WORK (UOW) AS THE    *
*                      UPDATING APPLICATION PROGRAM AND PROBABLE    *
*                      DPROP PROPAGATION REQUEST. THEREFORE YOU     *
*                      MUST AVOID THE USAGE OF FUNCTIONS AFFECTING  *
*                      THE PROCESS OF THESE, SUCH AS:               *
*                                                                   *
*                        - THE EXECUTION OF SQL COMMIT AND ROLLBACK *
*                        - IMS CHKP, SETS, ROLS, ROLL AND ROLB CALLS *
*                        - IMS INIT STATUS GROUPA AND GROUPB CALLS  *
*                        - THE EXECUTION OF IFI CALLS REQUESTING    *
*                          CAPTURED DATA                            *
*                        - ABENDS OF YOUR EXIT                      *
*                                                                   *
```

Figure 74 (Part 2 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
*       REGISTER CONVENTIONS=                                      *
*                       R0 = WORK / LINKAGE                        *
*                       R1 = WORK / LINKAGE                        *
*                       R2 = WORK                                  *
*                       R3 = WORK / SQLWA (SQLDSECT)               *
*                       R4 = -                                     *
*                       R5 = -                                     *
*                       R6 = -                                     *
*                       R7 = -                                     *
*                       R8 = CDCDA & CDCDD (QW0185)                *
*                       R9 = ANCHOR AREA   (ANCHOR)                *
*                       R10= HUP EXTERNAL CB (HEC)                 *
*                       R11= -                                     *
*                       R12= MODULE BASE REGISTER                  *
*                       R13= ADDRESS OF SAVE-AREA (WRK)            *
*                       R14= WORK / LINKAGE                        *
*                       R15= WORK / LINKAGE                        *
*                                                                  *
*       PATCH LABEL  = NONE                                        *
*                                                                  *
*    MODULE TYPE = PROCEDURE                                       *
*       PROCESSOR = ASSEMBLER                                      *
*       MODULE SIZE = APPROXIMATELY 2000 BYTES                     *
*       ATTRIBUTES = REENTRANT                                     *
*       RMODE      = ANY                                           *
*       AMODE      = 31                                            *
*                                                                  *
*    ENTRY POINT = EKYEDB2A                                        *
*       PURPOSE = SEE FUNCTION                                     *
*       LINKAGE = STANDARD OS/VS ASSEMBLER LINKAGE CONVENTIONS.    *
*                                                                  *
*                                                                  *
*    INPUT:                                                        *
*         - REGISTER 15 = ENTRY POINT ADDRESS                      *
*         - REGISTER 14 = RETURN ADDRESS                           *
*         - REGISTER 13 = ADDRESS OF SAVEAREA                      *
*         - REGISTER 1  = ADDRESS OF STANDARD PARAMETER LIST:      *
*                         1. PARAMETER - ADDRESS OF A 64 BYTE      *
*                                        ANCHOR AREA               *
*                         2. PARAMETER - ADDRESS OF THE HUP        *
*                                        EXTERNAL INTERFACE (HEC)  *
*                                                                  *
*                                                                  *
*    OUTPUT:                                                       *
*         - THE MIRROR TABLE 'TABLE0M' HAS BEEN UPDATED IF THE     *
*           CURRENT APPLICATION PROGRAM CHANGED THE ORIGINATING    *
*           TABLE 'TABLE02'.                                       *
*                                                                  *
*                                                                  *
*    EXIT-NORMAL=                                                  *
*       STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.               *
*       RETURN-CODES= 0                                            *
*                                                                  *
*    EXIT-ERROR= NONE                                              *
*                                                                  *
*                                                                  *
*    ABEND-CODE OF EKYEDB2A = NONE                                 *
*                                                                  *
*    ERROR-MESSAGES ISSUED BY EKYEDB2A:                            *
*                                                                  *
*       EKYEDB1E  INVALID OPERATION CODE IN CDC DATA DEFINITION    *
*       EKYEDB2E  ** MESSAGE RETURNED BY DSNTIAR AFTER SQL ERROR ** *
*       EKYEDB3E  UNEXPECTED COLUMN DATA TYPE ENCOUNTERED          *
*       EKYEDB4E  EXPECTED COLUMN NOT PASSED IN CDCDD              *
*       EKYEDB5I  COLUMN IN ERROR: ** COLUMN NAME **               *
*                                                                  *
```

Figure 74 (Part 3 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
*                                                                      *
*    EXTERNAL REFERENCES                                               *
*                                                                      *
*       ROUTINES       = SQL LANGUAGE INTERFACE                        *
*                                                                      *
*       DATA-AREAS     = SEE CONTROL BLOCKS                            *
*                                                                      *
*       CONTROL BLOCKS = WRK       MODULE OWN WORKAREA                 *
*                        HOSTTAB   HOST VARIABLE MAPPING TABLE         *
*                        HEC       HUP EXTERNAL INTERFACE              *
*                        ANCHOR    ANCHOR AREA PASSED BY HUP           *
*                        SQLCA     DB2 SQL COMMUNICATION AREA          *
*                        SQLDSECT  DB2 SQL WORK AREA                   *
*                        DSNDQWHS  IFI STANDARD HEADER AREA            *
*                        DSNDQWHC  IFI CORRELATION DATA AREA           *
*                        DSNDQW02  IFI IFCID 140 UP RECORDS            *
*                                                                      *
*    MACROS CODED IN MODULE= NONE                                      *
*                                                                      *
*    MACROS USED FROM MACRO-LIBRARY=                                   *
*        SAVE     - SAVE REGISTERS                                     *
*        GETMAIN  - OS/VS GETMAIN                                      *
*        WTO      - WRITE TO OPERATOR                                  *
*        LINK     - DYNAMIC PROGRAM CALL                               *
*        RETURN   - RETURN TO CALLING PROGRAM                          *
*        EKYHCHEC - MAPPING OF HUP EXTERNAL INTERFACE                  *
*        DSNDQWHS - MAPPING OF IFI STANDARD HEADER AREA                *
*        DSNDQWHC - MAPPING OF IFI CORRELATION DATA AREA               *
*        DSNDQW02 - MAPPING OF IFI IFCID 140 UP RECORDS                *
*                                                                      *
*    TABLES=          NONE                                             *
*                                                                      *
*    INCLUDE CODE FROM LIBRARY= NONE                                   *
*                                                                      *
*                                                                      *
*    CHANGE ACTIVITY=                                                  *
*                                                                      *
****************** END OF SPECIFICATIONS ******************************
****************** LOGIC OF EKYEDB2A **********************************
*                                                                      *
*                                                                      *
* (1)    PROGRAM PROLOG                                                *
*                                                                      *
*              - EXECUTE CSECT AND AMODE/RMODE DECLARATIONS            *
*              - GENERATE SAVE-ID WITH EXITNAME AND COMPILE TIMESTAMP *
*              - SAVE REGISTERS AND ESTABLISH MODULE ADDRESSABILITY   *
*              - POINT TO PASSED PARAMETERS                            *
*                                                                      *
*              - IF THIS IS THE FIRST INVOCATION                       *
*                - GETMAIN AN AREA CONTAINING                          *
*                  -- OUR SAVEAREA                                     *
*                  -- MODULE WORKSPACE                                 *
*                - CLEAR THE GETMAINED AREA                            *
*                                                                      *
*              - CHAIN SAVEAREAS AND ESTABLISH ADDRESSABILITY OF WA   *
*                                                                      *
*                                                                      *
* (2)    EXECUTE UPDATE OF THE MIRROR TABLE                            *
*                                                                      *
*              - ADDRESS CDCDD AND ANALYZE IF THIS IS THE TABLE        *
*                WE ARE LOOKING FOR (TABLE02)                          *
*                                                                      *
*              - SETUP OLD KEY FIELD VALUES FOR UPDATE OPERATIONS      *
*                                                                      *
*              - SETUP NEW FIELD VALUES FOR ANY OPERATION              *
*                                                                      *
```

Figure 74 (Part 4 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
*             - ANALYZE OPERATION CODE AND BRANCH ACCORDINGLY       *
*                                                                   *
*             - IF THERE IS AN INVALID OPERATION CODE               *
*               - ISSUE WTO TO INFORM OPERATOR                      *
*               - RETURN TO CALLING PROGRAM                         *
*                                                                   *
*             - IF OPERATION WAS 'INSERT'                           *
*               - INSERT ROW IN MIRROR TABLE USING NEW VALUES       *
*                                                                   *
*             - IF OPERATION WAS 'UPDATE'                           *
*               - UPDATE THE ROW IN MIRROR TABLE USING OLD KEYFIELD *
*                 VALUES IN THE WHERE CLAUSE                        *
*                                                                   *
*             - IF OPERATION WAS 'DELETE'                           *
*               - DELETE THE ROW USING NEW KEYFIELD VALUES IN       *
*                 THE WHERE CLAUSE                                  *
*                                                                   *
*                                                                   *
* (3)    CHECK RESULT OF MIRROR TABLE UPDATE                        *
*                                                                   *
*             - CHECK THE RESULTING SQL CODE                        *
*                                                                   *
*             - IF UPDATE OF MIRROR TABLE WAS SUCCESSFUL            *
*               - CONTINUE WITH RETURN TO CALLING PROGRAM           *
*                                                                   *
*             - IF MIRROR TABLE UPDATE FAILED                       *
*               - EXECUTE THE SQL ERROR LOGIC                       *
*                                                                   *
*                                                                   *
* (4)    IF SQL ERROR OCCURED                                       *
*                                                                   *
*             - PREPARE PARAMETER LIST FOR DSNTIAR                  *
*             - CALL DSNTIAR GO GET FORMATTED SQL ERROR MESSAGE     *
*             - WTO ANY NON-BLANK MESSAGE LINE RETURNED BY DSNTIAR  *
*             - CONTINUE WITH RETURN PROCESSING                     *
*                                                                   *
*                                                                   *
* (5)    RETURN PROCESSING                                          *
*                                                                   *
*             - RELOAD REGISTER AND RETURN TO DPROP                 *
*                                                                   *
*                                                                   *
****************** END-OF-LOGIC ***************************************
*********************************************************************
*                                                                   *
*        ASSEMBLER DCLGEN FOR TABLE TABLE0M                         *
*                                                                   *
*********************************************************************

*------- TABLE DECLARATION FOR TABLE TABLE0M
***$$$
*        EXEC  SQL                                                   *
               DECLARE TABLE0M TABLE                                 *
               (                                                     *
                 KEYFLD1              CHAR(2)              NOT NULL *
               , KEYFLD2              CHAR(6)              NOT NULL *
               , FAMILY               VARCHAR(30)                    *
               , FIRST                VARCHAR(20)                    *
               , CITY                 VARCHAR(35)                    *
               )
***$$$
*********************************************************************
*                                                                   *
*        MODULE WORKAREA DEFINTION                                  *
*                                                                   *
*********************************************************************
```

Figure 74 (Part 5 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                       326 *---------------------------------------------------------------------*
                                       327 *        MODULE OWN SAVEAREA - MUST PREFIX THE WORKAREA               *
                                       328 *---------------------------------------------------------------------*

000000                                 330 WRK      DSECT ,                        ENTER DSECT DECLARATION
000000 0000000000000000                331 SAVEAREA DC    18F'0'                   MODULE OWN WORKAREA

                                       333 *---------------------------------------------------------------------*
                                       334 *        DEFINITION OF SQL HOST VARIABLES                             *
                                       335 *---------------------------------------------------------------------*

                                       337 *------- FIELD DEFINITIONS FOR TABLE TABLE0M
000048                                 338 NEW_KEYFLD1 DS    CL2                   CHAR(2)         (NOT NULL)
                              00002    339 LEN_KEYFLD1 EQU   *-NEW_KEYFLD1
00004A                                 340 NEW_KEYFLD2 DS    CL6                   CHAR(6)         (NOT NULL)
                              00006    341 LEN_KEYFLD2 EQU   *-NEW_KEYFLD2
000050                                 342 NEW_FAMILY  DS    H,CL30                VARCHAR(30)
                              00020    343 LEN_FAMILY  EQU   *-NEW_FAMILY
000070                                 344 NEW_FIRST   DS    H,CL20                VARCHAR(20)
                              00016    345 LEN_FIRST   EQU   *-NEW_FIRST
000086                                 346 NEW_CITY    DS    H,CL35                VARCHAR(35)
                              00025    347 LEN_CITY    EQU   *-NEW_CITY

                                       349 *------- NULL INDICATORS FOR TABLE0M
0000AC                                 350 IND_FAMILY  DS    H                     VARCHAR(30)
0000AE                                 351 IND_FIRST   DS    H                     VARCHAR(20)
0000B0                                 352 IND_CITY    DS    H                     VARCHAR(35)

                                       354 *------- OLD KEY FIELD DEFINITIONS FOR TABLE TABLE0M
0000B2                                 355 OLD_KEYFLD1 DS    CL2                   CHAR(2)         (NOT NULL)
0000B4                                 356 OLD_KEYFLD2 DS    CL6                   CHAR(6)         (NOT NULL)

                                       358 *---------------------------------------------------------------------*
                                       359 *        AREA USED TO ISSUE ERROR MESSAGES                            *
                                       360 *---------------------------------------------------------------------*

0000BA 0000000000000000                362 WRKWTO   DC    XL(WTODSNTL)'0'          AREA FOR WTO PARMLIST COPY
                              000C7    363 WRKWTOTM EQU   WRKWTO+4+9,110           DEFINITION OF INSERTED TEXT

                                       365 *---------------------------------------------------------------------*
                                       366 *        AREA USED TO INVOKE DSNTIAR MESSAGE FORMATTER                *
                                       367 *---------------------------------------------------------------------*

000138                                 369 WRKDSNT  DS    0F                       DSNTIAR PARMLIST
000138 00000000                        370 WRKDSNT1 DC    A(*-*)                   - ADDRESS OF SQLCA
00013C 00000000                        371 WRKDSNT2 DC    A(*-*)                   - ADDRESS OF WRKMSG
000140 80000000                        372 WRKDSNT3 DC    A(*-*+X'80000000')       - ADDRESS OF LINE LENGTH
000144                                 373 WRKMSG   DC    0F'0'                    DSNTIAR MESSAGE AREA
000144 044C                            374 WRKMSGL  DC    AL2(10*L'WRKMSG1)        LENGTH OF MESSAGE AREA
000146 4040404040404040                375 WRKMSG1  DC    CL110' '                 MESSAGE LINE 1
0001B4 4040404040404040                376 WRKMSG2  DC    CL110' '                 MESSAGE LINE 2
000222 4040404040404040                377 WRKMSG3  DC    CL110' '                 MESSAGE LINE 3
000290 4040404040404040                378 WRKMSG4  DC    CL110' '                 MESSAGE LINE 4
0002FE 4040404040404040                379 WRKMSG5  DC    CL110' '                 MESSAGE LINE 5
00036C 4040404040404040                380 WRKMSG6  DC    CL110' '                 MESSAGE LINE 6
0003DA 4040404040404040                381 WRKMSG7  DC    CL110' '                 MESSAGE LINE 7
000448 4040404040404040                382 WRKMSG8  DC    CL110' '                 MESSAGE LINE 8
0004B6 4040404040404040                383 WRKMSG9  DC    CL110' '                 MESSAGE LINE 9
000524 4040404040404040                384 WRKMSG10 DC    CL110' '                 MESSAGE LINE 10

                                       386 *---------------------------------------------------------------------*
                                       387 *        SQL COMMUNICATION AREA                                       *
                                       388 *---------------------------------------------------------------------*

                                       390 ***$$$
```

Figure 74 (Part 6 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                          391 *        EXEC  SQL                                                    *
                                                             INCLUDE                                                *
                                                               SQLCA
                                          392 ***$$$ SQLCA
000594                                    393 SQLCA    DS    0F
000594                                    394 SQLCAID  DS    CL8      ID
00059C                                    395 SQLCABC  DS    F        BYTE COUNT
0005A0                                    396 SQLCODE  DS    F        RETURN CODE
0005A4                                    397 SQLERRM  DS    H,CL70   ERR MSG PARMS
0005EC                                    398 SQLERRP  DS    CL8      IMPL-DEPENDENT
0005F4                                    399 SQLERRD  DS    6F
00060C                                    400 SQLWARN  DS    0C       WARNING FLAGS
00060C                                    401 SQLWARN0 DS    C'W' IF ANY
00060D                                    402 SQLWARN1 DS    C'W' = WARNING
00060E                                    403 SQLWARN2 DS    C'W' = WARNING
00060F                                    404 SQLWARN3 DS    C'W' = WARNING
000610                                    405 SQLWARN4 DS    C'W' = WARNING
000611                                    406 SQLWARN5 DS    C'W' = WARNING
000612                                    407 SQLWARN6 DS    C'W' = WARNING
000613                                    408 SQLWARN7 DS    C'W' = WARNING
000614                                    409 SQLEXT   DS    CL8
00061C                            00614   410          ORG   SQLEXT
000614                                    411 SQLWARN8 DS    C
000615                                    412 SQLWARN9 DS    C
000616                                    413 SQLWARNA DS    C
000617                                    414 SQLSTATE DS    CL5
00061C                            0061C   415          ORG
                                          416 ***$$$

                                          418 *-------------------------------------------------------------------*
                                          419 *        SQL WORKAREA DEFINITION                                    *
                                          420 *-------------------------------------------------------------------*

00061C 0000000000000000                   422 SQLWA    DC    XL(SQLDLEN)'0'          SQL WORKAREA DEFINITION

                                          424 *-------------------------------------------------------------------*
                                          425 *        END OF WORKAREA DEFINITION                                 *
                                          426 *-------------------------------------------------------------------*

                                  006BC   428 WRKLEN   EQU   *-WRK                   LENGTH OF WHOLE WORKAREA
                                          430 *********************************************************************
                                          431 *                                                                   *
                                          432 *        DEFINTION OF THE PASSED ANCHOR AREA                        *
                                          433 *                                                                   *
                                          434 *********************************************************************

                                          436 *-------------------------------------------------------------------*
                                          437 *        DEFINE ANCHOR AREA AS USED BY OUR EXIT                     *
                                          438 *-------------------------------------------------------------------*

000000                                    440 ANCHOR   DSECT ,                       ENTER DSECT DECLARATION
000000 00000000                           441 ANC_PTR  DC    A(*-*)                  ADDRESS OF MODULE WORKAREA
000004 0000000000000000                   442 ANC_RES  DC    15F'0'                  RESERVED
                                          444 *********************************************************************
                                          445 *                                                                   *
                                          446 * (1)    PROGRAM PROLOG                                             *
                                          447 *                                                                   *
                                          448 *              - EXECUTE CSECT AND AMODE/RMODE DECLARATIONS         *
                                          449 *              - GENERATE SAVE-ID WITH EXITNAME AND COMPILE TIMESTAMP *
                                          450 *              - SAVE REGISTERS AND ESTABLISH MODULE ADDRESSABILITY   *
                                          451 *              - POINT TO PASSED PARAMETERS                         *
                                          452 *                                                                   *
                                          453 *              - IF THIS IS THE FIRST INVOCATION                    *
                                          454 *                - GETMAIN AN AREA CONTAINING                       *
                                          455 *                  -- OUR SAVEAREA                                  *
                                          456 *                  -- MODULE WORKSPACE                              *
                                          457 *                - CLEAR THE GETMAINED AREA                         *
```

Figure 74 (Part 7 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                   458 *                                                                   *
                                   459 *             - CHAIN SAVEAREAS AND ESTABLISH ADDRESSABILITY OF WA   *
                                   460 *                                                                   *
                                   461 *********************************************************************

                                   463 *-------------------------------------------------------------------*
                                   464 *        EXECUTE CSECT AND AMODE/RMODE DECLARATIONS                 *
                                   465 *-------------------------------------------------------------------*

000000                             467 EKYEDB2A CSECT ,                        ENTER CSECT OF SUBEXIT ROUTINE
                                   468 EKYEDB2A AMODE 31                       EXIT IS CALLED IN AMODE 31
                                   469 EKYEDB2A RMODE ANY                      EXIT CAN BE LOADED ANYWHERE

                                   471 *-------------------------------------------------------------------*
                                   472 *        GENERATE SAVE-ID WITH EXITNAME AND COMPILATION DATE AND TIME *
                                   473 *-------------------------------------------------------------------*

                                   475          LCLC  &SAVEID                  DEFINE LOCAL CHAR VARIABLE
                                   476 &SAVEID  SETC  'EKYEDB2A DPR120'.'-'.'&SYSDATE'.'-'.'&SYSTIME'

                                   478 *-------------------------------------------------------------------*
                                   479 *        SAVE REGISTERS AND ESTABLISH MODULE ADDRESSABILITY          *
                                   480 *-------------------------------------------------------------------*

                                   482          SAVE  (14,12),,&SAVEID         DEFINE ID-BLOCK AND SAVE REGS
000028 18CF                        490          LR    R12,R15                  GET ENTRY POINT IN BASE REG
                            00000  491          USING EKYEDB2A,R12             ESTABLISH BASE ADDRESSABILITY

                                   493 *-------------------------------------------------------------------*
                                   494 *        LETS POINT TO PASSED PARAMETERS                            *
                                   495 *-------------------------------------------------------------------*

00002A 989A 1000            00000  497          LM    R9,R10,0(R1)             GET POINTER TO PARAMETERS
                            00000  498          USING ANCHOR,R9                DECLARE ANCHOR STORAGE ADDRESS.
                            00000  499          USING HEC,R10                  DECLARE HEC ADDRESSABILITY

                                   501 *-------------------------------------------------------------------*
                                   502 *        IF THIS IS THE FIRST INVOCATION                            *
                                   503 *                                                                   *
                                   504 *              - GETMAIN AN AREA CONTAINING                         *
                                   505 *                -- OUR SAVEAREA                                    *
                                   506 *                -- MODULE WORKSPACE                                *
                                   507 *              - CLEAR THE GETMAINED AREA                           *
                                   508 *-------------------------------------------------------------------*

00002E 5820 9000            00000  510          L     R2,ANC_PTR               GET ADDRESS OF GETMAINED AREA
000032 1222                        511          LTR   R2,R2                    IS AREA ALREADY GETMAINED?
000034 4770 C05C            0005C  512          BNZ   CHAINING                 YES -> THEN USE FROM PREV. CALL
000038 5800 C6C8            006C8  513          L     R0,=A(WRKLEN)            NO  -> GET LENGTH OF AREA
                                   514          GETMAIN RU,                    THEN ISSUE OS/VS GETMAIN       *
                                                      LV=(0),                  TO ACQUIRE OUR MODULE SAVE     *
                                                      LOC=ANY                  AND WORK AREA
00004C 1821                        524          LR    R2,R1                    GET AREA ADDRESS IN CORRECT REG
00004E 5020 9000            00000  525          ST    R2,ANC_PTR               SAVE FOR NEXT CALL OF EXIT
000052 1801                        526          LR    R0,R1                    GET START ADDRESS OF AREA
000054 5810 C6C8            006C8  527          L     R1,=A(WRKLEN)            GET LENGTH OF WHOLE AREA
000058 17FF                        528          XR    R15,R15                  CLEAR SOURCE LEN AND PAD BYTE
00005A 0E0E                        529          MVCL  R0,R14                   AND MOVE BINARY ZEROES TO AREA

                                   531 *-------------------------------------------------------------------*
                                   532 *        CHAIN SAVEAREAS AND ESTABLISH ADDRESSABILITY OF WORKAREA    *
                                   533 *-------------------------------------------------------------------*
```

Figure 74 (Part 8 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
00005C                          535 CHAINING DS    0H
00005C 5020 D008         00008  536          ST    R2,8(0,R13)            FORWARD  CHAIN OUR SAVEAREA
000060 50D0 2004         00004  537          ST    R13,4(0,R2)            BACKWARD CHAIN THE PASSED ONE
000064 18D2                     538          LR    R13,R2                 SETUP CORRECT SAVEAREA POINTER
                         00000  539          USING WRK,R13                DECLARE WORKAREA ADDRESSABILITY
                                541 ***********************************************************************
                                542 *                                                                     *
                                543 *        EXPLANATIONS ABOUT PASSED DATA                               *
                                544 *                                                                     *
                                545 *              AT THIS POINT REGISTER 10 POINTS TO THE HUP            *
                                546 *              EXTERNAL CONTROL BLOCK (HEC). THE HEC CONTAINS         *
                                547 *              ITSELF POINTERS TO THE CHANGED DATA CAPTURE DATA       *
                                548 *              WHICH WAS RETRIEVED BY DPROP USING IFI CALL 185:       *
                                549 *                                                                     *
                                550 *                - HECQWHS   POINTS TO THE IFI STANDARD HEADER AREA.  *
                                551 *                            THIS AREA IS MAPPED BY THE DSNDQWHS      *
                                552 *                            MACRO. IF DATA FROM THIS CONTROL BLOCK   *
                                553 *                            IS NEEDED, THE FOLLOWING INSTRUCTIONS    *
                                554 *                            CAN BE USED TO ESTABLISH ADDRESSABILITY  *
                                555 *                            OF IT:                                   *
                                556 *                                       L     RX,HECQWHS              *
                                557 *                                       USING QWHS,RX                 *
                                558 *                                       ...                           *
                                559 *                                                                     *
                                560 *                - HECQWHC   POINTS TO THE IFI CORRELATION DATA AREA. *
                                561 *                            THIS AREA IS MAPPED BY THE DSNDQWHC      *
                                562 *                            MACRO. IF DATA FROM THIS CONTROL BLOCK   *
                                563 *                            IS NEEDED, THE FOLLOWING INSTRUCTIONS    *
                                564 *                            CAN BE USED TO ESTABLISH ADDRESSABILITY  *
                                565 *                            OF IT:                                   *
                                566 *                                       L     RX,HECQWHC              *
                                567 *                                       USING QWHC,RX                 *
                                568 *                                       ...                           *
                                569 *                                                                     *
                                570 *                - HECCDCDD  POINTS TO THE DB2 CHANGED DATA CAPTURE   *
                                571 *                            DATA DEFININTION (CDCDD). DPROP WILL     *
                                572 *                            ALWAYS PASS A DATA DEFINITION OF THE     *
                                573 *                            MODIFIED TABLE TO YOUR CHANGED DATA      *
                                574 *                            CAPTURE SUBEXIT ROUTINE. THIS AREA       *
                                575 *                            IS MAPPED BY THE QW0185 DSECT IN THE     *
                                576 *                            DSNDQW02 MACRO. AFTER A PREFIX COMMON    *
                                577 *                            TO CDCDD AND CDCDA (SEE BELOW) THIS      *
                                578 *                            AREA CONTAINS A DESCRIPTION OF THE       *
                                579 *                            COLUMNS IN THE TABLE, WHICH IS IN A      *
                                580 *                            SIMILAR MANNER AS IN THE STANDARD        *
                                581 *                            EXTERNAL SQLDA. NOTE, THAT QQ0195SI      *
                                582 *                            CONTAINS THE OFFSET OF THE COLUMN        *
                                583 *                            WITHIN THE DATA ROW (CDCDA) AND THAT     *
                                584 *                            THE LENGTH OF GRAPHIC AND VARGRAPHIC     *
                                585 *                            FIELDS IS SPECIFIED IN NUMBER OF BYTES   *
                                586 *                            (IN OPPOSITION TO THE SQLDA WHICH        *
                                587 *                            CONTAINS THE NUMBER OF DOUBLE BYTES).    *
                                588 *                                                                     *
                                589 *                - HECCDCDA  POINTS TO THE FIRST OR ONLY DATA ROW     *
                                590 *                            OF THE CHANGED DATA CAPTURE DATA ROW     *
                                591 *                            (CDCDA). THIS AREA IS ALWAYS PASSED      *
                                592 *                            TO YOUR EXIT ROUTINE AND IT WILL         *
                                593 *                            CONTAIN EITHER THE ONLY IMAGE OF THE     *
                                594 *                            ROW (FOR INSERT OR DELETE OPERATIONS)    *
                                595 *                            OR THE AFTER IMAGE (FOR UPDATES).        *
                                596 *                            THIS AREA IS ALSO MAPPED BY THE QW0185   *
                                597 *                            DSECT OF THE DSNDQW02 MACRO. AFTER A     *
                                598 *                            PREFIX COMMON TO CDCDA AND CDCDD (SEE    *
                                599 *                            ABOVE), THIS AREA CONTAINS THE COLUMN    *
                                600 *                            VALUES OF THE AFFECTED ROW.              *
                                601 *                                                                     *
```

Figure 74 (Part 9 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                        *               - HECCDCDB  POINTS TO THE CHANGED DATA CAPTURE       *
                                        *                           DATA ROW (CDCDA). THIS AREA CONTAINS     *
                                        *                           THE BEFORE IMAGE OF THE AFFECTED ROW,    *
                                        *                           AND IS THEREFORE ONLY PRESENT IF THE     *
                                        *                           ORIGINATING SQL CALL WAS AN UPDATE.      *
                                        *                           THIS AREA IS ALSO MAPPED BY THE QW0185   *
                                        *                           DSECT OF THE DSNDQW02 MACRO. AFTER A     *
                                        *                           PREFIX COMMON TO CDCDA AND CDCDD (SEE    *
                                        *                           ABOVE), THIS AREA CONTAINS THE COLUMN    *
                                        *                           VALUES OF THE AFFECTED ROW.              *
                                        *                                                                    *
                                        **********************************************************************
                                        **********************************************************************
                                        *                                                                    *
                                        * (2)    EXECUTE FETCH OF USED HOST VARIABLES                        *
                                        *                                                                    *
                                        *              - ADDRESS CDCDD AND ANALYZE IF THIS IS THE TABLE      *
                                        *                WE ARE LOOKING FOR (TABLE02)                        *
                                        *                                                                    *
                                        *              - SETUP OLD KEY FIELD VALUES FOR UPDATE OPERATIONS    *
                                        *                                                                    *
                                        *              - SETUP NEW FIELD VALUES FOR ANY OPERATION            *
                                        *                                                                    *
                                        *              - ANALYZE OPERATION CODE AND BRANCH ACCORDINGLY       *
                                        *                                                                    *
                                        *              - IF THERE IS AN INVALID OPERATION CODE               *
                                        *                - ISSUE WTO TO INFORM OPERATOR                      *
                                        *                - RETURN TO CALLING PROGRAM                         *
                                        *                                                                    *
                                        *              - IF OPERATION WAS 'INSERT'                           *
                                        *                - INSERT ROW IN MIRROR TABLE USING THE NEW VALUES   *
                                        *                                                                    *
                                        *              - IF OPERATION WAS 'UPDATE'                           *
                                        *                - UPDATE THE ROW IN MIRROR TABLE USING OLD KEYFIELD *
                                        *                  VALUES IN THE WHERE CLAUSE                        *
                                        *                                                                    *
                                        *              - IF OPERATION WAS 'DELETE'                           *
                                        *                - DELETE THE ROW USING NEW KEYFIELD VALUES IN       *
                                        *                  THE WHERE CLAUSE                                  *
                                        *                                                                    *
                                        **********************************************************************

                                        *--------------------------------------------------------------------*
                                        *        ADDRESS CDCDD AND ANALYZE IF REALLY TABLE02 IN PROCESS      *
                                        *--------------------------------------------------------------------*

000066 5880 A018            00018                L     R8,HECCDCDD              POINT CDCDD PASSED BY DPROP
                            00000                USING QW0185,R8                DECLARE CDCDD ADDRESSABILITY
00006A D511 8014 C6E8 00014 006E8                CLC   QW0185TB,=CL18'TABLE02'  IS THIS THE SEARCHED TABLE?
000070 4770 C3DA            003DA                BNE   RETURN                   NO -> THEN SKIP PROCESS

                                        *--------------------------------------------------------------------*
                                        *        SETUP OLD KEY FIELD VALUES FOR UPDATE OPERATIONS            *
                                        *--------------------------------------------------------------------*

000074 5820 A020            00020                L     R2,HECCDCDB              GET POINTER TO BEFORE IMAGE
000078 1222                                      LTR   R2,R2                    IS THERE A BEFORE IMAGE?
00007A 4780 C086            00086                BZ    SETAFTER                 NO -> SET ONLY AFTER IMAGE
00007E 4130 C494            00494                LA    R3,COLBTAB               POINT BEFORE IMAGE TABLE
000082 4DB0 C3E8            003E8                BAS   R11,SETHOST              AND SETUP HOST VARIABLES

                                        *--------------------------------------------------------------------*
                                        *        SETUP NEW FIELD VALUES (FOR ALL OPERATIONS)                 *
                                        *--------------------------------------------------------------------*
```

Figure 74 (Part 10 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
000086                          668 SETAFTER DS    0H
000086 5820 A01C           0001C  669          L     R2,HECCDCDA              GET POINTER TO AFTER IMAGE
00008A 4130 C4D0           004D0  670          LA    R3,COLATAB               POINT AFTER IMAGE TABLE
00008E 4DB0 C3E8           003E8  671          BAS   R11,SETHOST              AND SETUP HOST VARIABLES

                                  673 *---------------------------------------------------------------------*
                                  674 *        ANALYZE OPERATION CODE AND BRANCH ACCORDINGLY                *
                                  675 *---------------------------------------------------------------------*

000092                          677 ANALYZE  DS    0H
000092 5880 A01C           0001C  678          L     R8,HECCDCDA              GET POINTER TO AFTER IMAGE
000096 4130 D61C           0061C  679          LA    R3,SQLWA                 POINT SQL WORK AREA
                           00000  680          USING SQLDSECT,R3              DECLARE SQLDSECT ADDRESSABILITY
00009A D501 804A C6FA 0004A 006FA  681          CLC   QW0185PC,=C'IN'          IS IT AN INSERT OPERATION?
0000A0 4780 C0C2           000C2  682          BE    EXECISRT                 YES -> INSERT ROW IN MIRROR TAB
0000A4 D501 804A C6FC 0004A 006FC  683          CLC   QW0185PC,=C'UA'          IS IT AN UPDATE OPERATION?
0000AA 4780 C1AC           001AC  684          BE    EXECUPD                  YES -> UPDATE ROW IN MIRROR TAB
0000AE D501 804A C6FE 0004A 006FE  685          CLC   QW0185PC,=C'DE'          IS IT AN DELETE OPERATION?
0000B4 4780 C2CA           002CA  686          BE    EXECDEL                  YES -> DELETE ROW IN MIRROR TAB

                                  688 *---------------------------------------------------------------------*
                                  689 *        THERE IS AN INVALID OPERATION CODE - ISSUE WTO AND RETURN    *
                                  690 *---------------------------------------------------------------------*

                                  692          WTO   MF=(E,WTOERROP)          ISSUE OPERATION CODE ERROR WTO
0000BE 47F0 C3DA           003DA  695          B     RETURN                   AND SKIP UPDATE OF MIRROR TAB

                                  697 *---------------------------------------------------------------------*
                                  698 *        UPDATE MIRROR TABLE FOR AN INSERT OPERATION                  *
                                  699 *---------------------------------------------------------------------*

0000C2                          701 EXECISRT DS    0H
                                  702 ***$$$
                                  703 *        EXEC  SQL                                                      *
                                                     INSERT                                                   *
                                                     INTO                                                     *
                                                       TABLE0M                                                *
                                                       (                                                      *
                                                       KEYFLD1                                                *
                                                     , KEYFLD2                                                *
                                                     , FAMILY                                                 *
                                                     , FIRST                                                  *
                                                     , CITY                                                   *
                                                       )                                                      *
                                                       VALUES                                                 *
                                                       (                                                      *
                                                       :NEW_KEYFLD1                                           *
                                                     , :NEW_KEYFLD2                                           *
                                                     , :NEW_FAMILY:IND_FAMILY                                 *
                                                     , :NEW_FIRST:IND_FIRST                                   *
                                                     , :NEW_CITY:IND_CITY                                     *
                                                       )
0000C2 47F0 C0E2           000E2  704          B     *+32
0000C6 00288000001E               705          DC    H'40',X'8000',H'30'
0000CC E740404040404040           706          DC    CL8'X       ',XL8'14E73D270DA01CB4',H'1'
0000DE 029400E8                   707          DC    H'660,232'
0000E2 D217 3004 C0C6 00004 000C6  708          MVC   SQLPLLEN(24),*-28
0000E8 D203 3028 C0DE 00028 000DE  709          MVC   SQLSTNUM(4),*-10
0000EE 41F0 D594           00594  710          LA    15,SQLCA
0000F2 50F0 301C           0001C  711          ST    15,SQLCODEP
0000F6 41F0 D048           00048  712          LA    15,NEW_KEYFLD1
0000FA 50F0 3034           00034  713          ST    15,SQLPVARS+8
0000FE D201 3030 C700 00030 00700  714          MVC   SQLPVARS+4(2),=X'01C4'
000104 D201 3032 C702 00032 00702  715          MVC   SQLPVARS+6(2),=H'2'
00010A 1FFF                       716          SLR   15,15
00010C 50F0 3038           00038  717          ST    15,SQLPVARS+12
```

Figure 74 (Part 11 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
000110 41F0 D04A           0004A   718         LA    15,NEW_KEYFLD2
000114 50F0 3040           00040   719         ST    15,SQLPVARS+20
000118 D201 303C C700 0003C 00700  720         MVC   SQLPVARS+16(2),=X'01C4'
00011E D201 303E C704 0003E 00704  721         MVC   SQLPVARS+18(2),=H'6'
000124 1FFF                        722         SLR   15,15
000126 50F0 3044           00044   723         ST    15,SQLPVARS+24
00012A 41F0 D050           00050   724         LA    15,NEW_FAMILY
00012E 50F0 304C           0004C   725         ST    15,SQLPVARS+32
000132 D201 3048 C706 00048 00706  726         MVC   SQLPVARS+28(2),=X'01C1'
000138 D201 304A C708 0004A 00708  727         MVC   SQLPVARS+30(2),=H'30'
00013E 41F0 D0AC           000AC   728         LA    15,IND_FAMILY
000142 50F0 3050           00050   729         ST    15,SQLPVARS+36
000146 41F0 D070           00070   730         LA    15,NEW_FIRST
00014A 50F0 3058           00058   731         ST    15,SQLPVARS+44
00014E D201 3054 C706 00054 00706  732         MVC   SQLPVARS+40(2),=X'01C1'
000154 D201 3056 C70A 00056 0070A  733         MVC   SQLPVARS+42(2),=H'20'
00015A 41F0 D0AE           000AE   734         LA    15,IND_FIRST
00015E 50F0 305C           0005C   735         ST    15,SQLPVARS+48
000162 41F0 D086           00086   736         LA    15,NEW_CITY
000166 50F0 3064           00064   737         ST    15,SQLPVARS+56
00016A D201 3060 C706 00060 00706  738         MVC   SQLPVARS+52(2),=X'01C1'
000170 D201 3062 C70C 00062 0070C  739         MVC   SQLPVARS+54(2),=H'35'
000176 41F0 D0B0           000B0   740         LA    15,IND_CITY
00017A 50F0 3068           00068   741         ST    15,SQLPVARS+60
00017E D203 302C C6CC 0002C 006CC  742         MVC   SQLPVARS(4),=F'64'
000184 41F0 302C           0002C   743         LA    15,SQLPVARS
000188 50F0 3020           00020   744         ST    15,SQLVPARM
00018C D203 3024 C6D0 00024 006D0  745         MVC   SQLAPARM,=XL4'00000000'
000192 4110 3004           00004   746         LA    1,SQLPLLEN
000196 5010 3000           00000   747         ST    1,SQLPLIST
00019A 9680 3000      00000        748         OI    SQLPLIST,X'80'
00019E 4110 3000           00000   749         LA    1,SQLPLIST
0001A2 58F0 C6D4           006D4   750         L     15,=V(DSNHLI)
0001A6 05EF                        751         BALR  14,15
                                   752 ***$$$
0001A8 47F0 C360           00360   753         B     CHECKSQL

                                   755 *-------------------------------------------------------------------*
                                   756 *       UPDATE MIRROR TABLE FOR AN UPDATE OPERATION                 *
                                   757 *-------------------------------------------------------------------*

0001AC                             759 EXECUPD  DS    0H
                                   760 ***$$$
                                   761 *        EXEC  SQL                                                    *
                                                      UPDATE                                                 *
                                                        TABLE0M                                              *
                                                      SET                                                    *
                                                        KEYFLD1  = :NEW_KEYFLD1                              *
                                                      , KEYFLD2  = :NEW_KEYFLD2                              *
                                                      , FAMILY   = :NEW_FAMILY:IND_FAMILY                    *
                                                      , FIRST    = :NEW_FIRST:IND_FIRST                      *
                                                      , CITY     = :NEW_CITY:IND_CITY                        *
                                                      WHERE                                                  *
                                                        KEYFLD1  = :OLD_KEYFLD1  AND                         *
                                                        KEYFLD2  = :OLD_KEYFLD2
0001AC 47F0 C1CC           001CC   762         B     *+32
0001B0 00288000001E                763         DC    H'40',X'8000',H'30'
0001B6 E740404040404040            764         DC    CL8'X       ',XL8'14E73D270DA01CB4',H'2'
0001C8 02AE00EA                    765         DC    H'686,234'
0001CC D217 3004 C1B0 00004 001B0  766         MVC   SQLPLLEN(24),*-28
0001D2 D203 3028 C1C8 00028 001C8  767         MVC   SQLSTNUM(4),*-10
0001D8 41F0 D594           00594   768         LA    15,SQLCA
0001DC 50F0 301C           0001C   769         ST    15,SQLCODEP
0001E0 41F0 D048           00048   770         LA    15,NEW_KEYFLD1
0001E4 50F0 3034           00034   771         ST    15,SQLPVARS+8
0001E8 D201 3030 C700 00030 00700  772         MVC   SQLPVARS+4(2),=X'01C4'
```

Figure 74 (Part 12 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
0001EE D201 3032 C702 00032 00702   773          MVC   SQLPVARS+6(2),=H'2'
0001F4 1FFF                         774          SLR   15,15
0001F6 50F0 3038            00038   775          ST    15,SQLPVARS+12
0001FA 41F0 D04A            0004A   776          LA    15,NEW_KEYFLD2
0001FE 50F0 3040            00040   777          ST    15,SQLPVARS+20
000202 D201 303C C700 0003C 00700   778          MVC   SQLPVARS+16(2),=X'01C4'
000208 D201 303E C704 0003E 00704   779          MVC   SQLPVARS+18(2),=H'6'
00020E 1FFF                         780          SLR   15,15
000210 50F0 3044            00044   781          ST    15,SQLPVARS+24
000214 41F0 D050            00050   782          LA    15,NEW_FAMILY
000218 50F0 304C            0004C   783          ST    15,SQLPVARS+32
00021C D201 3048 C706 00048 00706   784          MVC   SQLPVARS+28(2),=X'01C1'
000222 D201 304A C708 0004A 00708   785          MVC   SQLPVARS+30(2),=H'30'
000228 41F0 D0AC            000AC   786          LA    15,IND_FAMILY
00022C 50F0 3050            00050   787          ST    15,SQLPVARS+36
000230 41F0 D070            00070   788          LA    15,NEW_FIRST
000234 50F0 3058            00058   789          ST    15,SQLPVARS+44
000238 D201 3054 C706 00054 00706   790          MVC   SQLPVARS+40(2),=X'01C1'
00023E D201 3056 C70A 00056 0070A   791          MVC   SQLPVARS+42(2),=H'20'
000244 41F0 D0AE            000AE   792          LA    15,IND_FIRST
000248 50F0 305C            0005C   793          ST    15,SQLPVARS+48
00024C 41F0 D086            00086   794          LA    15,NEW_CITY
000250 50F0 3064            00064   795          ST    15,SQLPVARS+56
000254 D201 3060 C706 00060 00706   796          MVC   SQLPVARS+52(2),=X'01C1'
00025A D201 3062 C70C 00062 0070C   797          MVC   SQLPVARS+54(2),=H'35'
000260 41F0 D0B0            000B0   798          LA    15,IND_CITY
000264 50F0 3068            00068   799          ST    15,SQLPVARS+60
000268 41F0 D0B2            000B2   800          LA    15,OLD_KEYFLD1
00026C 50F0 3070            00070   801          ST    15,SQLPVARS+68
000270 D201 306C C700 0006C 00700   802          MVC   SQLPVARS+64(2),=X'01C4'
000276 D201 306E C702 0006E 00702   803          MVC   SQLPVARS+66(2),=H'2'
00027C 1FFF                         804          SLR   15,15
00027E 50F0 3074            00074   805          ST    15,SQLPVARS+72
000282 41F0 D0B4            000B4   806          LA    15,OLD_KEYFLD2
000286 50F0 307C            0007C   807          ST    15,SQLPVARS+80
00028A D201 3078 C700 00078 00700   808          MVC   SQLPVARS+76(2),=X'01C4'
000290 D201 307A C704 0007A 00704   809          MVC   SQLPVARS+78(2),=H'6'
000296 1FFF                         810          SLR   15,15
000298 50F0 3080            00080   811          ST    15,SQLPVARS+84
00029C D203 302C C6D8 0002C 006D8   812          MVC   SQLPVARS(4),=F'88'
0002A2 41F0 302C            0002C   813          LA    15,SQLPVARS
0002A6 50F0 3020            00020   814          ST    15,SQLVPARM
0002AA D203 3024 C6D0 00024 006D0   815          MVC   SQLAPARM,=XL4'00000000'
0002B0 4110 3004            00004   816          LA    1,SQLPLLEN
0002B4 5010 3000            00000   817          ST    1,SQLPLIST
0002B8 9680 3000      00000         818          OI    SQLPLIST,X'80'
0002BC 4110 3000            00000   819          LA    1,SQLPLIST
0002C0 58F0 C6D4            006D4   820          L     15,=V(DSNHLI)
0002C4 05EF                         821          BALR  14,15
                                    822 ***$$$
0002C6 47F0 C360            00360   823          B     CHECKSQL

                                    825 *-------------------------------------------------------------------*
                                    826 *        UPDATE MIRROR TABLE FOR AN DELETE OPERATION                *
                                    827 *-------------------------------------------------------------------*

0002CA                              829 EXECDEL  DS    0H
                                    830 ***$$$
```

Figure 74 (Part 13 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                 831 *        EXEC  SQL                                                  *
                                                    DELETE                                               *
                                                    FROM                                                 *
                                                      TABLE0M                                            *
                                                    WHERE                                                *
                                                      KEYFLD1  = :NEW_KEYFLD1  AND                       *
                                                      KEYFLD2  = :NEW_KEYFLD2
0002CA 47F0 C2EA            002EA   832          B     *+32
0002CE 00288000001E                 833          DC    H'40',X'8000',H'30'
0002D4 E740404040404040             834          DC    CL8'X       ',XL8'14E73D270DA01CB4',H'3'
0002E6 02C100E9                     835          DC    H'705,233'
0002EA D217 3004 C2CE 00004 002CE   836          MVC   SQLPLLEN(24),*-28
0002F0 D203 3028 C2E6 00028 002E6   837          MVC   SQLSTNUM(4),*-10
0002F6 41F0 D594            00594   838          LA    15,SQLCA
0002FA 50F0 301C            0001C   839          ST    15,SQLCODEP
0002FE 41F0 D048            00048   840          LA    15,NEW_KEYFLD1
000302 50F0 3034            00034   841          ST    15,SQLPVARS+8
000306 D201 3030 C700 00030 00700   842          MVC   SQLPVARS+4(2),=X'01C4'
00030C D201 3032 C702 00032 00702   843          MVC   SQLPVARS+6(2),=H'2'
000312 1FFF                         844          SLR   15,15
000314 50F0 3038            00038   845          ST    15,SQLPVARS+12
000318 41F0 D04A            0004A   846          LA    15,NEW_KEYFLD2
00031C 50F0 3040            00040   847          ST    15,SQLPVARS+20
000320 D201 303C C700 0003C 00700   848          MVC   SQLPVARS+16(2),=X'01C4'
000326 D201 303E C704 0003E 00704   849          MVC   SQLPVARS+18(2),=H'6'
00032C 1FFF                         850          SLR   15,15
00032E 50F0 3044            00044   851          ST    15,SQLPVARS+24
000332 D203 302C C6DC 0002C 006DC   852          MVC   SQLPVARS(4),=F'28'
000338 41F0 302C            0002C   853          LA    15,SQLPVARS
00033C 50F0 3020            00020   854          ST    15,SQLVPARM
000340 D203 3024 C6D0 00024 006D0   855          MVC   SQLAPARM,=XL4'00000000'
000346 4110 3004            00004   856          LA    1,SQLPLLEN
00034A 5010 3000            00000   857          ST    1,SQLPLIST
00034E 9680 3000      00000         858          OI    SQLPLIST,X'80'
000352 4110 3000            00000   859          LA    1,SQLPLIST
000356 58F0 C6D4            006D4   860          L     15,=V(DSNHLI)
00035A 05EF                         861          BALR  14,15
                                    862 ***$$$
00035C 47F0 C360            00360   863          B     CHECKSQL
                                    865 ***********************************************************************
                                    866 *                                                                     *
                                    867 * (3)    CHECK RESULT OF MIRROR TABLE UPDATE                          *
                                    868 *                                                                     *
                                    869 *              - CHECK THE RESULTING SQL CODE                         *
                                    870 *                                                                     *
                                    871 *              - IF UPDATE OF MIRROR TABLE WAS SUCCESSFUL             *
                                    872 *                - CONTINUE WITH RETURN TO CALLING PROGRAM            *
                                    873 *                                                                     *
                                    874 *              - IF MIRROR TABLE UPDATE FAILED                        *
                                    875 *                - EXECUTE THE SQL ERROR LOGIC                        *
                                    876 *                                                                     *
                                    877 ***********************************************************************
                                    879 *---------------------------------------------------------------------*
                                    880 *        CHECK RESULTING SQL CODE                                     *
                                    881 *---------------------------------------------------------------------*

000360                              883 CHECKSQL DS    0H
000360 58F0 D5A0            005A0   884          L     R15,SQLCODE             GET SQLCODE IN REGISTER
000364 12FF                         885          LTR   R15,R15                 WAS MONITOR TABLE UPDATE OK?
000366 4780 C3DA            003DA   886          BZ    RETURN                  YES -> RETURN TO DPROP
                                    887          DROP  R3                      RELINQUISH SQLDSECT ADDRESS.
```

Figure 74 (Part 14 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                       889 ***********************************************************************
                                       890 *                                                                     *
                                       891 * (4)    AN SQL ERROR OCCURED                                         *
                                       892 *                                                                     *
                                       893 *              - PREPARE PARAMETER LIST FOR DSNTIAR                   *
                                       894 *              - CALL DSNTIAR GO GET FORMATTED SQL ERROR MESSAGE      *
                                       895 *              - WTO ANY NON-BLANK MESSAGE LINE RETURNED BY DSNTIAR   *
                                       896 *              - CONTINUE WITH RETURN PROCESSING                      *
                                       897 *                                                                     *
                                       898 ***********************************************************************

                                       900 *---------------------------------------------------------------------*
                                       901 *        PREPARE PARAMETER LIST FOR DSNTIAR                           *
                                       902 *---------------------------------------------------------------------*

00036A                                 904 SQLERR   DS    0H
00036A 4110 D594            00594      905          LA    R1,SQLCA                GET POINTER TO SQLCA
00036E 5010 D138            00138      906          ST    R1,WRKDSNT1             AND STORE AS PARAMETER 1
000372 4110 D144            00144      907          LA    R1,WRKMSG               GET POINTER TO MESSAGE AREA
000376 5010 D13C            0013C      908          ST    R1,WRKDSNT2             AND STORE AS PARAMETER 2
00037A 4110 C6E0            006E0      909          LA    R1,=A(L'WRKMSG1)        GET POINTER TO LINE LENGTH
00037E 5010 D140            00140      910          ST    R1,WRKDSNT3             AND STORE AS PARAMETER 3
000382 9680 D140      00140            911          OI    WRKDSNT3,X'80'          INDICATE LAST IN LIST
000386 D201 D144 C70E 00144 0070E      912          MVC   WRKMSGL,=AL2(10*L'WRKMSG1)   SETUP LENGTH OF MSG AREA

                                       914 *---------------------------------------------------------------------*
                                       915 *        CALL DSNTIAR TO GET FORMATTED SQL ERROR MESSAGE              *
                                       916 *---------------------------------------------------------------------*

00038C 4110 D138            00138      918          LA    R1,WRKDSNT              GET POINTER TO PARMLIST
                                       919          LINK  EP=DSNTIAR              THEN LINK TO DSNTIAR MODULE
0003A6 49F0 C710            00710      926          CH    R15,=H'4'               WAS MESSAGE FORMATTED?
0003AA 4720 C3DA            003DA      927          BH    RETURN                  NO  -> THEN IGNORE ERROR

                                       929 *---------------------------------------------------------------------*
                                       930 *        WTOA ANY NON-BLANK MESSAGE LINE RETURNED BY DSNTIAR          *
                                       931 *---------------------------------------------------------------------*

0003AE 4130 D146            00146      933          LA    R3,WRKMSG1              POINT FIRST MESSAGE LINE
0003B2 4120 000A            0000A      934          LA    R2,10                   INITIALYZE THE LOOP REGISTER
0003B6                                 935 DSNTLOOP DS    0H
0003B6 D56D 3000 C712 00000 00712      936          CLC   0(L'WRKMSG1,R3),=CL(L'WRKMSG1)' '   IS THE LINE BLANK?
0003BC 4780 C3DA            003DA      937          BE    RETURN                  YES -> THEN LEAVE THE LOOP
0003C0 D27D D0BA C5A0 000BA 005A0      938          MVC   WRKWTO,WTODSNTM         ELSE MOVE WTO SKEL TO WORKAREA
0003C6 D26D D0C7 3000 000C7 00000      939          MVC   WRKWTOTM,0(R3)          SETUP TEXT RETURNED BY DSNTIAR
                                       940          WTO   MF=(E,WRKWTO)           AND ISSUE THE WTO
0003D2 4130 306E            0006E      943          LA    R3,L'WRKMSG1(0,R3)      POINT NEXT MESSAGE LINE
0003D6 4620 C3B6            003B6      944          BCT   R2,DSNTLOOP             AND REPEAT THE WTO LOOP
                                       946 ***********************************************************************
                                       947 *                                                                     *
                                       948 * (5)    RETURN PROCESSING                                            *
                                       949 *                                                                     *
                                       950 *              - RELOAD REGISTER AND RETURN TO DPROP                  *
                                       951 *                                                                     *
                                       952 ***********************************************************************

                                       954 *---------------------------------------------------------------------*
                                       955 *        RELOAD REGISTERS AND RETURN TO DPROP                         *
                                       956 *---------------------------------------------------------------------*

0003DA                                 958 RETURN   DS    0H
0003DA 58D0 D004            00004      959          L     R13,4(0,R13)            POINT CALLERS SAVEAREA
                                       960          RETURN (14,12),RC=0           AND RETURN TO DPROP
```

Figure 74 (Part 15 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                   ***********************************************************************
                                   *                                                                     *
                                   *        SETUP HOST VARIABLE ROUTINE                                  *
                                   *                                                                     *
                                   *        - ENTRY:  R2      - POINTER TO CDCDA OF BEFORE OR AFTER IMG. *
                                   *                  R3      - POINTER TO HOST VARIABLE SETUP TABLE     *
                                   *                  R8      - POINTER TO CDCDD                         *
                                   *                  R11     - RETURN ADDRESS                           *
                                   *                  R13     - POINTER TO HOST VARIABLE WORK AREA       *
                                   *                                                                     *
                                   *        - RETURN: HOST VARIABLES ARE COPIED IN THE WORK AREA         *
                                   *                  ACCORDING TO THE DESCRIPTIONS IN THE SETUP TABLE.  *
                                   *                                                                     *
                                   ***********************************************************************

                                   *---------------------------------------------------------------------*
                                   *        POINT TO CORRECT CDCDA START ADDRESS AND DECLARE ADDRESS.    *
                                   *---------------------------------------------------------------------*

                             00000          USING QW0185,R8               DECLARE CDCDD ADDRESSABILITY
                             00000          USING HOSTTAB,R3              DECLARE HOSTTAB ENTRY ADDRESS.
0003E8                             SETHOST  DS    0H
0003E8 4120 2054             00054          LA    R2,QW0185DA-QW0185(0,R2) POINT TO CORRECT DATA START

                                   *---------------------------------------------------------------------*
                                   *        PROCESS NEXT ENTRY IN PASSED HOST VARIABLE TABLE             *
                                   *---------------------------------------------------------------------*

0003EC                             SETH010  DS    0H
0003EC D501 C780 3000 00780 00000           CLC   =X'FFFF',0(R3)          END OF TABLE REACHED?
0003F2 078B                                 BER   R11                     YES -> RETURN TO CALLER
0003F4 48E0 8062             00062          LH    R14,QW0185LD            GET NUMBER OF QW0185VR
0003F8 41F0 8064             00064          LA    R15,QW0185VR            POINT FIRST QW0185VR OCCURENCE
                             00064          USING QW0185VR,R15            DECLARE QW0185VR ADDRESSABILITY

                                   *---------------------------------------------------------------------*
                                   *        FIND CORRESPONDING COLUMN IN CDCDD                           *
                                   *---------------------------------------------------------------------*

0003FC                             SETH020  DS    0H
0003FC D501 F00C 3000 00070 00000           CLC   QW0185NL,HOSTTBNL       SAME COLUMN NAME LENGTH?
000402 4770 C414             00414          BNE   SETH030                 NO  -> PROCESS NEXT QW0185VR
000406 4810 3000             00000          LH    R1,HOSTTBNL             YES -> GET LENGTH OF COLNAME
00040A 0610                                 BCTR  R1,0                    ADJUST IT FOR EXECUTE
00040C 4410 C48E             0048E          EX    R1,CLCTBNL              CHECK IF COLUMN NAME MATCH
000410 4780 C420             00420          BE    SETH040                 YES -> LOOK IF SAME DATA TYPE

                                   *---------------------------------------------------------------------*
                                   *        SETUP TO PROCESS NEXT QW0185VR IN CDCDD                      *
                                   *---------------------------------------------------------------------*

000414                             SETH030  DS    0H
000414 41F0 F02C             0002C          LA    R15,L'QW0185VR(0,R15)   POINT NEXT QW0185VR ENTRY
000418 46E0 C3FC             003FC          BCT   R14,SETH020             AND REPEAT THE SEARCH LOOP
00041C 47F0 C464             00464          B     SETHE10                 COLUMN NOT FOUND -> ERROR

                                   *---------------------------------------------------------------------*
                                   *        COLUMN FOUND - LOOK IF SAME DATA TYPE                        *
                                   *---------------------------------------------------------------------*

000420                             SETH040  DS    0H
000420 D501 F000 3014 00064 00014           CLC   QW0185ST,HOSTTBST       SAME COLUMN DATA TYPE?
000426 4770 C46E             0046E          BNE   SETHE20                 NO -> MISMATCHING DATA TYPES

                                   *---------------------------------------------------------------------*
                                   *        SETUP HOST VARIABLE VALUES FROM CDCDA                        *
```

Figure 74 (Part 16 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                                1031 *--------------------------------------------------------------------*

00042A                          1033 SETH050  DS    0H
00042A 5810 F008          0006C  1034          L     R1,QW0185SI            GET COLUMN OFFSET IN CDCDA
00042E 5410 C6E4          006E4  1035          N     R1,=X'00FFFFFF'        RESET UNUSED HIGH ORDER BYTE
000432 1E12                     1036          ALR   R1,R2                  POINT TO COLUMN DATA IN CDCDA

                                1038 *------- FILL EVENTUALLY NULL INDICATOR VARIABLE
000434 9101 3015      00015     1039          TM    HOSTTBST+1,NULL        IS THE COLUMN NULLABLE?
000438 4780 C44C          0044C  1040          BZ    SETH060                NO -> CONTINUE BELOW
00043C 48E0 3016          00016  1041          LH    R14,HOSTTBIO           GET OFFSET OF NULL INDICATOR
000440 1EED                     1042          ALR   R14,R13                AND POINT TO NULL INDICATOR
000442 D201 E000 1000 00000 00000 1043          MVC   0(2,R14),0(R1)         COPY NULL INDICATOR FROM CDCDA
000448 4110 1002          00002  1044          LA    R1,2(0,R1)             THEN POINT PAST NULL INDICATOR

                                1046 *------- COPY HOST VARIABLE FROM PASSED CDCDA AREA
00044C                          1047 SETH060  DS    0H
00044C 1801                     1048          LR    R0,R1                  POINT TO SOURCE (IN CDCDA)
00044E 48E0 3018          00018  1049          LH    R14,HOSTTBDO           GET OFFSET OF TARGET (IN WRK)
000452 1EED                     1050          ALR   R14,R13                THEN POINT TO TARGET (IN WRK)
000454 48F0 301A          0001A  1051          LH    R15,HOSTTBDL           GET LENGTH OF TARGET
000458 181F                     1052          LR    R1,R15                 SET ALSO AS SOURCE LENGTH
00045A 0EE0                     1053          MVCL  R14,R0                 THEN COPY THE HOST VARIABLE

                                1055 *--------------------------------------------------------------------*
                                1056 *        POINT PAST HOSTTAB ENTRY AND BRANCH TO PROCESS NEXT         *
                                1057 *--------------------------------------------------------------------*

00045C                          1059 SETH070  DS    0H
00045C 4130 301C          0001C  1060          LA    R3,HOSTTABN            POINT NEXT ENTRY IN HOSTTAB
000460 47F0 C3EC          003EC  1061          B     SETH010                AND BRANCH TO PROCESS IT

                                1063 *--------------------------------------------------------------------*
                                1064 *        A FIELD WAS NOT FOUND IN THE CDCDD DATA STREAM              *
                                1065 *--------------------------------------------------------------------*

000464                          1067 SETHE10  DS    0H
                                1068          WTO   MF=(E,WTOERRMF)        ISSUE MISSING COLUMN ERROR WTO
00046A 47F0 C478          00478  1071          B     SETHE90                THEN CONTINUE BELOW

                                1073 *--------------------------------------------------------------------*
                                1074 *        A FIELD DOES NOT MATCH THE EXPECTED DATA TYPE               *
                                1075 *--------------------------------------------------------------------*

00046E                          1077 SETHE20  DS    0H
                                1078          WTO   MF=(E,WTOERRDT)        ISSUE DATA TYPE ERROR WTO
000474 47F0 C478          00478  1081          B     SETHE90                THEN CONTINUE BELOW

                                1083 *--------------------------------------------------------------------*
                                1084 *        REPORT THE COLUMN IN ERROR                                  *
                                1085 *--------------------------------------------------------------------*

000478                          1087 SETHE90  DS    0H
000478 D232 D0BA C694 000BA 00694 1088          MVC   WRKWTO(WTOCOLEL),WTOCOLEM  MOVE WTO SKEL TO WORKAREA
00047E D211 D0D8 3002 000D8 00002 1089          MVC   WRKWTO+30(L'HOSTTBCN),HOSTTBCN  SETUP COLUMN NAME
                                1090          WTO   MF=(E,WRKWTO)          AND ISSUE THE WTO
00048A 47F0 C3DA          003DA  1093          B     RETURN                 THEN RETURN TO CALLER

                                1095 *------- EXECUTED SUBJECT INSTRUCTIONS WITHIN SETHOST ROUTINE
00048E D500 F00E 3002 00072 00002 1096 CLCTBNL  CLC   QW0185CN(*-*),HOSTTBCN   EXECUTED SUBJECT INSTRUCTION
                                1097          DROP  R15                    RELINQUISH QW0185VR ADDRESS.
                                1098          DROP  R8                     RELINQUISH CDCDD    ADDRESS.
                                1099          DROP  R3                     RELINQUISH HOSTTAB  ADDRESS.
```

Figure 74 (Part 17 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                     ***********************************************************************
                     *                                                                     *
                     *       DEFINITIONS                                                   *
                     *                                                                     *
                     *             - READ-ONLY CONSTANTS                                   *
                     *             - LITERAL POOL                                          *
                     *             - EQUATES                                               *
                     *                                                                     *
                     ***********************************************************************

                     *---------------------------------------------------------------------*
                     *       HOST VARIABLE MAPPING TABLE FOR BEFORE IMAGE COLUMNS          *
                     *---------------------------------------------------------------------*

000494               COLBTAB  DS    0F
                     *------- ENTRY FOR OLD_KEYFLD1 HOST VARIABLE
000494 0007                   DC    AL2(7)                   LENGTH OF COLUMN NAME
000496 D2C5E8C6D3C4F140       DC    CL18'KEYFLD1'            COLUMN NAME
0004A8 01C4                   DC    AL2(CHAR)                COLUMN DATA TYPE
0004AA 0000                   DC    AL2(0)                   OFFSET OF NULL INDICATOR
0004AC 00B2                   DC    AL2(OLD_KEYFLD1-WRK)     OFFSET OF HOST VARIABLE
0004AE 0002                   DC    AL2(LEN_KEYFLD1)         LENGTH OF HOST VARIABLE
                     *------- ENTRY FOR OLD_KEYFLD2 HOST VARIABLE
0004B0 0007                   DC    AL2(7)                   LENGTH OF COLUMN NAME
0004B2 D2C5E8C6D3C4F240       DC    CL18'KEYFLD2'            COLUMN NAME
0004C4 01C4                   DC    AL2(CHAR)                COLUMN DATA TYPE
0004C6 0000                   DC    AL2(0)                   OFFSET OF NULL INDICATOR
0004C8 00B4                   DC    AL2(OLD_KEYFLD2-WRK)     OFFSET OF HOST VARIABLE
0004CA 0006                   DC    AL2(LEN_KEYFLD2)         LENGTH OF HOST VARIABLE
                     *------- END OF TABLE MARKER
0004CC FFFF                   DC    X'FFFF'                  END OF TABLE MARKER

                     *---------------------------------------------------------------------*
                     *       HOST VARIABLE MAPPING TABLE FOR AFTER IMAGE COLUMNS           *
                     *---------------------------------------------------------------------*

0004D0               COLATAB  DS    0F
                     *------- ENTRY FOR NEW_KEYFLD1 HOST VARIABLE
0004D0 0007                   DC    AL2(7)                   LENGTH OF COLUMN NAME
0004D2 D2C5E8C6D3C4F140       DC    CL18'KEYFLD1'            COLUMN NAME
0004E4 01C4                   DC    AL2(CHAR)                COLUMN DATA TYPE
0004E6 0000                   DC    AL2(0)                   OFFSET OF NULL INDICATOR
0004E8 0048                   DC    AL2(NEW_KEYFLD1-WRK)     OFFSET OF HOST VARIABLE
0004EA 0002                   DC    AL2(LEN_KEYFLD1)         LENGTH OF HOST VARIABLE
                     *------- ENTRY FOR NEW_KEYFLD2 HOST VARIABLE
0004EC 0007                   DC    AL2(7)                   LENGTH OF COLUMN NAME
0004EE D2C5E8C6D3C4F240       DC    CL18'KEYFLD2'            COLUMN NAME
000500 01C4                   DC    AL2(CHAR)                COLUMN DATA TYPE
000502 0000                   DC    AL2(0)                   OFFSET OF NULL INDICATOR
000504 004A                   DC    AL2(NEW_KEYFLD2-WRK)     OFFSET OF HOST VARIABLE
000506 0006                   DC    AL2(LEN_KEYFLD2)         LENGTH OF HOST VARIABLE
                     *------- ENTRY FOR NEW_FAMILY HOST VARIABLE
000508 0006                   DC    AL2(6)                   LENGTH OF COLUMN NAME
00050A C6C1D4C9D3E84040       DC    CL18'FAMILY'             COLUMN NAME
00051C 01C1                   DC    AL2(VARCHAR+NULL)        COLUMN DATA TYPE
00051E 00AC                   DC    AL2(IND_FAMILY-WRK)      OFFSET OF NULL INDICATOR
000520 0050                   DC    AL2(NEW_FAMILY-WRK)      OFFSET OF HOST VARIABLE
000522 0020                   DC    AL2(LEN_FAMILY)          LENGTH OF HOST VARIABLE
                     *------- ENTRY FOR NEW_FIRST HOST VARIABLE
000524 0005                   DC    AL2(5)                   LENGTH OF COLUMN NAME
000526 C6C9D9E2E3404040       DC    CL18'FIRST'              COLUMN NAME
000538 01C1                   DC    AL2(VARCHAR+NULL)        COLUMN DATA TYPE
00053A 00AE                   DC    AL2(IND_FIRST-WRK)       OFFSET OF NULL INDICATOR
00053C 0070                   DC    AL2(NEW_FIRST-WRK)       OFFSET OF HOST VARIABLE
00053E 0016                   DC    AL2(LEN_FIRST)           LENGTH OF HOST VARIABLE
```

Figure 74 (Part 18 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                              1166 *------- ENTRY FOR NEW_CITY HOST VARIABLE
000540 0004                   1167         DC    AL2(4)                  LENGTH OF COLUMN NAME
000542 C3C9E3E840404040       1168         DC    CL18'CITY'              COLUMN NAME
000554 01C1                   1169         DC    AL2(VARCHAR+NULL)       COLUMN DATA TYPE
000556 00B0                   1170         DC    AL2(IND_CITY-WRK)       OFFSET OF NULL INDICATOR
000558 0086                   1171         DC    AL2(NEW_CITY-WRK)       OFFSET OF HOST VARIABLE
00055A 0025                   1172         DC    AL2(LEN_CITY)           LENGTH OF HOST VARIABLE
                              1173 *------- END OF TABLE MARKER
00055C FFFF                   1174         DC    X'FFFF'                 END OF TABLE MARKER

                              1176 *-----------------------------------------------------------------*
                              1177 *       WTO MACRO LIST FORMATS                                    *
                              1178 *-----------------------------------------------------------------*

                              1180 WTOERROP WTO   'EKYEDB1E  INVALID OPERATION CODE IN CDC DATA DEFINITION*
                                                  ',                                                 *
                                                  ROUTCDE=11,                                        *
                                                  MF=L
                              1187 WTODSNTM WTO   'EKYEDB2E ----+----1----+----2----+----3----+----4----+-*
                                                  ---5----+----6----+----7----+----8----+----9----+----0--*
                                                  -+----1',                                          *
                                                  ROUTCDE=11,                                        *
                                                  MF=L
                        0007E 1194 WTODSNTL EQU   *-WTODSNTM              LENGTH OF WTO PARMLIST
                              1195 WTOERRDT WTO   'EKYEDB3E  UNEXPECTED COLUMN DATA TYPE ENCOUNTERED',    *
                                                  ROUTCDE=11,                                        *
                                                  MF=L
                              1202 WTOERRMF WTO   'EKYEDB4E  EXPECTED COLUMN NOT IN PASSED CDCDD',        *
                                                  ROUTCDE=11,                                        *
                                                  MF=L
                              1209 WTOCOLEM WTO   'EKYEDB5I COLUMN IN ERROR: ----+----1----+--',          *
                                                  ROUTCDE=11,                                        *
                                                  MF=L
                        00033 1216 WTOCOLEL EQU   *-WTOCOLEM              LENGTH OF WTO PARMLIST

                              1218 *-----------------------------------------------------------------*
                              1219 *       LITERAL POOL                                              *
                              1220 *-----------------------------------------------------------------*

0006C8                        1222         LTORG ,                       EXPAND LITERAL POOL
0006C8 000006BC               1223               =A(WRKLEN)
0006CC 00000040               1224               =F'64'
0006D0 00000000               1225               =XL4'00000000'
0006D4 00000000               1226               =V(DSNHLI)
0006D8 00000058               1227               =F'88'
0006DC 0000001C               1228               =F'28'
0006E0 0000006E               1229               =A(L'WRKMSG1)
0006E4 00FFFFFF               1230               =X'00FFFFFF'
0006E8 E3C1C2D3C5F0F240       1231               =CL18'TABLE02'
0006FA C9D5                   1232               =C'IN'
0006FC E4C1                   1233               =C'UA'
0006FE C4C5                   1234               =C'DE'
000700 01C4                   1235               =X'01C4'
000702 0002                   1236               =H'2'
000704 0006                   1237               =H'6'
000706 01C1                   1238               =X'01C1'
000708 001E                   1239               =H'30'
00070A 0014                   1240               =H'20'
00070C 0023                   1241               =H'35'
00070E 044C                   1242               =AL2(10*L'WRKMSG1)
000710 0004                   1243               =H'4'
000712 4040404040404040       1244               =CL(L'WRKMSG1)' '
000780 FFFF                   1245               =X'FFFF'
```

Figure 74 (Part 19 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                               *-------------------------------------------------------------------*
                               *       EQUATES                                                     *
                               *-------------------------------------------------------------------*

                        00000  R0       EQU   0                       WORK / LINKAGE
                        00001  R1       EQU   1                       WORK / LINKAGE
                        00002  R2       EQU   2                       WORK
                        00003  R3       EQU   3                       WORK / SQLWA (SQLDSECT)
                        00004  R4       EQU   4                       -
                        00005  R5       EQU   5                       -
                        00006  R6       EQU   6                       -
                        00007  R7       EQU   7                       -
                        00008  R8       EQU   8                       CDCDA & CDCDD (QW0185)
                        00009  R9       EQU   9                       ANCHOR AREA   (ANCHOR)
                        0000A  R10      EQU   10                      HUP EXTERNAL CB (HEC)
                        0000B  R11      EQU   11                      -
                        0000C  R12      EQU   12                      MODULE BASE REGISTER
                        0000D  R13      EQU   13                      SAVE AND WORKAREA (WRK)
                        0000E  R14      EQU   14                      WORK / LINKAGE
                        0000F  R15      EQU   15                      WORK / LINKAGE
                               *********************************************************************
                               *                                                                   *
                               *       DUMMY SECTIONS                                              *
                               *                                                                   *
                               *             - HOST VARIABLE SETUP TABLE (HOSTTAB)                 *
                               *             - HUP EXTERNAL INTERFACE (HEC)                        *
                               *             - IFI STANDARD HEADER AREA (QWHS)                     *
                               *             - IFI CORRELATION DATA AREA (QWHC)                    *
                               *             - IFI IFCIDS 140 UP MAPPING (QW02)                    *
                               *                                                                   *
                               *********************************************************************

                               *-------------------------------------------------------------------*
                               *       HOST VARIABLE SETUP TABLE - HOSTTAB                         *
                               *-------------------------------------------------------------------*

000000                         HOSTTAB  DSECT ,
000000                         HOSTTBNL DS    H                       LENGTH OF COLUMN NAME
000002                         HOSTTBCN DS    CL18                    NAME OF COLUMN
000014                         HOSTTBST DS    H                       DATA TYPE OF COLUMN
000016                         HOSTTBIO DS    H                       OFFSET (IN WRK) OF NULL IND.
000018                         HOSTTBDO DS    H                       OFFSET (IN WRK) OF DATA FIELD
00001A                         HOSTTBDL DS    H                       LENGTH (IN WRK) OF DATA FIELD
                        0001C  HOSTTABN EQU   *                       NEXT ENTRY OF HOSTTAB

                               *------- EQUATES FOR COLUMN DATA TYPES (HOSTTBST)
                        00180  DATE     EQU   384                     - DATE TYPE COLUMN
                        00184  TIME     EQU   388                     - TIME TYPE COLUMN
                        00188  TIMESTMP EQU   392                     - TIMESTAMP TYPE COLUMN
                        001C0  VARCHAR  EQU   448                     - VARCHAR COLUMN
                        001C4  CHAR     EQU   452                     - CHAR COLUMN
                        001C8  LONGVAR  EQU   456                     - LONGVARCHAR COLUMN
                        001D0  VARG     EQU   464                     - VARGRAPHIC COLUMN
                        001D4  GRAPHIC  EQU   468                     - GRAPHIC COLUMN
                        001D8  LONGVARG EQU   472                     - LONG VARGRAPHIC COLUMN
                        001E0  FLOAT    EQU   480                     - FLOAT TYPE COLUMN
                        001E4  DECIMAL  EQU   484                     - DECIMAL TYPE COLUMN
                        001F0  INTEGER  EQU   496                     - INTEGER TYPE COLUMN
                        001F4  SMALLINT EQU   500                     - SMALL INTEGER TYPE COLUMN
                        00001  NULL     EQU   1                       - ADDITIONAL NULL INDICATOR

                               *-------------------------------------------------------------------*
                               *       HUP EXTERNAL INTERFACE - HEC                                *
                               *-------------------------------------------------------------------*

                                        EKYHCHEC
```

Figure 74 (Part 20 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

```
                          1413 *-------------------------------------------------------------------*
                          1414 *       INSTRUMENTATION FACILITY STANDARD HEADER AREA - QWHS        *
                          1415 *-------------------------------------------------------------------*

                          1417          DSNDQWHS
                          1475+         PRINT NOGEN

                          1880 *-------------------------------------------------------------------*
                          1881 *       INSTRUMENTATION FACILITY CORRELATION DATA AREA - QWHC       *
                          1882 *-------------------------------------------------------------------*

                          1884          DSNDQWHC

                          1927 *-------------------------------------------------------------------*
                          1928 *       INSTRUMENTATION FACILITY IFCIDS 140 UP MAPPING - QW02       *
                          1929 *-------------------------------------------------------------------*

                          1931          DSNDQW02
                          4891 ***$$$ SQL WORKING STORAGE
000010 000000A0           4892 SQLDSIZ  DC    A(SQLDLEN) SQLDSECT SIZE
000000                    4893 SQLDSECT DSECT
000000                    4894 SQLPLIST DS    F
000004                    4895 SQLPLLEN DS    H         PLIST LENGTH
000006                    4896 SQLFLAGS DS    XL2       FLAGS
000008                    4897 SQLCTYPE DS    H         CALL-TYPE
00000A                    4898 SQLPROGN DS    CL8       PROGRAM NAME
000012                    4899 SQLTIMES DS    CL8       TIMESTAMP
00001A                    4900 SQLSECTN DS    H         SECTION
00001C                    4901 SQLCODEP DS    A         CODE POINTER
000020                    4902 SQLVPARM DS    A         VPARAM POINTER
000024                    4903 SQLAPARM DS    A         AUX PARAM PTR
000028                    4904 SQLSTNUM DS    H         STATEMENT NUMBER
00002A                    4905 SQLSTYPE DS    H         STATEMENT TYPE
00002C                    4906 SQLPVARS DS    F,7CL12
000084                    4907 SQLAVARS DS    F,0CL12
000088                    4908 SQLTEMP  DS    CL18      TEMPLATE
0000A0                    4909          DS    0D
                   000A0  4910 SQLDLEN  EQU   *-SQLDSECT
                          4911          END
```

Figure 74 (Part 21 of 21). Sample DB2 Data Capture Subexit Routine (Assembler)

Definitions for Sample DB2 Data Capture Subexit Routine

The following statements illustrate how to specify the use of the DB2 Data Capture subexit routine and illustrate the environment that was set up for the exit routine shown in Figure 74 on page 274.

DPROPGEN Definitions

Figure 75 on page 295 shows a DBDGEN definition for the Segment exit routine in Figure 74 on page 274.

```
EKYGJCL JCL='//T096277G  JOB  (00,000,,500),''DPROP GEN'','
EKYGJCL JCL='//          REGION=0K,NOTIFY=T096277'
EKYGSYS                                                   *
      ROUTCDE=11,                                         *
      SVCNO=227,                                          *
      ILOGREC=E0,                                         *
      SQLDLM=D,                                           *
      SMFREC=245,                                         *
      PRSET=PRSET1,                                       *
      DATE=ISO,                                           *
      TIME=ISO,                                           *
      DBDV=(6,0),                                         *
      EKYRESLB='KOE.DPM120.LOAD'
EKYGDPR                                                   *
      SNAME=T096277,                                      *
      SNR=4,                                              *
      SUBX=EKYEDB2A,                                     *
      STATF='KOE.FF.STATF',                               *
      TQUAL=T096277,                                      *
      DB2SYS=DSN,                                         *
      VLFCLASS=PM1
EKYGEN
END
```

Figure 75. DPROPGEN Definition

Note: The SUBX= keyword of the EKYGSYS Macro specifies the use of a DB2 Data Capture subexit routine for this DPROP system.

CREATE TABLE Statement for Source Table

Figure 76 shows a CREATE TABLE statement for the source table for the Segment exit routine in Figure 74 on page 274.

```
CREATE TABLE  TABLE02
             (KEYFLD1  CHAR(2)       NOT NULL,
              KEYFLD2  CHAR(6)       NOT NULL,
              FAMILY   VARCHAR(30)   ,
              FIRST    VARCHAR(20)   ,
              CITY     VARCHAR(35)   ,
       PRIMARY KEY    (KEYFLD1, KEYFLD2))
       DATA CAPTURE CHANGES
       IN   DU096277.DPROPTS2  ;

CREATE UNIQUE INDEX DPROPIX2
       ON TABLE02     (KEYFLD1, KEYFLD2)
       USING VCAT KOE ;
```

Figure 76. CREATE TABLE Statement for Source Table

Note: The DATA CAPTURE CHANGES clause specifies that the changed DB2 rows are captured and that the DB2CDCEX routine (the HUP) is called when a row of this table is changed.

CREATE TABLE Statement for Mirror Table

Figure 77 on page 296 shows a CREATE TABLE statement for the mirror table for the Segment exit routine in Figure 74 on page 274.

```
CREATE TABLE  TABLE0M
             (KEYFLD1  CHAR(2)       NOT NULL,
              KEYFLD2  CHAR(6)       NOT NULL,
              FAMILY   VARCHAR(30)   ,
              FIRST    VARCHAR(20)   ,
              CITY     VARCHAR(35)   ,
       PRIMARY KEY    (KEYFLD1, KEYFLD2))
       IN   DU096277.DPROPTSM  ;

CREATE UNIQUE INDEX DPROPIXM
       ON TABLE0M     (KEYFLD1, KEYFLD2)
       USING VCAT KOE ;
```

Figure 77. CREATE TABLE Statement for Mirror Table

Note: The mirror table cannot have the DATA CAPTURE CHANGES clause because table updates done within the DB2 Data Capture exit cannot be captured themselves.

Chapter 6. EKYRESLB Dynamic Allocation Exit Routine

DPROP needs to load some DPROP modules from an APF-authorized library allocated to the EKYRESLB DD name. The EKYRESLB DD name is either:

- Allocated through a JCL DD statement that you provide, or
- Dynamically allocated by DPROP to a data set name that your System Administrator provided during DPROP installation.

If neither of these methods suits your needs, then you can provide an EKYRESLB Dynamic Allocation exit routine. For example, the EKYRESLB Dynamic Allocation exit routine can be useful if your installation uses an online change philosophy based on two load module libraries (for example, DPROP.RESLB1 and DPROP.RESLB2), and switches dynamically between these two libraries. Your EKYRESLB Dynamic Allocation exit routine can be used to decide dynamically which one of the two libraries must be dynamically allocated; for example, making the decision based on a specification located in a SYS1.PARMLIB member, or in a linklist load module.

Providing an EKYRESLB Dynamic Allocation exit routine is optional. Its load module name must be **EKYDAEX0** and DPROP loads it from the usual //STEPLIB, //JOBLIB, linklist, LPA concatenation.

Your exit routine can be written in Assembler, but not in COBOL, PL/I, or C.

DPROP calls this exit routine with two different call functions: an AL (ALLOCATE) call function and a DE (DEALLOCATE) call function.

AL During an ALLOCATE call, your exit routine must dynamically allocate the EKYRESLB DD statement (if not already allocated).

DPROP tells your exit routine whether the EKYRESLB DD statement is already allocated or not. If upon return from your exit routine, the EKYRESLB DD statement is not allocated, DPROP dynamically allocates the data set name that your System Administrator identified during DPROP installation.

DE During the DEALLOCATE call, your exit routine can dynamically deallocate the EKYRESLB DD statement, or return without doing any processing.

Your exit routine is called with one ALLOCATE and one DEALLOCATE call function within each OS/VS task executing DPROP functions. This can occur multiple times within the same job step, for example, in MPP regions after pseudo-ABENDs that do not result in a job step ABEND. This can also occur when the RUP is called to perform asynchronous data propagation.

Make sure that each allocation is performed in the same way. To avoid inconsistent allocations, your exit routine must deallocate the EKYRESLB DD Statement during DEALLOCATE calls only if your exit routine performed the original allocation.

Interface Control Block

Code the EKYDAE macro statement to create the following DSECT in your Assembler exit routine.

The interface control block is followed by a detailed description of its fields.

```
                                  1         EKYDAE
                                  2+****************** START OF CONTROL BLOCK SPECIFICATION **************/
                                  3+*                                                                     */
                                  4+*       CONTROL BLOCK NAME:                                           */
                                  5+*         EKYDAE (DAE)                                                */
                                  6+*                                                                     */
                                  7+*       DESCRIPTIVE NAME:                                             */
                                  8+*         INTERFACE CONTROL BLOCK FOR DPROP USER EXIT ROUTINE         */
                                  9+*         PERFORMING DYNAMIC ALLOCATION OF THE EKYRESLB DD STATEMENT*/
                                 10+*                                                                     */
                                 11+*                                                                     */
                                 12+***********************************************************************
                                 13+*                                                                     *
                                 14+*         THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".        *
                                 15+*                                                                     *
                                 16+*         5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                *
                                 17+*         ALL RIGHTS RESERVED.                                        *
                                 18+*                                                                     *
                                 19+*         U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                   *
                                 20+*         USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY               *
                                 21+*         GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                    *
                                 22+*                                                                     *
                                 23+*         LICENSED MATERIALS - PROPERTY OF IBM.                       *
                                 24+*                                                                     *
                                 25+***********************************************************************
                                 26+*                                                                     */
                                 27+*       STATUS: V1 R2 M0                                              */
                                 28+*                                                                     */
                                 29+*       FUNCTION:                                                     */
                                 30+*         THIS IS THE CONTROL BLOCK USED TO INTERFACE BETWEEN         */
                                 31+*              - DPROP                                                */
                                 32+*              AND                                                    */
                                 33+*              - A USER'S EXIT ROUTINE SUPPORTING DYNAMIC             */
                                 34+*                ALLOCATION AND DEALLOCATION OF THE EKYRESLB DD       */
                                 35+*                STATEMENT.                                           */
                                 36+*                                                                     */
                                 37+*                                                                     */
                                 38+*       MODULE TYPE= MACRO                                            */
                                 39+*          PROCESSOR= ASSEMBLER H                                     */
                                 40+*                                                                     */
                                 41+*       INNER CONTROL BLOCKS: NONE                                    */
                                 42+*                                                                     */
                                 43+*       MACROS USED FROM MACRO LIBRARY: NONE                          */
                                 44+*                                                                     */
                                 45+*       CHANGE ACTIVITY: NONE                                         */
                                 46+*                                                                     */
                                 47+******************** END OF CONTROL BLOCK SPECIFICATION **************/

000000                           49+EKYDAE   DSECT
000000 C5D2E8C4C1C54040          50+DAENAME  DC    CL8'EKYDAE  '      EYE CATCHER 'EKYDAE '
000008 4040                      51+DAECALL  DC    CL2' '             TYPE OF CALL TO EXIT:
                                 52+*                                  - 'AL': ALLOCATE EKYRESLB
                                 53+*                                  - 'DE': DEALLOCATE EKYRESLB

00000A 40                        55+DAEALLOC DC    C' '               FLAG INDICATING WHETHER //EKYRESLB
                                 56+*                                 IS ALREADY ALLOCATED.
                       000E8     57+DAEALLOY EQU   C'Y'                - Y: ALREADY ALLOCATED
                       000D5     58+DAEALLON EQU   C'N'                - N: NOT ALREADY ALLOCATED
00000B 40                        59+         DC    CL1' '             RESERVED FOR DPROP
00000C 0000000000000000          60+         DC    5F'0'              RESERVED FOR DPROP

000020                           62+         DS    0D
000020 0000000000000000          63+DAEUSER  DC    256X'00'           CAN BE USED BY USER EXIT ROUTINE
                       00120     64+DAEEND   EQU   *                  END OF DAE DSECT
                       00120     65+DAELEN   EQU   *-EKYDAE           LENGTH OF DAE DSECT
                                 66         END
```

Figure 78. Interface Control Block for EKYRESLB Dynamic Allocation Exit Routine

DAENAME Contains the constant EKYDAE, which is used to identify the control block in a storage dump.

DAECALL The call function that describes whether the exit routine is called to allocate or deallocate the EKYRESLB DD name.

DAEALLOC When called for an ALLOCATE call function, your exit routine must test the content of this field. When called for a DEALLOCATE call function, this field has no meaning. The content can be:

Y The EKYRESLB is already allocated; for example, it was allocated either through a JCL DD statement, or because your exit routine called it, but it was never deallocated.

N The EKYRESLB DD statement is not yet allocated. Your exit routine must allocate the EKYRESLB DD statement dynamically. If your exit routine does not allocate the EKYRESLB DD statement, DPROP dynamically allocates it to the data set name that your System Administrator provided during DPROP installation.

DAEUSER Your exit routine can use this field to exchange information between the ALLOCATE and DEALLOCATE call.

At entry to an ALLOCATE call, DPROP sets this field to binary zeros. DPROP does not change the content of DAEUSER after this first call to your exit routine.

Exit Routine Processing

Your EKYDAEX0 routine must be written in Assembler and must conform to the following linkage conventions:

1. Your exit routine is called and must return in AMODE 31. The call parameter that DPROP provides to your exit routine is usually located above the 16-MB line.
2. On entry, your exit routine must save the registers into the save area that the caller provides, and must provide a save area of its own.

   The exit routine must return to its caller using normal OS/VS conventions after restoring the registers.
3. On entry:

   Register 1 Points to a parameter list pointing to one single parameter, an interface control block.

   Register 13 Points to a register save area.

   Register 14 Contains the return address.

   Register 15 Contains the entry point address of your exit routine.
4. It is recommended that your exit routine be written and linked as reentrant.

When dynamically allocating the EKYRESLB DD statement, your exit routine must not specify deallocation at CLOSE. This is because DPROP opens and closes the EKYRESLB DD name more than once.

To avoid possible conflicts with STIMER and STIMERM macros generated when application programs perform synchronous data propagation, DPROP calls the exit in an MVS subtask created specifically for the call of your exit routine. DPROP attaches and detaches this subtask for every call of your exit routine.

Return Codes

When returning, your exit routine must provide a return code in register 15.

A nonzero return code results in an ABEND.

Telling DPROP about The EKYRESLB Dynamic Allocation Exit

To activate your EKYRESLB dynamic allocation exit, compile and link edit your exit routine with the load module name EKYDAEX0 into the //JOBLIB, //STEPLIB, and linklist LPA concatenation.

During DPROP Installation, your DPROP System Administrator can create a dummy, IEFBR14-type, EKYDAEX0 load module in your DPROP RESLIB. In this case, to use your real EKYDAEX0, one of the following must be done:

- The dummy, IEFBR14-type, EKYDAEX0 load module must be deleted from the DPROP RESLIB.
- The load module library containing your real EKYDAEX0 module must be concatenated ahead of the DPROP RESLIB in the //JOBLIB, STEPLIB, and LINKLIB LPA concatenation.

Sample EKYRESLB Dynamic Allocation Exit

The sample EKYRESLB dynamic allocation exit below is provided in the DPROP Sample Source Library (EKYSAMP) under the member name EKYEDA1A. To activate the dynamic allocation exit, you must link edit the load module as EKYDAEX0 in the //STEPLIB, //JOBLIB, linklist, or LPA concatenation.

```
         PRINT NOGEN
********** START OF SPECIFICATIONS ************************************
*    MODULE NAME = EKYEDA1A                                           *
*                                                                     *
*    DESCRIPTIVE NAME = SAMPLE 'EKYRESLB DYNAMIC ALLOCATION EXIT      *
*                       ROUTINE'                                      *
*                                                                     *
*    STATUS: V1 R2 M0                                                 *
*                                                                     *
*    FUNCTION = EKYEDA1A IS A SAMPLE DPROP USER EXIT ROUTINE          *
*               USED TO ALLOCATE DYNAMICALLY THE EKYRESLB             *
*               DD STATEMENT.                                         *
*                                                                     *
*               EKYEDA1A IS CALLED WITH ONE SINGLE PARAMETER:         *
*               THE EKYDAE PARAMETER BLOCK.                           *
*               IN THIS PARAMETER BLOCK THE FIELD DAECALL             *
*               CONTAINS THE CALL FUNCTION. THE CALL FUNCTION         *
*               IS EITHER:                                            *
*                 - 'AL' (= 'ALLOCATE')                               *
*                 - 'DE' (= 'DE-ALLOCATE')                            *
*               THE FUNCTIONS OF THIS SAMPLE EXIT ROUTINE CAN BE      *
*               SKETCHED AS FOLLOWS:                                  *
*                                                                     *
*               FOR AN 'AL' CALL FUNCTION  ('ALLOCATE'):              *
*               ----------------------------------------              *
*               WHEN BEING CALLED WITH AN 'AL' CALL-FUNCTION THIS     *
*               SAMPLE EXIT ROUTINE CHECKS IN INFORMATION PROVIDED    *
*               BY THE CALLER IN EKYDAE WHETHER //EKYRESLB            *
*               IS ALREADY ALLOCATED.                                 *
*                                                                     *
*                - IF //EKYRESLB IS ALREADY ALLOCATED, THE SAMPLE     *
*                  EXIT ROUTINE RETURNS WITHOUT FURTHER PROCESSING.   *
*                                                                     *
*                - IF //EKYRESLB IS NOT ALREADY ALLOCATED, THE        *
*                  SAMPLE EXIT ROUTINE USES MVS DYNALLOC SERVICES     *
*                  TO ALLOCATE DYNAMICALLY THE //EKYRESLB             *
*                  DD STATEMENT WITH A DISPOSITION OF 'SHR'.          *
*                                                                     *
*                  THE DATASET-NAME ALLOCATED TO //EKYRESLB IS        *
*                  A HARD-CODED/FIXED DATA-SET NAME. IN REAL-LIFE,    *
*                  YOUR INSTALLATION WILL PROBABLY PROVIDE SOME       *
*                  ADDITIONAL LOGIC ALLOWING TO ALLOCATE              *
*                  DIFFERENT/VARIABLE DATA-SET NAMES TO EKYRESLB.     *
*                  FOR EXAMPLE, THIS CAN BE ACHIEVED BY READING A     *
*                  SYS1.PARMLIB MEMBER CONTAINING THE DATASET-NAME    *
*                  TO BE ALLOCATED.                                   *
*                                                                     *
*               'DE' CALL FUNCTION  ('DE-ALLOCATE')                   *
*               -----------------------------------                   *
*               WHEN BEING CALLED WITH A  'DE' CALL-FUNCTION, THIS    *
*               SAMPLE EXIT ROUTINE CHECKS WHETHER THE ALLOCATION     *
*               OF //EKYRESLB WAS PERFORMED BY THE SAMPLE EXIT        *
*               ROUTINE.                                              *
*                                                                     *
*                - IF THIS IS NOT THE CASE, THE SAMPLE                *
*                  EXIT ROUTINE RETURNS WITHOUT FURTHER PROCESSING.   *
*                                                                     *
*                - ELSE, THE SAMPLE EXIT ROUTINE                      *
*                  USES MVS DYNALLOC SERVICES TO                      *
*                  DE-ALLOCATE DYNAMICALLY THE //EKYRESLB             *
*                  DD STATEMENT.                                      *
*                                                                     *
```

Figure 79 (Part 1 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
*             NOTES:                                                 *
*               1)  IF WRITING YOUR OWN EXIT ROUTINE, THERE IS NO    *
*                   REAL NEED TO DE-ALLOCATE THE //EKYRESLB          *
*                   DD-STATEMENT. FOR A 'DE' CALL FUNCTION, YOUR     *
*                   ROUTINE CAN RETURN WITHOUT ANY PROCESSING. IN    *
*                   THIS CASE, //EKYRESLB WILL REMAIN ALLOCATED      *
*                   UNTIL THE END OF THE JOBSTEP.                    *
*                                                                    *
*                   IN FACT, WE RECOMMEND THAT YOU DO ***NOT***      *
*                   DE-ALLOCATE THE //EKYRESLB DD-STATEMENT, SINCE   *
*                   THIS REDUCES THE AMOUNT OF CODING THAT YOU       *
*                   MUST PROVIDE FOR YOUR OWN EXIT ROUTINE.          *
*                                                                    *
*               2)  BE CAREFUL, IF IGNORING ABOVE RECOMMENDATION.    *
*                   YOUR EXIT SHOULD DE-ALLOCATE DURING              *
*                   A 'DE' CALL THE //EKYRESLB DD-STATEMENT ONLY IF  *
*                   THE ALLOCATION HAS ALSO BEEN DONE PREVIOUSLY BY  *
*                   YOUR EXIT. IGNORING THIS WARNING CAN RESULT IN   *
*                   SOME ENVIRONMENTS (FOR EXAMPLE MPP REGIONS)      *
*                   IN DIFFERENT/INCONSISTENT CONSECUTIVE            *
*                   ALLOCATIONS OF //EKYRESLB.                       *
*                                                                    *
*                                                                    *
*   ACTIVATION OF THIS EXIT ROUTINE=                                 *
*             THIS EXIT ROUTINE GETS ACTIVATED BY COMPILING AND      *
*             LINKING THIS EXIT-ROUTINE INTO                         *
*             THE USUAL JOBLIB/STEPLIB/LINKLIB-CONCATENATION USED    *
*             FOR THE EXECUTION OF YOUR DPROP JOBSTEPS.              *
*                                                                    *
*             NOTE THAT THE LOAD MODULE NAME OF THIS EXIT ROUTINE    *
*             MUST BE EKYDAEX0 (NOT EKYEDA1A) IN ORDER TO GET        *
*             INVOKED BY DPROP.                                      *
*                                                                    *
*      RESTRICTIONS:                                                 *
*             THIS EXIT ROUTINE SHOULD NOT PERFORM ANY FUNCTION      *
*             WHICH IS NOT SUPPORTED IN THE ENVIRONMENT IT           *
*             EXECUTES (FOR EXAMPLE, IF DPROP IS USED TO PERFORM     *
*             SYNCHRONOUS PROPAGATION IN MPP REGIONS, THEN THE       *
*             EXIT ROUTINE SHOULD NOT PERFORM ANY FUNCTION WHICH     *
*             IS NOT SUPPORTED BY IMS IN A MPP ENVIRONMENT).         *
*                                                                    *
*      REGISTER CONVENTIONS=                                         *
*                  R13= ADDRESS OF SAVE AREA                         *
*                  R12= MODULE BASE REGISTER                         *
*                  R11= BAS REGISTER TO CALL SUBROUTINE              *
*                  R9 = EKYDAE INTERFACE PARAMETER BLOCK             *
*      PATCH LABEL  = - (NONE)                                       *
*                                                                    *
*   MODULE TYPE = PROCEDURE                                          *
*      PROCESSOR = ASSEMBLER                                         *
*      MODULE SIZE = LESS THAN 1000 BYTES.                           *
*      ATTRIBUTES = REENTRANT                                        *
*      RMODE      = ANY                                              *
*      AMODE      = 31                                               *
*                                                                    *
*   ENTRY POINT = EKYEDA1A                                           *
*      PURPOSE = SEE FUNCTION                                        *
*      LINKAGE = STANDARD OS/VS ASSEMBLER LINKAGE CONVENTIONS.       *
*                                                                    *
*   INPUT  : R1 = POINTING TO A STANDARD PARAMETER ADDRESS LIST.     *
*                1ST AND ONLY PARAMETER: EKYDAE CONTROL-BLOCK        *
*                                                                    *
*   OUTPUT : FOR A 'AL' CALL, IF THE //EKYRESLB DD-STATEMENT HAS     *
*           BEEN ALLOCATED:                                          *
*                  - DAEALLOC IS SET TO 'Y'                          *
*                                                                    *
```

Figure 79 (Part 2 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
*    EXIT-NORMAL=                                                     *
*       STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.                  *
*       RETURN CODES = 0                                              *
*                                                                     *
*    EXIT-ERROR=                                                      *
*       STANDARD OS/VS ASSEMBLER RETURN CONVENTIONS.                  *
*       RETURN CODE  = 4                                              *
*                                                                     *
*                                                                     *
*    ABEND-CODE OF EKYEDA1A = 1106                                    *
*       ABEND-REASON CODES  = X'99999999'                             *
*                                                                     *
*    ERROR MESSAGES ISSUED BY EKYEDA1A                                *
*        EKYEDA1E   : FAILURE DURING DYNAMIC ALLOCATION.              *
*        EKYEDA2E   : INVALID-CALL FUNCTION.                          *
*        EKYEDA3E   : FAILURE DURING DYNAMIC DEALLOCATION.            *
*        ADDITIONAL ERROR-MESSAGES MIGHT BY ISSUED BY SVC 99/DYNALLOC.*
*                                                                     *
*                                                                     *
*                                                                     *
*    EXTERNAL REFERENCES                                              *
*                                                                     *
*       ROUTINES=      = NONE                                         *
*                                                                     *
*       CONTROL BLOCKS = DAE   INTERFACE CB FOR DYNALLOC EXIT ROUTINE *
*                        S99RB  SVC 99 REQUEST BLOCK                  *
*                        S99RBX SVC 99 REQUEST BLOCK EXTENSION        *
*                        S99TUNIT SVC 99 TEXT UNITS                   *
*                                                                     *
*                                                                     *
*    MACROS USED FROM MACRO LIBRARY=                                  *
*           SAVE     - SAVE REGISTERS                                 *
*           GETMAIN  - OS/VS GETMAIN                                  *
*           DYNALLOC - OS/VS SVC 99 CALL                              *
*                                                                     *
*           EKYDAE   - INTERFACE CONTROL-BLOCK FOR DYNALLOC           *
*                      EXIT ROUTINE.                                  *
*           IEFZB4D2 - OS/VS SVC 99 DYNALLOC KEYS                     *
*                                                                     *
*    CHANGE ACTIVITY= NONE                                            *
*                                                                     *
************* END OF SPECIFICATIONS ***********************************
************ LOGIC OF EKYEDA1A ***************************************
*                                                                    *
*                                                                    *
*    MAIN LINE LOGIC:                                                *
*    ================                                                *
*                                                                    *
*    1) MODULE ENTRY LOGIC:                                          *
*    ----------------------                                          *
*          - PROVIDE REGISTER EQUATES                                *
*                                                                    *
*          - GENERATE A MODULE SAVEID                                *
*                                                                    *
*          - SAVE REGISTERS AND ESTABLISH MODULE-BASE REGISTER       *
*                                                                    *
*          - LOAD ADDRESSES OF CALL PARAMETER                        *
*                                                                    *
*          - GETMAIN AN AREA CONTAINING                              *
*            A MODULE SAVE AREA AND MODULE WORKSPACE.                *
*            CLEAR THE GETMAINED AREA.                               *
*                                                                    *
*          - CHAIN MODULE SAVE AREA AND SAVE AREA OF CALLER.         *
*                                                                    *
```

Figure 79 (Part 3 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
*    2) FOR 'AL' (='ALLOCATE') CALLS                                  *
*    --------------------------------                                 *
*          - IF THE EKYRESLB DD-STATEMENT IS ALREADY ALLOCATED:       *
*            RETURN WITHOUT FURTHER PROCESSING.                       *
*                                                                     *
*          - IF THE EKYRESLB DD-STATEMENT IS NOT ALREADY ALLOCATED:   *
*                                                                     *
*              - PREPARE INFORMATION REQUIRED TO CALL THE MVS         *
*                DYNALLOC MACRO FOR DYNAMIC ALLOCATION OF             *
*                THE EKYRESLB DD STATEMENT WITH DISP=SHR.             *
*                                                                     *
*                THE PREPARED DYNALLOC CALL-PARAMETERS REQUEST        *
*                AMONG OTHER THAT MVS GENERATES AND WRITES ERROR      *
*                MESSAGES ABOUT DYNALLOC FAILURE.                     *
*                                                                     *
*              - ISSUE DYNALLOC MACRO.                                *
*                                                                     *
*              - IF RETURN-CODE FROM DYNALLOC IS NON-ZERO:            *
*                  -- ISSUE ERROR-MESSAGE                             *
*                  -- BRANCH TO THE RETURN-CODE 4 LOGIC (THIS WILL    *
*                     (RESULT IN A ABEND ISSUED BY DPROP).            *
*                                                                     *
*              - IF RETURN-CODE FROM DYNALLOC IS ZERO:                *
*                  -- RECORD THAT THE EKYRESLB DD STATEMENT HAS BEEN  *
*                     ALLOCATED BY EKYEDA1A.                          *
*                  -- BRANCH TO THE RETURN-CODE 0 LOGIC.              *
*                                                                     *
*                                                                     *
*    3) FOR 'DE' (='DE-ALLOCATE') CALLS                               *
*    ----------------------------------                               *
*          - IF IT IS NOT THIS SAMPLE EXIT ROUTINE WHICH ALLOCATED    *
*            PREVIOUSLY //EKYRESLB:                                   *
*              - THE EXIT RETURNS WITHOUT FURTHER PROCESSING.         *
*                                                                     *
*          - IF IT IS THIS SAMPLE EXIT ROUTINE WHICH ALLOCATED        *
*            PREVIOUSLY //EKYRESLB:                                   *
*                                                                     *
*              - PREPARE INFORMATION REQUIRED TO CALL THE MVS         *
*                DYNALLOC MACRO FOR DYNAMIC ALLOCATION OF             *
*                THE EKYRESLB DD STATEMENT WITH DISP=SHR.             *
*                                                                     *
*                THE PREPARED DYNALLOC CALL-PARAMETERS REQUEST        *
*                AMONG OTHER THAT MVS GENERATES AND WRITES ERROR      *
*                MESSAGES ABOUT DYNALLOC FAILURE.                     *
*                                                                     *
*              - ISSUE DYNALLOC MACRO.                                *
*                                                                     *
*              - BRANCH TO RETURN-CODE 0 LOGIC.                       *
*                                                                     *
*    4) RETURN LOGIC                                                  *
*    ----------------                                                 *
*          - FREEMAIN AREA CONTAING SAVE-AREA AND WORKSPACE.          *
*                                                                     *
*          - RESTORE REGISTERS OF THE CALLER                          *
*                                                                     *
*          - RETURN TO THE CALLER.                                    *
*                                                                     *
*                                                                     *
************ END-OF-LOGIC *********************************************
***********************************************************************
***********************************************************************
***********************************************************************
****                                                               ****
****          MODULE ENTRY LOGIC                                   ****
****                                                               ****
***********************************************************************
```

Figure 79 (Part 4 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
                                 263 ***********************************************************************
                                 264 ***********************************************************************

000000                           266 EKYEDA1A START
                                 267 *
                                 268 EKYEDA1A AMODE 31              EXIT EXPECTS TO BE CALLED IN AMODE-31
                                 269 EKYEDA1A RMODE ANY             EXIT CAN BE LOADED ANYWHERE
                                 270 *
                                 271 *---------------------------------------------------------------------*
                                 272 *        DEFINITION OF REGISTER EQUATES                               *
                                 273 *---------------------------------------------------------------------*
                                 274 *
                         00000   275 R0       EQU   0
                         00001   276 R1       EQU   1
                         00002   277 R2       EQU   2
                         00003   278 R3       EQU   3
                         00004   279 R4       EQU   4
                         00005   280 R5       EQU   5
                         00006   281 R6       EQU   6              ABEND REASON CODE
                         00007   282 R7       EQU   7
                         00008   283 R8       EQU   8
                         00009   284 R9       EQU   9              A(DAE)
                         0000A   285 R10      EQU   10
                         0000B   286 R11      EQU   11             BAS REGISTER TO CALL SUBROUTINES
                         0000C   287 R12      EQU   12             MODULE BASE REGISTER
                         0000D   288 R13      EQU   13             A(SAVEAREA / GETMAINED AREA)
                         0000E   289 R14      EQU   14
                         0000F   290 R15      EQU   15

                                 292 *---------------------------------------------------------------------*
                                 293 *        GENERATE SAVE-ID CONSISTING OF EXIT NAME,                    *
                                 294 *        COMPILATION DATE AND COMPILATION TIME.                       *
                                 295 *---------------------------------------------------------------------*

                                 297          LCLC  &SAVEID
                                 298 &SAVEID  SETC  'EKYEDA1A DPR120'.'-'.'&SYSDATE'.'-'.'&SYSTIME'

                                 300 *---------------------------------------------------------------------*
                                 301 *        SAVE REGISTERS AND ESTABLISH MODULE-BASE REGISTER            *
                                 302 *---------------------------------------------------------------------*

                                 304          SAVE  (14,12),,&SAVEID SAVE REGISTERS

000028 18CF                      313          LR    R12,R15        R12=ENTRY POINT OF THIS EXIT
                         00000   314          USING EKYEDA1A,R12   ESTABLISH BASE REGISTER

                                 316 *---------------------------------------------------------------------*
                                 317 *        LOAD ADDRESS OF CALL PARAMETERS                              *
                                 318 *---------------------------------------------------------------------*

00002A 5891 0000         00000   320          L     R9,0(R1)       LOAD ADDRESS OF 1ST CALL PARAMETERS
                         00000   321          USING EKYDAE,R9      R9=BASE FOR INTERFACE CONTROL BLOCK

                                 323 *---------------------------------------------------------------------*
                                 324 *         - GETMAIN AN AREA CONTAINING                                *
                                 325 *             -- OUR SAVE AREA                                        *
                                 326 *             -- MODULE WORKSPACE                                     *
                                 327 *         - CLEAR THE GETMAINED AREA WITH BINARY ZEROES               *
                                 328 *---------------------------------------------------------------------*

                                 330          GETMAIN RU,LV=GETML,LOC=ANY GETMAIN AN AREA

000048 18B1                      343          LR    R11,R1         R11=A(GETMAINED AREA)
```

Figure 79 (Part 5 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
00004A 1801                                   LR    R0,R1          SET UP
00004C 4110 00DF                000DF         LA    R1,GETML       ...FOR A
000050 1BFF                                   SR    R15,R15        ...ZEROING
000052 0E0E                                   MVCL  R0,R14         ...MVCL

                                     *-------------------------------------------------------------------*
                                     *       CHAIN TOGETHER OUR SAVEAREA AND THE HIGHER-LEVEL SAVEAREA    *
                                     *       AND LOAD INTO R13 THE ADDRESS OF OUR SAVEAREA                *
                                     *-------------------------------------------------------------------*

000054 50BD 0008                00008         ST    R11,8(R13)     CHAIN OUR SAVEAREA INTO HIGHER
000058 50DB 0004                00004         ST    R13,4(R11)     CHAIN HIGHER SAVEAREA INTO OUR
00005C 18DB                                   LR    R13,R11        R13=A(OUR SAVEAREA)
                                00000         USING GETM,R13       ESTABLISH BASE REGISTER FOR WORKAREA

                                     *-------------------------------------------------------------------*
                                     *       BRANCH DEPENDING ON CALL-FUNCTION.                           *
                                     *-------------------------------------------------------------------*

00005E D501 9008 C308 00008 00308             CLC   DAECALL,=CL2'AL' CALLED TO ALLOCATE EKYRESLB?
000064 4780 C076                00076         BE    ALLOC          ...YES>>>B
000068 D501 9008 C30A 00008 0030A             CLC   DAECALL,=CL2'DE' CALLED TO DE-ALLOCATE EKYRESLB?
00006E 4780 C184                00184         BE    DEALLOC        ...YES>>>B
000072 47F0 C242                00242         B     INVFUNC         CALL-FUNCTION IS INVALID
                                     ********************************************************************
                                     ********************************************************************
                                     ********************************************************************
                                     ****                                                            ****
                                     ****    'AL' (ALLOCATE) CALL                                    ****
                                     ****          - THE EXIT IS INVOKED TO PERFORM A DYNAMIC        ****
                                     ****            ALLOCATION OF THE EKYRESLB DD STATEMENT         ****
                                     ****                                                            ****
                                     ********************************************************************
                                     ********************************************************************
                                     ********************************************************************

000076                               ALLOC    DS    0H
000076 95E8 900A      0000A                   CLI   DAEALLOC,DAEALLOY       EKYRESLB ALREADY ALLOCATED?
00007A 4770 C088                00088         BNE   ALLOC20                 ...NO>>>LETS DO THE ALOCATION
00007E D700 9020 9020 00020 00020             XC    OURALLO,OURALLO         CLEAR OURALLO FLAG
000084 47F0 C294                00294         B     RETURN0                 ...AND RETURN

                                     *------------------------------------------------------------------*
                                     *       PERFORM THE DYNAMIC ALLOCATION BY PREPARING:                *
                                     *        - DYNALLOC REQUEST BLOCK                                   *
                                     *        - DYNALLOC REQUEST BLOCK EXTENSION                         *
                                     *        - DYNALLOC TEXT UNITS                                      *
                                     *       AND BY CALLING THE DYNALLOC MACRO.                          *
                                     *                                                                   *
                                     *       PLEASE REFER TO MVS/ESA DOCUMENTATION, IF YOU NEED          *
                                     *       EXPLANATIONS ON THIS SUBJECT (SEE 'MVS/ESA APPLICATION      *
                                     *       DEVELOPMENT GUIDE: AUTHORIZED ASSEMBLER LANGUAGE            *
                                     *       PROGRAMS (GC28-1645)' CHAPTER 'REQUESTING SVC 99            *
                                     *       FUNCTIONS').                                                *
                                     *------------------------------------------------------------------*

000088                               ALLOC20  DS    0H

                                     *------- PREPARE SVC 99 REQUEST BLOCK

000088 D792 D04C D04C 0004C 0004C             XC    Z99RB(Z99END-Z99RB),Z99RB CLEAR SVC 99 AREA

00008E 4110 D04C                0004C         LA    R1,Z99RB                GET POINTER TO RB AREA
000092 5010 D048                00048         ST    R1,Z99RBPTR             STORE ADDRESS INTO RBPTR
000096 9680 D048      00048                   OI    Z99RBPTR,X'80'          AND TURN HIGH BIT ON
00009A 9214 D04C      0004C                   MVI   Z99RBLN,Z99RBEND-Z99RB  SETUP CB LENGTH
```

Figure 79 (Part 6 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
00009E 9201 D04D      0004D        412         MVI   Z99VERB,Z99VRBAL        SETUP VERB CODE
0000A2 9200 D04E      0004E        413         MVI   Z99FLG11,0              SETUP FLAG BYTES
0000A6 4110 D084            00084  414         LA    R1,Z99TUPL              GET POINTER TO TEXT UNITS
0000AA 5010 D054            00054  415         ST    R1,Z99TXTPP             AND STORE INTO RB
0000AE 4110 D060            00060  416         LA    R1,Z99RBX               GET POINTER TO RB EXTENSION
0000B2 5010 D058            00058  417         ST    R1,Z99Z99X              AND STORE INTO RB

                                   419 *------- PREPARE REQUEST BLOCK EXTENSION

0000B6 D205 D060 C30C 00060 0030C  421         MVC   Z99EID,=CL6'S99RBX'     SETUP REQUEST BLOCK ID
0000BC 9201 D066      00066        422         MVI   Z99EVER,Z99RBXVR        SETUP VERSION NUMBER
0000C0 9284 D067      00067        423         MVI   Z99EOPTS,Z99EIMSG+Z99EWTP REQUEST MESSAGE WRITING
0000C4 9200 D06A      0006A        424         MVI   Z99EMGSV,Z99XINFO       SETUP MESSAGE LEVEL

                                   426 *------- PREPARE POINTER LIST TO UNIT TEXT-UNITS

0000C8 4110 D090            00090  428         LA    R1,Z99T1NIT             R1=A(1ST TEXT-UNIT)
0000CC 5010 D084            00084  429         ST    R1,Z99TUPT1             STORE A(1ST TEXT UNIT)
0000D0 4110 D0A0            000A0  430         LA    R1,Z99T2NIT             R1=A(2ND TEXT-UNIT)
0000D4 5010 D088            00088  431         ST    R1,Z99TUPT2             STORE A(2ND TEXT UNIT)
0000D8 4110 D0D8            000D8  432         LA    R1,Z99T3NIT             R1=A(3RD TEXT-UNIT)
0000DC 5010 D08C            0008C  433         ST    R1,Z99TUPT3             STORE A(3RD TEXT UNIT)
0000E0 9680 D08C      0008C        434         OI    Z99TUPT3,X'80'          INDICATE LAST POINTER

                                   436 *------- PREPARE DDNAME TEXT UNIT

0000E4 D201 D090 C312 00090 00312  438         MVC   Z99T1KEY,=AL2(DALDDNAM) SETUP UNIT KEY
0000EA D201 D092 C314 00092 00314  439         MVC   Z99T1NUM,=AL2(1)        SETUP NUMBER OF ENTRIES
0000F0 D201 D094 C316 00094 00316  440         MVC   Z99T1LNG,=AL2(8)        SETUP PARM LENGTH
0000F6 D207 D096 C2D0 00096 002D0  441         MVC   Z99T1DDN(8),=CL8'EKYRESLB' SETUP DDNAME IN PARM

                                   443 *------- PREPARE DSNAME TEXT UNIT

0000FC D201 D0A0 C318 000A0 00318  445         MVC   Z99T2KEY,=AL2(DALDSNAM) SETUP UNIT KEY
000102 D201 D0A2 C314 000A2 00314  446         MVC   Z99T2NUM,=AL2(1)        SETUP NUMBER OF ENTRIES
000108 D201 D0A4 C31A 000A4 0031A  447         MVC   Z99T2LNG,=AL2(44)       SETUP PARM LENGTH
00010E D22B D0A6 C2D8 000A6 002D8  448         MVC   Z99T2DSN(44),=CL44'KOE.DPM120.LOAD' DSNAME IN PARM

                                   450 *------- PREPARE DATASET STATUS TEXT UNIT

000114 D201 D0D8 C31C 000D8 0031C  452         MVC   Z99T3KEY,=AL2(DALSTATS) SETUP UNIT KEY
00011A D201 D0DA C314 000DA 00314  453         MVC   Z99T3NUM,=AL2(1)        SETUP NUMBER OF ENTRIES
000120 D201 D0DC C314 000DC 00314  454         MVC   Z99T3LNG,=AL2(1)        SETUP PARM LENGTH
000126 9208 D0DE      000DE        455         MVI   Z99T3ST,X'08'           SETUP DISPOSITION

                                   457 *------- LETS CALL SVC 99 FOR DYNAMIC ALLOCATION

00012A 4110 D048            00048  459         LA    R1,Z99RBPTR             R1=A(RB-POINTER)
                                   460         DYNALLOC ,                    AND CALL SVC 99

000130 12FF                        464         LTR   R15,R15                 IS ALL OK?
000132 4770 C13E            0013E  465         BNZ   ALERROR                 YES -> RETURN TO CALLER

                                   467 *-------------------------------------------------------------------*
                                   468 *       DYNAMIC ALLOCATION WAS SUCESSFULL.                          *
                                   469 *                                                                   *
                                   470 *       - RECORD IN OUR GETMAINED-AREA THAT THE DYNAMIC ALLOCATION  *
                                   471 *         WAS PERFORMED BY THIS EXIT ROUTINE.                       *
                                   472 *       - BRANCH TO RETURN WITH A RETURN-CODE 0.                    *
                                   473 *-------------------------------------------------------------------*

000136 96E8 9020      00020        475         OI    OURALLO,OURALLOY        EKYRESLB ALLOCATED BY US
00013A 47F0 C294            00294  476         B     RETURN0
```

Figure 79 (Part 7 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
                                   478 *-------------------------------------------------------------------*
                                   479 *       DYNAMIC ALLOCATION FAILED.                                  *
                                   480 *                                                                   *
                                   481 *       - ISSUE ERROR-MESSAGE.                                      *
                                   482 *       - BRANCH TO RETURN WITH A RETURN-CODE 4.                    *
                                   483 *-------------------------------------------------------------------*

00013E                             485 ALERROR  DS    0H
                                   486          WTO   'EKYEDA1A: DYNAMIC ALLOCATION OF //EKYRESLB FAILED',    C
                                                      ROUTCDE=11

000180 47F0 C29C           0029C   497          B     RETURN4
                                   499 ***********************************************************************
                                   500 ***********************************************************************
                                   501 ***********************************************************************
                                   502 ****                                                              ****
                                   503 ****     'DE' (DE-ALLOCATE) CALL                                  ****
                                   504 ****           - THE EXIT IS INVOKED TO ALLOW A DYNAMIC           ****
                                   505 ****             DE-ALLOCATION OF THE EKYRESLB DD STATEMENT       ****
                                   506 ****                                                              ****
                                   507 ***********************************************************************
                                   508 ***********************************************************************
                                   509 ***********************************************************************

000184                             511 DEALLOC  DS    0H
000184 91E8 9020      00020        512          TM    OURALLO,OURALLOY        DID WE ALLOCATE?
000188 4780 C294           00294   513          BZ    RETURN0                 ....NO>>>RETURN

                                   515 *-------------------------------------------------------------------*
                                   516 *       PERFORM THE DYNAMIC DE-ALLOCATION BY PREPARING:             *
                                   517 *        - DYNALLOC REQUEST BLOCK                                   *
                                   518 *        - DYNALLOC REQUEST BLOCK EXTENSION                         *
                                   519 *        - DYNALLOC TEXT UNITS                                      *
                                   520 *       AND BY CALLING THE DYNALLOC MACRO.                          *
                                   521 *                                                                   *
                                   522 *       PLEASE REFER TO MVS/ESA DOCUMENTATION, IF YOU NEED          *
                                   523 *       EXPLANATIONS ON THIS SUBJECT.                               *
                                   524 *-------------------------------------------------------------------*

                                   526 *------- PREPARE SVC 99 REQUEST BLOCK

00018C D792 D04C D04C 0004C 0004C  528          XC    Z99RB(Z99END-Z99RB),Z99RB CLEAR SVC 99 RB/PARMS

000192 4110 D04C           0004C   530          LA    R1,Z99RB                GET POINTER TO RB AREA
000196 5010 D048           00048   531          ST    R1,Z99RBPTR             STORE ADDRESS INTO RBPTR
00019A 9680 D048      00048        532          OI    Z99RBPTR,X'80'          AND TURN HIGH BIT ON
00019E 9214 D04C      0004C        533          MVI   Z99RBLN,Z99RBEND-Z99RB  SETUP CB LENGTH
0001A2 9202 D04D      0004D        534          MVI   Z99VERB,Z99VRBUN        SETUP VERB CODE
0001A6 9200 D04E      0004E        535          MVI   Z99FLG11,0              SETUP FLAG BYTES
0001AA 4110 D084           00084   536          LA    R1,Z99TUPL              GET POINTER TO TEXT UNITS
0001AE 5010 D054           00054   537          ST    R1,Z99TXTPP             AND STORE INTO RB
0001B2 4110 D060           00060   538          LA    R1,Z99RBX               GET POINTER TO RB EXTENSION
0001B6 5010 D058           00058   539          ST    R1,Z99Z99X              AND STORE INTO RB

                                   541 *------- PREPARE REQUEST BLOCK EXTENSION

0001BA D205 D060 C30C 00060 0030C  543          MVC   Z99EID,=CL6'S99RBX'     SETUP REQUEST BLOCK ID
0001C0 9201 D066      00066        544          MVI   Z99EVER,Z99RBXVR        SETUP VERSION NUMBER
0001C4 9284 D067      00067        545          MVI   Z99EOPTS,Z99EIMSG+Z99EWTP REQUEST MESSAGE WRITING
0001C8 9200 D06A      0006A        546          MVI   Z99EMGSV,Z99XINFO       SETUP MESSAGE LEVEL

                                   548 *------- PREPARE POINTER LIST TO UNIT TEXT-UNITS

0001CC 4110 D090           00090   550          LA    R1,Z99T1NIT             R1=A(1ST TEXT UNIT)
0001D0 5010 D084           00084   551          ST    R1,Z99TUPT1             STORE A(1ST TEXT UNIT)
0001D4 9680 D084      00084        552          OI    Z99TUPT1,X'80'          INDICATE LAST POINTER
```

Figure 79 (Part 8 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
                                   554 *------- PREPARE DDNAME TEXT UNIT

0001D8 D201 D090 C312 00090 00312  556          MVC   Z99T1KEY,=AL2(DALDDNAM) SETUP UNIT KEY
0001DE D201 D092 C314 00092 00314  557          MVC   Z99T1NUM,=AL2(1)        SETUP NUMBER OF ENTRIES
0001E4 D201 D094 C316 00094 00316  558          MVC   Z99T1LNG,=AL2(8)        SETUP PARM LENGTH
0001EA D207 D096 C2D0 00096 002D0  559          MVC   Z99T1DDN(8),=CL8'EKYRESLB' SETUP DDNAME IN PARM

                                   561 *------- LETS CALL SVC 99 FOR DYNAMIC DE-ALLOCATION

0001F0 4110 D048            00048  563          LA    R1,Z99RBPTR             R1=A(RB-POINTER)
                                   564          DYNALLOC ,                    AND CALL SVC 99

0001F6 12FF                        568          LTR   R15,R15                 DE-ALLOCATION OK?
0001F8 4780 C294            00294  569          BZ    RETURN0                 ...YES>>>RETURN

                                   571 *---------------------------------------------------------------------*
                                   572 *        DYNAMIC DE-ALLOCATION FAILED                                 *
                                   573 *                                                                     *
                                   574 *        - ISSUE ERROR-MESSAGE.                                       *
                                   575 *        - BRANCH TO RETURN WITH A RETURN-CODE 0                      *
                                   576 *          (SINCE DE-ALLOCATION FAILURES DO NOT PREVENT               *
                                   577 *          SUCCESSFUL DPROP OPERATIONS, IT IS BY PURPOSE THAT WE      *
                                   578 *          RETURN WITH RC=0 -- AS OPPOSED TO RC=4).                   *
                                   579 *---------------------------------------------------------------------*

                                   581          WTO   'EKYEDA3E: DYNAMIC DE-ALLOCATION OF //EKYRESLB FAILED', C
                                                      ROUTCDE=11

00023E 47F0 C294            00294  592          B     RETURN0                 RETURN WITH ZERO RC
                                   594 ***********************************************************************
                                   595 ***********************************************************************
                                   596 ***********************************************************************
                                   597 ****                                                               ****
                                   598 ****     INVALID CALL FUNCTION IN DAECALL.                         ****
                                   599 ****                                                               ****
                                   600 ***********************************************************************
                                   601 ***********************************************************************
                                   602 ***********************************************************************

000242                             604 INVFUNC  DS    0H
                                   605          WTO   'EKYEDA2E: INVALID CALL-FUNCTION FOR EKYEDA1A',        C
                                                      ROUTCDE=11

00027E 5860 C304            00304  616          L     R6,=X'99999999'    R6= ABEND REASON CODE

                                   618          ABEND 1106,REASON=(R6),DUMP
                                   627 ***********************************************************************
                                   628 ***********************************************************************
                                   629 ***********************************************************************
                                   630 ****                                                               ****
                                   631 ****     RETURN LOGIC:                                             ****
                                   632 ****          - RETURN TO CALLER OF EXIT                           ****
                                   633 ****                                                               ****
                                   634 ***********************************************************************
                                   635 ***********************************************************************
                                   636 ***********************************************************************
                                   637 ***********************************************************************

000294                             639 RETURN0  DS    0H
000294 41F0 0000            00000  640          LA    R15,0               LOAD 0 AS RETURN-CODE
000298 47F0 C2A0            002A0  641          B     RETURN99

00029C                             643 RETURN4  DS    0H
00029C 41F0 0004            00004  644          LA    R15,4               LOAD 4 AS RETURN-CODE
```

Figure 79 (Part 9 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
0002A0                              646 RETURN99 DS    0H
0002A0 181D                         647          LR    R1,R13             R1=A(AREA TO BE FREEMAINED)
0002A2 58DD 0004           00004    648          L     R13,4(R13)         R13=A(HIGHER SAVE AREA)
0002A6 183F                         649          LR    R3,R15             SAVE RETURN-CODE INTO R3
                                    650          FREEMAIN RU,LV=GETML,A=(R1)

0002C0 18F3                         663          LR    R15,R3             RESTORE RETURN-CODE INTO R15
0002C2 980C D014           00014    664          LM    R0,R12,20(R13)     RELOAD REGISTERS OF CALLER
0002C6 58ED 000C           0000C    665          L     R14,12(R13)        RELOAD REGISTER 14 OF CALLER
0002CA 9601 D00F      0000F         666          OI    15(R13),X'01'      SET RETURN INDICATION
0002CE 07FE                         667          BR    R14                RETURN TO CALLER


0002D0                              669          LTORG
0002D0 C5D2E8D9C5E2D3C2             670                =CL8'EKYRESLB'
0002D8 D2D6C54BC4D7D4F1             671                =CL44'KOE.DPM120.LOAD'
000304 99999999                     672                =X'99999999'
000308 C1D3                         673                =CL2'AL'
00030A C4C5                         674                =CL2'DE'
00030C E2F9F9D9C2E7                 675                =CL6'S99RBX'
000312 0001                         676                =AL2(DALDDNAM)
000314 0001                         677                =AL2(1)
000316 0008                         678                =AL2(8)
000318 0002                         679                =AL2(DALDSNAM)
00031A 002C                         680                =AL2(44)
00031C 0004                         681                =AL2(DALSTATS)
                                    683          EKYDAE   ,               EXIT INTERFACE CONTROL BLOCK


000120                     00020    749          ORG   DAEUSER
000020 00                           750 OURALLO  DC    X'00'              ALLOCATION FLAG
                           000E8    751 OURALLOY EQU   C'Y'               ...EKYRESLB ALLOCATED BY EKYEDA1A
000021                     00120    752          ORG
                                    754 ***********************************************************************
                                    755 ***********************************************************************
                                    756 ***********************************************************************
                                    757 ****     DESCRIPTION OF GETMAINED AREA CONTAINING AMONG OTHER:     ***
                                    758 ****          - SAVEAREA                                           ***
                                    759 ****          - EXIT WORKSPACE                                     ***
                                    760 ***********************************************************************
                                    761 ***********************************************************************
                                    762 ***********************************************************************

000000                              764 GETM     DSECT
                                    765 ***********************************************************************
                                    766 *        REGISTER SAVEAREA                                            *
                                    767 ***********************************************************************
000000                              768 SAVE     DS    18F'0'             REGISTER SAVEAREA

                                    770 ***********************************************************************
                                    771 *        WORK SPACE FOR EXIT                                          *
                                    772 ***********************************************************************

                                    774 *---------------------------------------------------------------------*
                                    775 *        SVC 99 REQUEST BLOCK-POINTER                                 *
                                    776 *---------------------------------------------------------------------*
000048 00000000                     777 Z99RBPTR DC     F'0'                  REQUEST BLOCK POINTER


                                    779 *---------------------------------------------------------------------*
                                    780 *        SVC 99 REQUEST BLOCK                                         *
                                    781 *---------------------------------------------------------------------*
00004C                              782 Z99RB    DS     0F
00004C                              783 Z99RBLN  DS     CL1                   LENGTH OF REQUEST BLOCK
00004D                              784 Z99VERB  DS     CL1                   VERB CODE
                           00001    785 Z99VRBAL EQU    X'01'                 ALLOCATION
```

Figure 79 (Part 10 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
                            00002   786 Z99VRBUN EQU    X'02'                  UNALLOCATION
                            00003   787 Z99VRBCC EQU    X'03'                  CONCATENATION
                            00004   788 Z99VRBDC EQU    X'04'                  DECONCATENATION
                            00005   789 Z99VRBRI EQU    X'05'                  REMOVE IN-USE
                            00006   790 Z99VRBDN EQU    X'06'                  DDNAME ALLOCATION
                            00007   791 Z99VRBIN EQU    X'07'                  INFORMATION RETRIEVAL
00004E                              792 Z99FLAG1 DS     0CL2                   FLAGS
00004E                              793 Z99FLG11 DS     CL1                    FIRST FLAGS BYTE
00004F                              794 Z99FLG12 DS     CL1                    SECOND BYTE OF FLAGS

000050                              796 Z99RSC   DS     0CL4                   REASON CODE FIELDS
000050                              797 Z99ERROR DS     XL2                    ERROR REASON CODE
000052                              798 Z99INFO  DS     XL2                    INFORMATION REASON CODE

000054                              800 Z99TXTPP DS     F                      ADDR OF LIST OF TEXT UNIT PTRS
000058                              801 Z99Z99X  DS     F                      ADDR OF REQ BLK EXTENSION

00005C                              803 Z99FLAG2 DS     0CL4                   FLAGS FOR AUTHORIZED FUNCTIONS
00005C                              804 Z99FLG21 DS     CL1                    FIRST BYTE OF FLAGS
                            00040   805 Z99WTDSN EQU    X'40'                  ALLOC FUNCTION-WAIT FOR DSNAME
                            00010   806 Z99WTUNT EQU    X'10'                  ALLOC FUNCTION-WAIT FOR UNITS
00005D                              807 Z99FLG22 DS     CL1                    SECOND BYTE OF FLAGS
00005E                              808 Z99FLG23 DS     CL1                    THIRD BYTE OF FLAGS
00005F                              809 Z99FLG24 DS     CL1                    FOURTH BYTE OF FLAGS
                            00060   810 Z99RBEND EQU    *                      END MARKER


                                    812 *---------------------------------------------------------------------*
                                    813 *       SVC 99 REQUEST BLOCK EXTENSION
                                    814 *---------------------------------------------------------------------*
000060                              815 Z99RBX   DS     0D                     REQUEST BLOCK EXTENSION
000060                              816 Z99EID   DS     CL6                    CONTROL BLOCK ID ='S99RBX'
000066                              817 Z99EVER  DS     CL1                    VERSION NUMBER
                            00001   818 Z99RBXVR EQU    X'01'                  CURRENT VERSION NUMBER
000067                              819 Z99EOPTS DS     CL1                    PROCESSING OPTIONS
                            00080   820 Z99EIMSG EQU    X'80'                  ISSUE MSG BEFORE RETURNING
                                    821 *                                      TO CALLER
                            00004   822 Z99EWTP  EQU    X'04'                  USE WTO FOR MESSAGE OUTPUT
000068                              823          DS     CL2                    SUBPOOL FOR MESSAGE BLOCKS
00006A                              824 Z99EMGSV DS     CL1                    SEVERITY LEVEL FOR MESSAGES
                                    825 *                                      PROCESSING
                            00000   826 Z99XINFO EQU    X'00'                  INFORMATIONAL MSG SEVERITY
                            00004   827 Z99XWARN EQU    X'04'                  WARNING MESSAGE SEVERITY
                            00008   828 Z99XSEVE EQU    X'08'                  SEVERE MESSAGE SEVERITY
00006B                              829          DS     CL1                    NUMBER OF MESSAGE BLOCKS
                                    830 *                                       RETURNED
00006C                              831 Z99ECPPL DS     6F                     ADDRESS OF CPPL


                                    833 *---------------------------------------------------------------------*
                                    834 *        SVC 99 TEXT UNIT POINTER LIST
                                    835 *---------------------------------------------------------------------*

000084                              837 Z99TUPL  DS     0F                     TEXT UNIT POINTER LIST
000084                              838 Z99TUPT1 DS     F                      POINTER TO 1ST TEXT UNIT
000088                              839 Z99TUPT2 DS     F                      POINTER TO 2ND TEXT UNIT
00008C                              840 Z99TUPT3 DS     F                      POINTER TO 3RD TEXT UNIT


                                    842 *---------------------------------------------------------------------*
                                    843 *       SVC 99 TEXT UNIT'S
                                    844 *---------------------------------------------------------------------*
```

Figure 79 (Part 11 of 12). Sample EKYRESLB Dynamic Allocation Exit

```
000090                      846 Z99T1NIT DS     0D                     1ST TEXT UNIT -- DDNAME
000090                      847 Z99T1KEY DS     XL2                    KEY
000092                      848 Z99T1NUM DS     XL2                    NO. OF ENTRIES
000094                      849 Z99T1ENT DS     0C                     ENTRY OF LENGTH+PARAMETER
000094                      850 Z99T1LNG DS     XL2                    LENGH OF PARAMTER
000096                      851 Z99T1DDN DS     CL8                    DDN PARAMETER

0000A0                      853 Z99T2NIT DS     0D                     2ND TEXT UNIT -- DSNAME
0000A0                      854 Z99T2KEY DS     XL2                    KEY
0000A2                      855 Z99T2NUM DS     XL2                    NO. OF ENTRIES
0000A4                      856 Z99T2ENT DS     0C                     ENTRY OF PARAMETER
0000A4                      857 Z99T2LNG DS     XL2                    LENGH OF 1ST (OR ONLY) PARAMETER
0000A6                      858 Z99T2DSN DS     CL44                   DSN PARAMETER

0000D8                      860 Z99T3NIT DS     0D                     3RD TEXT UNIT -- DATASET STATUS
0000D8                      861 Z99T3KEY DS     XL2                    KEY
0000DA                      862 Z99T3NUM DS     XL2                    NO. OF ENTRIES
0000DC                      863 Z99T3ENT DS     0C                     LENGTH OF PARAMETER
0000DC                      864 Z99T3LNG DS     XL2                    LENGH OF 1ST (OR ONLY) PARAMETER
0000DE                      865 Z99T3ST  DS     CL1                    STATUS PARAMETER

                    000DF   867 Z99END   EQU    *                      END OF SVC 99 INFO


                    000DF   869 GETML    EQU   *-GETM                  LENGTH OF GETMAINED AREA
                            871          IEFZB4D2 ,                    TEXT UNIT MNEMONICS
000000                     1155          END   EKYEDA1A
```

Figure 79 (Part 12 of 12). Sample EKYRESLB Dynamic Allocation Exit

Chapter 7. TSMF Callable Interface

This Chapter describes the timestamp marker facility (TSMF) callable interface. This interface is used with LOG-ASYNC.

The timestamp marker facility (TSMF) callable interface allows a user application program to create a stop timestamp marker (TSM) for one or more propagation groups. Refer to *IMS DPROP Reference* for details on the use of TSMs.

The user application program can pass the stop timestamp in ISO/DB2 format (local time) or in MVS TOD time format (GMT time).

The user application program must include the object module EKYT099X in its link-edit. EKYT099X is an assembler module provided with IMS DPROP to dynamically link the user application program with the TSMF. This means that if changes are made to IMS DPROP, the user application program does not need to be relinked.

The TSMF callable interface provides an alternative to the SCU for creating group stop timestamps. The JCL to run the user application program should fulfill the requirements for the JCL used to run the SCU with the CREATETSM STOP control statement. Refer to *IMS DPROP Reference* for details on using the SCU.

TSMF Callable Interface Parameters

The user application program invokes one of two entry points within EKYT099X, depending on the format of timestamp being passed:

```
EKYB097X (RC,TODTIME,ID,GRPLIST_COUNT,GRPLIST_ARRAY)
EKYB098X (RC,DB2TIME,ID,GRPLIST_COUNT,GRPLIST_ARRAY)
```

The parameters to be used when you call EKYT099X are detailed in Figure 80.

Figure 80. Parameters passed to the TSMF callable interface

Parameter	Type	No. of Bytes	Purpose
RC	BIN(31)	4	Passes the return code back to the caller.
DB2TIME	BIT(64)	8	Contains ISO/DB2 format timestamp.
TODTIME	CHAR(26)	26	Contains MVS TOD format timestamp.
ID	CHAR(8)	8	Contains the timestamp ID, left aligned, padded with blanks. All blanks means that an ID is not supplied.
GRPLIST_COUNT	BIN(31)	4	Contains the number of group IDs.
GRPLIST_ARRAY	CHAR(8)	8	Array of group IDs. Each group ID is 8 bytes, left-aligned, padded with blanks.

Calling the TSMF Callable Interface from PL/I

To call the callable interface from PL/I you must declare the assembler module (EKYT97X or EKYT98X) as an external module and then declare the variables and array used in the call to the TSMF callable interface. Refer to Figure 81 for an example of these declarations.

```
/* DECLARE THE ASSEMBLER MODULE (EKYT98X) AS AN EXTERNAL MODULE.      */

DCL  EKYT98X
     ENTRY
       (FIXED BIN(31),        /* return code                          */
        CHAR(26),             /* USERTIME in ISO/DB2 format           */
        CHAR(8),              /* TSMID                                */
        FIXED BIN(31),        /* Count of group IDs                   */
        (3) CHAR(8))          /* Array of group IDs, in this case 3   */
     EXTERNAL OPTIONS(ASSEMBLER INTER);



/* DECLARE LOCAL VARIABLES                                            */

DCL CURRENT_TSTAMP  CHAR(26)       INIT('1993-01-01-00.00.00.000000');
........
........


/* DECLARE THE VARIABLES USED IN THE CALL TO THE TSMF CALLABLE        */
/* INTERFACE.                                                         */

DCL 1 TSMF,
     2 RC           FIXED BIN(31)  INIT(0000),
     2 USERTIME     CHAR(26)       INIT('0001-01-01-01.01.01.000001'),
     2 TSMID        CHAR(8)        INIT('DEFAULT '),
     2 GRPLIST_COUNT FIXED BIN(31)  INIT(0003);


/* DECLARE THE ARRAY USED IN THE CALL TO THE TSMF.                    */
/* IN THIS EXAMPLE AN ARRAY WITH THREE GROUP NAMES IS USED. YOU CAN    */
/* DEFINE AS MANY GROUP NAMES AS YOU WISH BY MAKING THE ARRAY BIGGER   */
/* AND PASSING THE NUMBER OF GROUP NAMES IN THE GRPLIST_COUNT VARIABLE.*/

DCL GRPLIST (3)      CHAR(8)       INIT('GROUP01 ','GROUP02 ','GROUP3 ');
```

Figure 81. TSMF Callable Interface, Declarations for PL/I

Figure 82 is an example of how to call the TSMF callable interface from a PL/I program by using EKYT98X. The PL/I program uses a timestamp which is in DB2 format.

```
/* SET THE VARIABLE VALUES AS APPROPRIATE TO YOUR SITUATION          */

TSMF.USERTIME = CURRENT_TSTAMP ;
..........
..........


/* CALL THE TSMF CALLABLE INTERFACE PASSING A TIMESTAMP IN DB2 FORMAT */

CALL EKYT98X(TSMF.RC,
             TSMF.USERTIME,
             TSMF.TSMID,
             TSMF.GRPLIST_COUNT,
             GRPLIST ) ;


/* CHECK THE RETURN CODE FROM THE TSMF CALLABLE INTERFACE AND        */
/* HANDLE ANY ERRORS WHICH OCCUR.                                    */

IF TSMF.RC ¬= 0 THEN
   DO;
     /* handle error */
   END;
END;
```

Figure 82. TSMF Callable Interface, Call from a PL/I Program

Calling the TSMF Callable Interface from COBOL

To call the callable interface from COBOL you must declare the local variables and then declare the variables used in the call to the TSMF callable interface. Refer to Figure 83 for an example of this.

```
* DECLARE LOCAL VARIABLES                                             *

 WORKING-STORAGE SECTION.

 01  CURRENT-TSTAMP              PIC(26)   VALUE
         '1993-01-01-00.00.00.000000'.
 01  ........
 01  ........


* DECLARE THE VARIABLES USED IN THE CALL TO THE TSMF CALLABLE         *
* INTERFACE.                                                          *
* IN THIS EXAMPLE THREE GROUP NAMES ARE SPECIFIED IN THE              *
* VARIABLE TS-GROUPS. YOU CAN DEFINE AS MANY GROUP NAMES AS YOU       *
* WISH BY MAKING THE VARIABLE TS-GROUPS BIGGER AND PASSING THE        *
* NUMBER OF GROUP NAMES IN THE TS-GROUP-COUNT VARIABLE.               *

 01  TSMF-PARMETERS.
     03  TS-RETURN-CODE          PIC 9(8)  COMP VALUE ZERO.
     03  TS-USERTIME             PIC X(26) VALUE
         '1994-02-17-13.00.00.000000'.
     03  TS-TSMID                PIC X(8)  VALUE
         'TSM00003'.
     03  TS-GROUP-COUNT          PIC 9(8)  COMP VALUE 3.
     03  TS-GROUPS               PIC X(24) VALUE
         'GROUP01 GROUP02 GROUP03 '.
```

Figure 83. TSMF Callable Interface, Declarations for COBOL

Figure 84 is an example of how to call the TSMF callable interface from a COBOL program by using EKYT98X. The PL/I program uses a timestamp which is in DB2 format.

```
* SET THE VARIABLE VALUES AS APPROPRIATE TO YOUR SITUATION            *

     TS-USERTIME = CURRENT-TSTAMP.
     ..........
     ..........


* CALL THE TSMF CALLABLE INTERFACE PASSING A TIMESTAMP IN DB2         *
* FORMAT                                                              *

     CALL 'EKYT98X' USING BY REFERENCE
                         TS-RETURN-CODE,
                         TS-USERTIME,
                         TS-TSMID,
                         TS-GROUP-COUNT,
                         TS-GROUPS.

     IF  TS-RETURN-CODE NOT EQUAL ZERO THEN
         * handle error *
```

Figure 84. TSMF Callable Interface, Call from a COBOL Program

Return Codes from the TSMF Callable Interface

The TSMF Callable Interface provides the following return codes:

Code	Meaning
0	Successful creation of stop timestamp for group(s)
4	Warning message has been issued. One or more of the groups may already have a group stop timestamp equal to the timestamp passed by the user application.
8	Error message has been issued. The group stop timestamp is not created. This result occurs when there is insufficient information supplied to the callable interface. For example: *Unable to Open SCF* means that the user did not supply the //EKYSCF DD statement.
12	Error message has been issued. Invalid parameter passed by the user application.
16	Error message has been issued. The group stop timestamp was not created, probably because of an internal DPROP error. It is unlikely that this error would occur as a result of invalid data supplied by the user application.
20	Error message has been issued The group stop timestamp was not created. This error can probably be traced to environmental considerations that are not specific to the request. For example: *Out of Storage* means that the request would complete normally if there was only one user or if sufficient resources were supplied to the system.

Chapter 8. EMF Callable Interface

The event marker facility (EMF) callable interface allows a user application program to create an event marker (EM) for one or more Propagation Data Streams. Refer to *IMS DPROP Reference* for details on the use of EMs.

The EMF callable interface provides an alternative to the Capture System Utility (CUT) for creating Event Markers. The JCL to run the user application program should fulfill the requirements for the JCL used to run the CUT with the EM control statement. Refer to *IMS DPROP Reference* for details on using the CUT.

Note: when the EMF callable interface is called by IMS Batch Application Programs or by a non-IMS Batch Application Programs that isssue their own MQSeries calls, then these Application Programs:

- must issue their MQSeries calls through the use of the CSQBRSTB batch stub of MQSeries (this is the RRS batch stub, that provides a RRS-based two phase commit coordination between multiple Resource Managers).
- should not issue following types of MQSeries calls: MQCMIT and MQBACK.

EMF Callable Interface Parameters

The user application program invokes EKYI950X as follows:

```
EKYI950X (RC,RESERVD,ID,PRSTREAM_COUNT,PRSTREAM_ARRAY)
```

The parameters to be used when you call EKYI950X are detailed in Figure 85.

Figure 85. Parameters passed to the EMF callable interface

Parameter	Type	No. of Bytes	Purpose
RC	BIN(31)	4	Passes the return code back to the caller.
RESERVD	CHAR(26)	26	A reserved Field
ID	CHAR(8)	8	Contains the Event Marker ID, left aligned, padded with blanks.
PRSTREAM_COUNT	BIN(31)	4	Contains the number of PRSTREAM Names.
PRSTREAM_ARRAY	CHAR(8)	8	Array of PRSTREAM Names. Each PRSTREAM Name is 8 bytes, left-aligned, padded with blanks.

Calling the EMF Callable Interface from COBOL

To call the callable interface from COBOL you must declare the local variables and then declare the variables used in the call to the EMF callable interface. Refer to Figure 86 for an example of this.

```
* DECLARE LOCAL VARIABLES                                         *

 WORKING-STORAGE SECTION.

 01  .............
 01  .............



* DECLARE THE VARIABLES USED IN THE CALL TO THE EMF CALLABLE      *
* INTERFACE.                                                       *
* IN THIS EXAMPLE THREE PRSTREAM NAMES ARE SPECIFIED IN THE       *
* VARIABLE TS-PRSTREAMS. YOU CAN DEFINE AS MANY PRSTREAM's AS YOU *
* WISH BY MAKING THE VARIABLE TS-PRSTREAMS BIGGER AND PASSING THE *
* NUMBER OF PRSTREAM NAMES IN THE TS-PRSTREAM-COUNT VARIABLE.     *

 01  EMF-PARMETERS.
     03  TS-RETURN-CODE         PIC 9(8)  COMP VALUE ZERO.
     03  TS-RESERVD             PIC X(26) VALUE
         '                          '.
     03  TS-EMID                PIC X(8)  VALUE
         'EM00003'.
     03  TS-PRSTREAM-COUNT      PIC 9(8)  COMP VALUE 3.
     03  TS-PRSTREAMS           PIC X(24) VALUE
         'PRSTR01 PRSTR02 PRSTR03 '.
```

Figure 86. EMF Callable Interface, Declarations for COBOL

Figure 87 is an example of how to call the EMF callable interface from a COBOL program by using EKYI950X.

```
* CALL THE EMF CALLABLE INTERFACE                                 *

     CALL 'EKYI950X' USING BY REFERENCE
                         TS-RETURN-CODE,
                         TS-RESERVD,
                         TS-EMID,
                         TS-PRSTREAM-COUNT,
                         TS-PRSTREAMS.

     IF  TS-RETURN-CODE NOT EQUAL ZERO THEN
         * handle error *
```

Figure 87. EMF Callable Interface, Call from a COBOL Program

Return Codes from the EMF Callable Interface

The EMF Callable Interface provides the following return codes:

Code **Meaning**

0 Successful creation of Event Marker.

4 Warning: the requested Event Marker has not been created, for example, because the IMS DPROP Capture System is in emergency stopped status or because the Jobstep executes with a 'PROP OFF' Control Statement in the //EKYIN File.

8 Error: the requested Event Marker has not been created, for example, because a specified PRSTREAM name is not defined in the //EKYTRANS File.

Chapter 9. User-Implemented Asynchronous Data Propagation (USER-ASYNC)

IMS DPROP Version 3 supports two methods of asynchronous propagation: MQ-ASYNC and LOG-ASYNC. These methods of asynchronous propagation are fully described in the appropriate *Administrators Guide* for your propagation mode.

This chapter describes a third method of asynchronous propagation: USER-ASYNC. USER-ASYNC propagation is implemented by combining IMS DPROP components with user-provided programs. USER-ASYNC propagation was previously documented in the IMS DataPropagator for OS/390 and z/OS library when IMS DataPropagator for OS/390 and z/OS did not support either MQ-ASYNC nor LOG-ASYNC. With the advent of MQ-ASYNC and of LOG-ASYNC, you will no longer be required to develop programs to implement your own USER-ASYNC solutions. Instead, you can use MQ-ASYNC or LOG-ASYNC methods.

However, if you still want to implement your own solution, this chapter outlines what is required to develop a USER-ASYNC solution.

User asynchronous propagation can be based on either of the following :

- The IMS Asynchronous Data Capture function to harden the data on the log
- A user-written IMS Data Capture exit routine to capture the data and harden it

Refer to the following for information on the IMS Asynchronous Data Capture function and user-written IMS Data Capture exit routines:

- *IMS/ESA Administration Guide: Database Manager*
- *IMS/ESA Utilities Reference: Database Manager*
- *IMS/ESA Customization Guide*

For a detailed description of the log records written by the IMS Asynchronous Data Capture function, see:

- *IMS/ESA Customization Guide*

Implementation Based on IMS Asynchronous Data Capture Function

IMS application programs update the IMS databases. The IMS Asynchronous Data Capture function writes the changed data to the IMS log.

Later, a program that you write gathers the changed data from the IMS log data sets. This program is often referred to as the *selector*. It selects and gathers changed data to be propagated from all those IMS logs that contain changed data. It makes the changed data available (in sequential files, for example) for processing by another program that you write, the *receiver*.

When you want to apply the updates, the receiver accesses or receives the changed data, and calls the RUP to update the DB2 table.

The selector and receiver are discussed in more detail in "Writing A Selector Program" on page 326 and "Writing A Receiver Program" on page 327.

Figure 88 on page 323 provides an overview of this implementation.

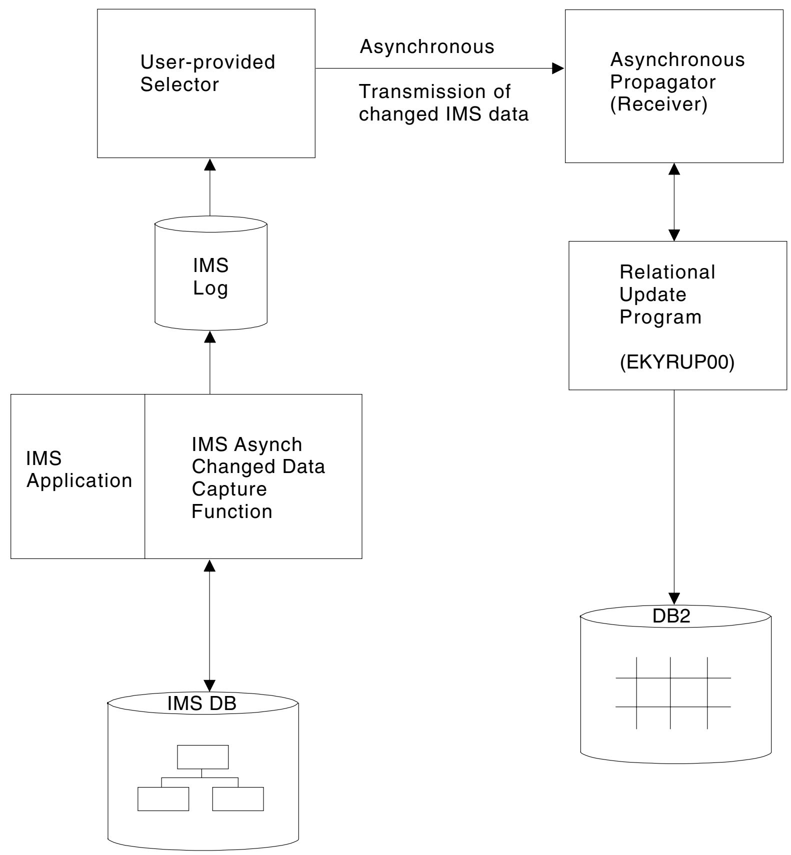

Figure 88. HR Asynchronous Propagation With the IMS Asynchronous Data Capture Function

An implementation based on the IMS Asynchronous Changed Data Capture function supports propagation of updates performed by IMS application programs executing in the following environments:

- IMS Fast Path Regions
- IMS MPP Regions
- IMS BMP Regions
- IMS Batch Regions
- CICS (only when executing with DBCTL)

Implementation Based on User-Written IMS Data Capture Exit

IMS application programs update the IMS databases. The IMS data capture function provides the changed data to your IMS Data Capture exit routine, which is referred to as the *sender program*. Your sender program must either store the IMS updates until you want to apply the updates to the DB2 table, or send them directly to the receiver. When you want to apply the updates, the receiver accesses or receives the changed data, and calls the RUP to update the DB2 table. The sender and receiver are discussed in more detail in "Writing A Selector Program" on page 326 and "Writing A Receiver Program" on page 327.

Figure 89 on page 324 provides an overview of this implementation.

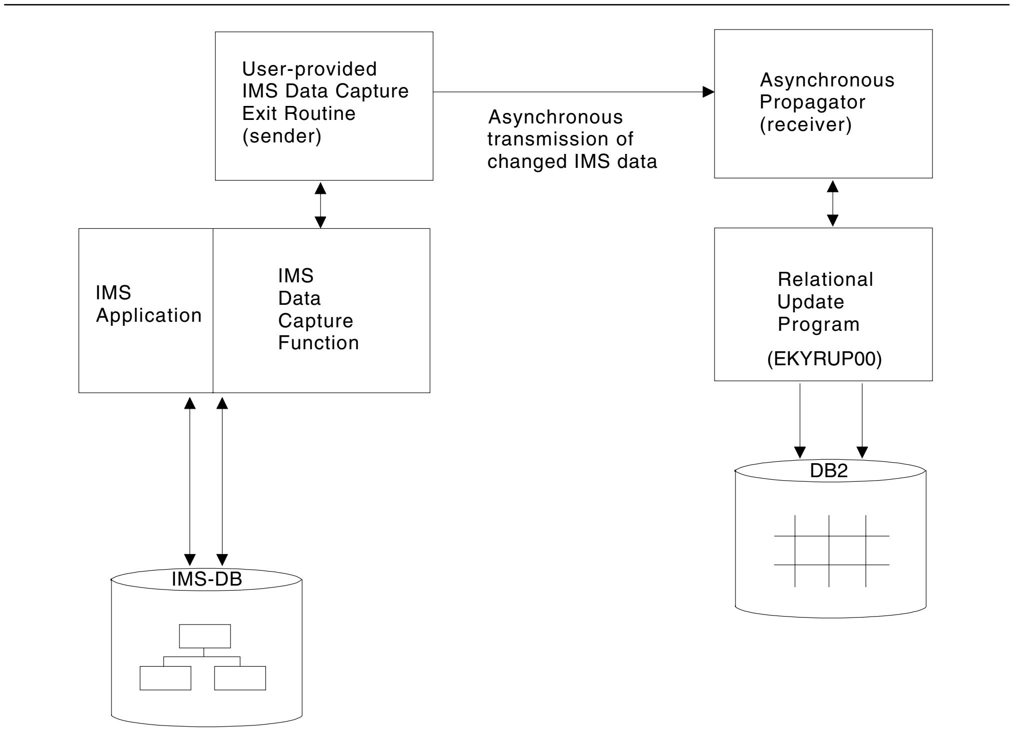

Figure 89. HR Asynchronous Propagation With a User-Written IMS Data Capture Exit Routine

An implementation based on a user-written IMS Data Capture exit routine supports propagation of updates performed by IMS application programs executing in the following environments:

- IMS Fast Path Regions
- IMS MPP Regions
- IMS BMP Regions
- IMS Batch Regions

This implementation does not support propagation of updates performed by IMS application programs executing in a CICS environment.

Developing Your Asynchronous System

This section explains how you can develop your asynchronous system.

Setting Up Your Asynchronous System

You must determine the exact processes that the selector, sender, and receiver use to call the RUP asynchronously.

Because the IMS Data Capture function does not call the RUP directly, your programs must provide several processing features that are described in this section. Keep these features in mind while developing your asynchronous system.

Calling the RUP

The fact that data propagation is asynchronous must be invisible to the RUP. That is, your programs must call the RUP in exactly the same way as the IMS Data Capture function during synchronous propagation. Therefore, if you develop asynchronous propagation based on an IMS Data Capture exit routine, your sender must record all the information passed to it by the IMS Data Capture Function. If you develop asynchronous propagation based on the IMS Asynchronous Changed Data Capture Function, your selector must record all the information available in the IMS log records containing changed data. The receiver must call the RUP using this information exactly as the IMS Data Capture function uses it. This is discussed further in “Writing A Receiver Program” on page 327.

Programming languages supported

The RUP can be called from a program written in Assembler, COBOL, PL/I, or C languages. However, this too must be transparent to the RUP. Support of COBOL and PL/I programs assumes that the RUP can function as though it were called by a program written in Assembler.

Handling the Changed Data

While you want your programs to be efficient and provide a reasonable throughput, your programs must ensure that the propagated data changes are presented to the RUP in the correct sequence. Your programs must also avoid losing propagated data, or propagating changes multiple times. These situations cause inconsistencies between the IMS data and DB2 data.

When called for asynchronous propagation, the RUP always propagates IMS inserts, including those made with an IMS processing option load. Your programs must filter out the inserts that you do not want propagated.

Your programs must also provide some operational support; for example, avoid losing changed data in both normal and abnormal situations.

Propagation Failures

With asynchronous data propagation, failures do not automatically trigger a coordinated backout of the IMS update and the DB2 updates. If you encounter a propagation failure, the RUP signals the failure to your calling receiver program. The RUP does not perform a rollback.

Your receiver program must provide the logic to handle any propagation failures that can occur. The receiver must not call the RUP after a propagation failure until the problem is fixed. This can cause many more data inconsistencies and propagation failures.

More information on error handling for the receiver is discussed in “Writing A Receiver Program.” Also, because the RUP can abend, your programs must provide restart logic.

You can also provide trace and audit capabilities for those parts of your system that the DPROP tracing functions do not trace.

Sync Point Processing

Your asynchronous propagation system must perform its own sync point processing. You can begin processing after completing each original unit of work (UOW).

Splitting the IMS Data

The sender and selector store IMS data. To increase efficiency, split this data into parts. Then call multiple copies of the receiver in parallel. Each copy of the receiver is called in a distinct address space to process its portion of the IMS data.

You can split the IMS data many different ways. Examples include splitting the data by DBD name, segment type, key range values of the root segment, and so on.

Writing A Selector Program

The selector gathers the log records containing changed data from the IMS log datasets, and makes the data available for receiver processing.

If multiple IMS subsystems are updating the same databases, the selector needs to merge the IMS log records containing changed data in a sequence consistent with that in which IMS generated them. If your selector and receiver programs maintain the correct sequence, this ensures that your data remains consistent between IMS and DB2.

The selector creates output data sets containing the changed data to be propagated. If the receiver executes on a different remote MVS system, the output data sets can be transmitted with file transfer programs.

Processing log records containing changed data requires a detailed understanding of the format of these log records. Refer to *IMS/ESA Customization Guide* for a detailed description of these log records.

Writing A Sender Program

The sender program is defined to IMS in the DBDGEN as an IMS Data Capture exit routine. The sender stores the propagated data changes and IMS Data Capture interface information, or sends this information directly to the receiver. If the IMS updates are made in an IMS online environment, the sender can continuously send the updates to the receiver or a remote destination by inserting the data into IMS output messages; or, if the sender and receiver are on different MVS systems, the messages can be sent across MSC or ISC links. If you plan to temporarily store the updates before sending them to the receiver, you can store them in the:

- IMS log
- IMS full-function database
- DEDB sequential dependent segments
- MVS flat file

If you store the changed data on the IMS log, use the Remote Recovery Data Facility (RRDF) when you send the data to a remote destination or to the receiver. For more information on RRDF, refer to *RRDF Program Description and Operations*.

Remember to present the updates to the RUP in a sequence consistent with that which IMS created. If your sender and receiver programs maintain the correct sequence, your data remains consistent between IMS and DB2.

See *IMS/ESA Customization Guide* for details on the IMS Data Capture interface that IMS uses to call the sender. Also, see the next section for more details on duplicating the interface to call the RUP.

Writing A Receiver Program

The receiver program receives the changed data information from the sender or selector, and calls the RUP to update the propagated DB2 table. You can write the receiver program in Assembler, COBOL, C, or PL/I. Whatever language you choose, the RUP must run as though it were called from an Assembler program.

Your receiver must provide the necessary JCL to call the RUP. Also, because the RUP accesses the DPROP directory, you must provide a usable DB2 plan. If you are using DPROP for both synchronous and asynchronous data propagation, you must generate two DPROP systems.

The fact that data propagation is asynchronous must be invisible to the RUP. Therefore, your receiver must duplicate the IMS Data Capture interface. This and other requirements for the receiver are discussed in more detail below.

For each job step in which your receiver program calls the RUP, the receiver must generate:

1. One initialization call to the housekeeping module EKYZ800X. This module initializes the DPROP environment.
2. One call to the RUP for each changed data segment. Again, the receiver must provide all the IMS Data Capture interface information.
3. Termination calls to the housekeeping module to complete DPROP activities.

When calling the RUP, it is very important to present the updates to the RUP in a sequence that ensures the consistency of your data between IMS and DB2. To do this, present the updates to the RUP in the same sequence as that in which IMS created them. The receiver must maintain the concept of the original unit of work (UOW) while presenting updates to the RUP.

Interface Used to Call the RUP

Your receiver program must duplicate the IMS Data Capture Interface to call the RUP. The IMS Data Capture interface consists of eight parts; one part is a parameter that is passed to the RUP, and this part contains pointers to the other parts. The parameter and pointers include:

1. A DL/I XPCB control block containing pointers to the next parts
2. One or more DL/I XSDB control blocks
3. The output of a DL/I INQY call, which IMS created when the DL/I data changed
4. The fully concatenated key of the changed DL/I segment
5. The changed DL/I segment (for replace calls, both the before- and after-images)

6. The hierarchical parent and ancestors of the changed segment, if the IMS data capture function provided them
7. The DBD version ID
8. A 256-byte area reserved for the RUP

The RUP is called with only one parameter: the XPCB. It contains both a description of the data change and pointers to the other information listed above. When your receiver program calls the RUP, the receiver program must provide the XPCB parameter. The XPCB must have the same format and content as when used in the IMS Data Capture interface.

Your receiver must also provide the RUP with access to all the other information in the list above. For example, you must provide the same XSDB control blocks that IMS provided to the Data Capture exit routine (the sender) when the segment was changed; all of the XSDB fields must be filled in before calling the RUP. You must provide the same INQY call output.

Observe the following conventions for the 256-byte area pointed to by the XPCB:

- The receiver must initialize this area with binary zeroes before the first call to the RUP, and must not change its content afterward.
- If your receiver program uses more than one XPCB copy, then each XPCB copy must point to the same copy of the 256-byte area.

This interface information is described in detail in *IMS/ESA Customization Guide*. The return and reason codes that the RUP returns to your program are discussed in “Error Handling” on page 337.

To make maintenance and migration activities easier, avoid link-editing your receiver program with the RUP. Instead, the receiver must call the RUP dynamically. If the receiver is written in Assembler:

1. Generate an MVS LOAD macro to load the RUP (EKYRUP00) and save its entry point address.
2. Provide all interface information.
3. Branch to the RUP entry point (using a BASR).

If the receiver is written in COBOL, you can call the RUP dynamically. Use a Call Identifier statement.

If the receiver is written in PL/I, refer to PL/I documentation for the interface used to call an Assembler program.

XPCB and XSDB Interfaces: The XPCB is the only parameter passed by your receiver to the RUP. It is used to provide information about the changed data and to point to XSDBs. An XSDB points to, and describes, either a changed segment occurrence or a physical ancestor of a changed segment.

IMS defines the XPCB and the XSDB control blocks.

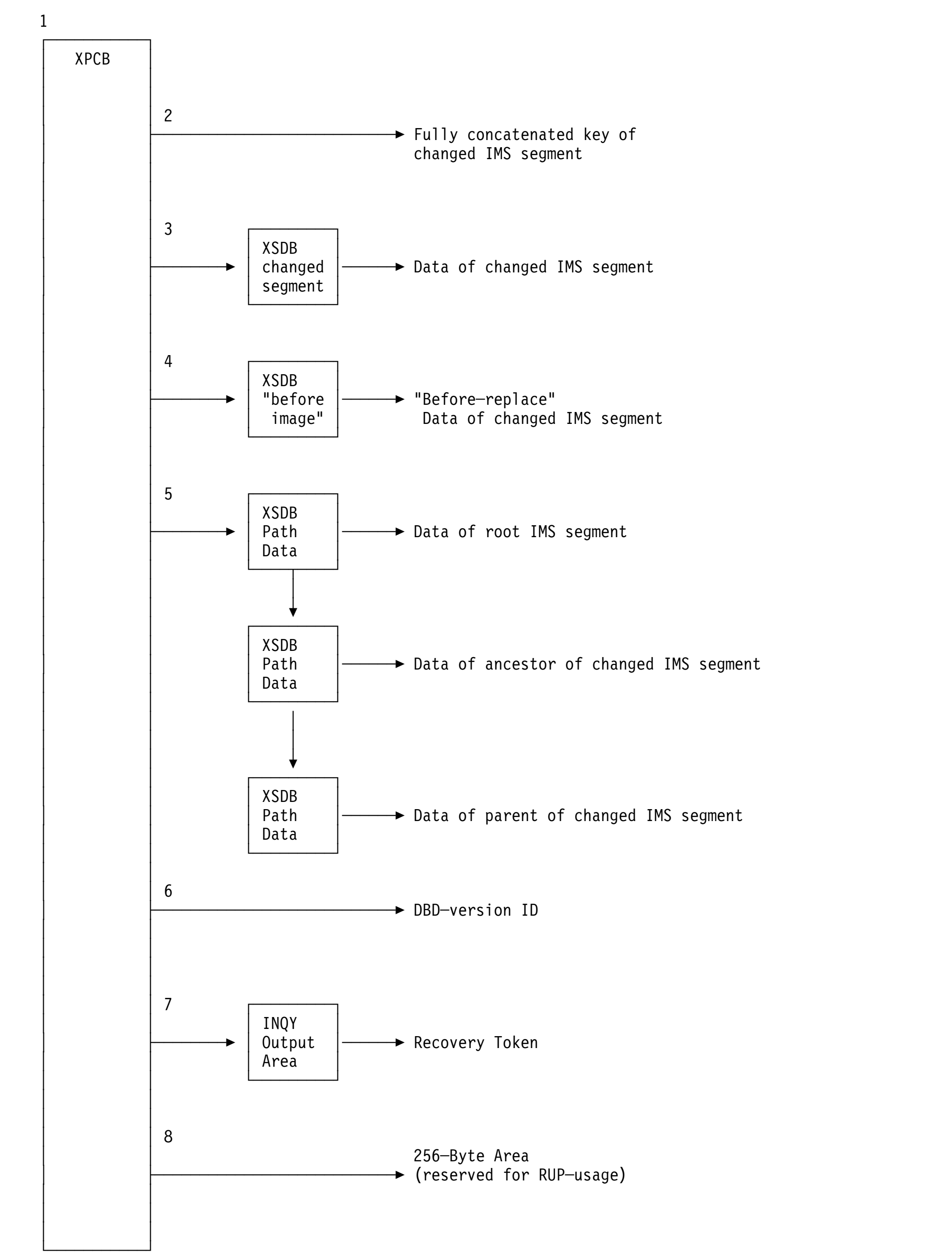

Figure 90. The XPCB and XSDB Control Block Structure

As shown by the numbered sections of the figure, the interface consists of:

1. One XPCB control block, which provides a description of the changed data and contains various pointers.
2. A pointer to the fully concatenated key of the changed segment.
3. A pointer to the XSDB control block describing the changed segment. This XSDB points to the data of the changed segment.
4. For Replace operations, a pointer to an XSDB describing the segment before it was replaced. The XSDB also points to the data of the before image of the segment.

5. A pointer to the first XSDB in a chain of XSDBs for the hierarchical ancestors of the changed segment. The chain is in descending hierarchical order, with each XSDB pointing to the segment data of the segment itself and to the next XSDB in descending order.
6. A pointer to the DBD version ID.
7. A pointer to an area containing the output of an implied IMS INQY ENVIRON call.
8. A pointer to a 256-byte area reserved for RUP-usage.

The XPCB Control Blocks: Figure 91 on page 331 shows the DSECT for the XPCB. In the figure, each field is marked with a note number, which refers to a note (located after the figure) describing how the receiver should set the field.

You can generate the XPCB control block (together with the XSDB and the output area of an IMS INQY call) by coding the EKYRCDL1 macro statement.

```
                                 1          EKYRCDL1
                                 3+*********************************************************************
                                 4++                                                                    *
                                 5++        E X T E N D E D   D A T A  B A S E  P C B  -- X P C B      *
                                 6++                                                                    *
                                 7+*********************************************************************
000000                           9+XPCB      DSECT  ,    NOTE
000000                          10+XPCBEYE   DS    CL4    2.a   "XPCB" EYECATCHER
000004                          11+XPCBVER   DS    CL2    2.b   XPCB VERSION INDICATOR
000006                          12+XPCBREL   DS    CL2    2.c   XPCB RELEASE INDICATOR
000008                          13+XPCBEXIT  DS    CL8    1.a   SEGMENT USER EXIT NAME
000010                          14+XPCBRC    DS    H     11     RETURN-CODE
000012                          15+XPCBRSNC  DS    H     11     REASON-CODE
000014                          16+XPCBDBD   DS    CL8    3     PHYSICAL DATA BASE NAME
00001C                          17+XPCBVERA  DS    A      4     ADDRESS OF DBD VERSION ID
000020                          18+XPCBSEG   DS    CL8    3     PHYSICAL SEGMENT NAME
000028                          19+XPCBCALL  DS    CL4    3     'CALL FUNCTION' DEFINED BY IMS/ESA
                                20++                            ISRT: INSERT
                                21++                            REPL: REPLACE
                                22++                            DLET: DELETE
                                23++                            CASC: CASCADING DELETE
                                24++                            DLLP: NOW ALSO DELETD FROM LOGI.PATH
00002C                          25+XPCBPCALL DS    CL4    3     'PHYSICAL UPDATE TYPE' DEFINED BY IMS
                                26++                            ISRT: INSERT
                                27++                            REIN: RE-INSERT VIA LOGICAL PATH
                                28++                            REPL: REPLACE
                                29++                            DLET: DELETE
                                30++                            DLPP: DELETED ONLY FROM PHYSIC. PATH
000030                          31+          DS    CL4    1.a   RESERVED
000034                          32+XPCBPCBA  DS    A      1.b   ADDRESS OF DB PCB
000038                          33+XPCBPCBN  DS    CL8    1.a   NAME OF DB PCB
000040                          34+XPCBINQA  DS    A      5     ADDRESS OF "INQY" OUTPUT
000044                          35+XPCBIOPA  DS    A      1.b   ADDRESS OF I/O PCB
000048                          36+          DS    H      1.b   RESERVED
00004A                          37+XPCBCKEYL DS    H      3     LENGTH OF FULLY CONCATENATED KEY
00004C                          38+XPCBCKEYA DS    A      6     ADDRESS OF FULLY CONCATENATED KEY
000050                          39+XPCBXSDBD DS    A      7     ADDRESS OF XSDB FOR DATA
000054                          40+XPCBXSDBB DS    A      8     ADDRESS OF XSDB FOR REPL DATA
000058                          41+XPCBXSDBP DS    A      9     ADDRESS OF XSDB FOR PATH DATA
00005C                          42+          DS    F      1.b   RESERVED
000060                          43+          DS    F      1.b   RESERVED
000064                          44+          DS    F      1.b   RESERVED
000068                          45+XPCBEXIWP DS    A     10     ADDRESS OF 256-BYTE AREA RESERVED FOR RUP
00006C                          46+          DS    F      1.b   RESERVED
000070                          47+          DS    F      1.b   RESERVED
000074                          48+XPCBTIMST DS    CL8    3     TIMESTAMP OF CALL
00007C                          49+          DS    F      1.b   RESERVED
                       00080    50+XPCBLEN   EQU   *-XPCB       LENGTH OF XPCB
```

Figure 91. Extended Program Communication Block (XPCB)

Notes:

1. Before calling the RUP, the receiver should set:
   a. Blanks in the XPCB fields
   b. Binary zeroes in the XPCB fields
2. Before calling the RUP, the receiver should initialize the following fields of the XPCB with constants as follows:

   XPCBEYE Should be initialized with the value XPCB
   XPCBVER Should be initialized with the value V1
   XPCBREL Should be initialized with the value R1
3. Before calling the RUP, the receiver should set the XPCB fields identified with 3, to the value provided by IMS, either:
   - In the XPCB, when calling your user-written IMS Data Capture exit routine

- In the CAPD block of the changed data capture IMS log records, if using the IMS Asynchronous Data Capture function

4. ***XPCBVERA*** (pointer to the DBD Version ID):

   Before calling the RUP, the receiver should provide in this field a pointer to a variable-length character string that contains the DBD version. Unless the character string is set from the DBD VERSION= keyword, it will be the time stamp of the DBDGEN. The first two bytes are a halfword containing the length of the string, and are followed by the string itself.

   The DBD Version ID must have the same value provided by IMS either:

   - Via the XPCBVERA pointer, when calling your user-written IMS Data Capture exit routine
   - In the changed data capture IMS log record, if using the IMS Asynchronous Data Capture function

5. **XPCBINQA** (pointer to INQY ENVIRON output area):

   Before calling the RUP, the receiver should provide in this field a pointer to an area that has the same layout as the output area of a INQY ENVIRON DL/I call. See “The INQY ENVIRON output area” on page 335 for more information on the output area.

6. **XPCBCKEYA** (pointer to the fully concatenated key)

   Before calling the RUP, the receiver should provide in this field a pointer to the fully concatenated key of the changed IMS segment.

   The fully concatenated key must have the same value provided by IMS either:

   - Via the XPCBCKEYA pointer, when calling your user-written IMS Data Capture exit routine
   - In the changed data capture IMS log record, if using the IMS Asynchronous Data Capture function

7. **XPCBXSDBD** (pointer to the XSDB describing changed segment):

   Before calling the RUP, the receiver should provide in this field a pointer to the XSDB control block describing the changed IMS segment.

   Your receiver should set this field to zero before calling RUP if IMS does not provide a description of the changed IMS segment:

   - In an XSDB control block, when calling your user-written IMS Data Capture exit routine, or
   - In a CAPD_DATA block in the changed data capture IMS log records

8. **XPCBXSDBB** (pointer to the XSDB describing the “before-image” of the changed segment):

   Before calling the RUP, the receiver should provide in this field a pointer to the XSDB control block describing the before-image of the changed IMS segment.

   Your receiver should set this field to zero before calling RUP if IMS does not provide a description of the before-image of the changed IMS segment either:

   - In an XSDB control block, when calling your user-written IMS Data Capture exit routine, or
   - In a CAPD_DATA block within the changed data capture IMS log records

9. **XPCBXSDBP** (pointer to the first XSDB describing the segments in the hierarchic path, in descending order, above the changed segment):

   Before calling the RUP, the receiver should provide in this field a pointer to the XSDB control block that describes the first segment in the hierarchic path above the changed segment.

   Your receiver should set this field to zero before calling RUP if IMS does not provide a description of the segments in the hierarchic path above the changed segment either:

   - In XSDB Control blocks when calling your user-written IMS Data Capture exit routine, or
   - In CAPD_DATA blocks within the changed data capture IMS log records.

10. **XPCBEXIWP** (pointer to a 256-byte area reserved for RUP):

    Before calling the RUP, the receiver should provide in this field a pointer to a 256-byte area reserved for RUP usage.

    Your receiver should observe the following conventions for the 256-byte that area the XPCB points to:

    - The receiver must initialize this area with binary zeroes before the first call to the RUP.
    - The receiver must not change its content afterward.

    If your receiver uses more than one XPCB copy, then each XPCB copy should point to the same copy of the 256-byte area.

11. **XPCBRC** and **XPCBRSNC** (return code and reason code)

    The RUP returns on call completion a return code and a reason code in these fields. See “Error Handling” on page 337 for a description of the return codes and reason codes.

The XSDB Control blocks: Figure 92 on page 334 shows the DSECT for the XSDB. In the figure, each field is marked with a note number, which refers to a note (located after the figure) describing how the receiver should set the field.

You can generate the XSDB control block (together with the XPCB and the output area of an IMS INQY call) by coding the EKYRCDL1 macro statement.

```
                                   52+*********************************************************************
                                   53+*                                                                   *
                                   54+*        E X T E N D E D   S E G M E N T   D A T A   -- X S D B      *
                                   55+*                                                                   *
                                   56+*********************************************************************
000000                             58+XSDB      DSECT  ,    NOTE
000000                             59+XSDBEYE   DS    CL4    2.a   "XSDB" EYECATCHER
000004                             60+XSDBVER   DS    CL2    2.b   XSDB VERSION INDICATOR
000006                             61+XSDBREL   DS    CL2    2.c   XSDB RELEASE INDICATOR
000008                             62+XSDBNXSDB DS    A      4     NEXT XSDB POINTER
00000C                             63+XSDBDBD   DS    CL8    8     PHYSICAL DATA BASE NAME
000014                             64+XSDBSEG   DS    CL8    3     PHYSICAL SEGMENT NAME
00001C                             65+XSDBPHP   DS    CL1    5     PHYSICAL PATH ACCESSIBILITY
                          000E8    66+XSDBPHPY  EQU   C'Y'         ...SEGM ACCESSIBLE VIA PHYSICAL PATH
                          000D5    67+XSDBPHPN  EQU   C'N'         ...SEGM NOT ACCESSIBLE VIA PH. PATH
00001D                             68+          DS    CL3    1.a   RESERVED
000020                             69+XSDBSEGLV DS    H      3     SEGMENT DATA BASE LEVEL
000022                             70+XSDBKEYL  DS    H      3     LENGTH OF PHYSICAL KEY
000024                             71+XSDBKEYA  DS    A      6     ADDRESS OF PHYSICAL KEY
000028                             72+XSDBFIL1  DS    H      1.b   RESERVED
00002A                             73+XSDBSEGL  DS    H      3     LENGTH OF SEGMENT DATA
00002C                             74+XSDBSEGA  DS    A      7     ADDRESS OF SEGMENT DATA
000030                             75+XSDBFIL2  DS    F      1.b   RESERVED
000034                             76+XSDBFIL3  DS    F      1.b   RESERVED
000038                             77+XSDBFIL4  DS    F      1.b   RESERVED
                          0003C    78+XSDBLEN   EQU   *-XSDB       LENGTH OF XSDB
```

Figure 92. Extended Segment Data block (XSDB)

Notes:

1. Before calling the RUP, the receiver should set:
   a. Blanks in the XSDB fields
   b. Binary zeroes in the XSDB fields
2. Before calling the RUP, the receiver should initialize the following fields of the XSDB with constants as follows:

   XSDBEYE Should be initialized with the value XSDB
   XSDBVER Should be initialized with the value V1
   XSDBREL Should be initialized with the value R1
3. Before calling the RUP, the receiver should set the XSDB fields identified with 3 to the value provided by IMS, either:
   - In the XSDB, when calling your user-written IMS Data Capture exit routine)
   - In the CAPD_DATA block of the changed data capture IMS log records, if using the IMS Asynchronous Data Capture function)
4. **XSDBNXSDB** (pointer to the next XSDB describing the segments in the hierarchic path, in descending order, above the changed segment).

   Before calling the RUP, the receiver should provide in this field a pointer to that XSDB control block that describes the next segment in the hierarchic path above the changed segment.

   Your receiver should set this field to zero before calling the RUP if IMS did not provide a description of the next segment in the hierarchic path above the changed segment either:
   - In an XSDB Control block, when calling your user-written IMS Data Capture exit routine, or
   - In a CAPD_DATA block within the changed data capture IMS log records.

5. **XSDBPHPY** (physical path accessibility):

   Before calling the RUP, the receiver should set this field to N if either:

   - IMS set this field to N when calling your user-written IMS Data Capture exit routine, or
   - The DEL_ON_PHY_PATH flag is set to On in the CAPD_DATA block of the changed data capture IMS log records, if using the IMS Asynchronous Data Capture function).

   In other cases, XSDBPHPY should be set to Y.

6. **XSDBKEYA** (address of physical key)

   Before calling the RUP, the receiver should provide in this field a pointer to the keyfield of the segment described by this XSDB.

   Your receiver should set this field to zero before calling RUP if IMS did not provide a pointer to the keyfield either:

   - In the XSDB Control block when calling your user-written IMS Data Capture exit routine, or
   - In a CAPD_DATA block within the changed data capture IMS log records.

7. **XSDBSEGA** (address of segment data)

   Before calling the RUP, the receiver should provide in this field a pointer to the segment described by this XSDB.

   Your receiver should set this field to zero before calling the RUP if IMS did not provide either one of the following:

   - In the XSDB control block a pointer to the segment data when calling your user-written IMS Data Capture exit routine, or
   - The segment data in the changed data capture IMS log records.

8. **XSDBDBD** (physical database name)

   Before calling the RUP, the receiver should provide in this field the same value as in XPCBDBD.

The INQY ENVIRON output area: Figure 93 on page 336 shows the DSECT for the output area of the INQY output area. In the figure, each field is marked with a note number, which refers to a note (located after the figure) describing how the receiver should set the field.

You can generate the DSECT for the INQY ENVIRON output area, together with the XPCB and the XSDB, by coding the EKYRCDL1 macro statement.

```
                              100+*********************************************************************
                              101+*                                                                   *
                              102+*      INQUIRY (INQY) CALL OUTPUT                                   *
                              111+*                                                                   *
                              112+*********************************************************************

                              114+*-------------------------------------------------------------------*
                              115+*                                                                   *
                              116+*      ----------------------                                       *
                              117+*      SUBFUNCTION = 'ENVIRON'                                      *
                              118+*      ----------------------                                       *
                              119+*                                                                   *
                              120+*-------------------------------------------------------------------*

000000                        122+INQENVRN DSECT ,    NOTE
000000                        123+INQEIMID DS    CL8     3         IMS IDENTIFIER
000008                        124+INQEIMRL DS    F       3         IMS RELEASE LEVEL
00000C                        131+INQECRT  DS    CL8     3         CONTROL REGION TYPE
000014                        140+INQEART  DS    CL8     3         APPLICATION REGION TYPE
00001C                        141+INQEARID DS    F       3         APPLICATION RGN IDENTIFIER
000020                        142+INQEPGM  DS    CL8     3         APPLICATION PROGRAM NAME
000028                        143+INQEPSB  DS    CL8     2         ALLOCATED PSB NAME
000030                        144+INQETRAN DS    CL8     2         TRANSACTION NAME
000038                        145+INQEUSER DS    CL8     2         USER IDENTIFIER
000040                        146+INQEGPNM DS    CL8     3         GROUP NAME
000048                        153+INQESGID DS    CL4     3         HIGHEST STATUS GROUP ID
00004C                        154+INQERTA  DS    A       4         ADDRESS OF RECOVERY TOKEN
000050                        156+INQEAPA  DS    A       1         ADDRESS OF APPLICATION PARM
                      00054   158+INQELEN  EQU   *-INQENVRN        ENVIRON OUTPUT LENGTH
```

Figure 93. INQY ENVIRON Output Area

Notes:

1. Before calling the RUP, the receiver should set binary zeroes in the INQEAPA field.
2. Before calling the RUP, the receiver should set the fields identified with 2 to the value provided by IMS, either:
   - In the INQY ENVIRON output area, when calling your user-written IMS Data Capture exit routine.
   - In the LOG_INQY_PSBNAME, LOG_INQY_TRANNAME, and LOG_INQY_USERID fields of the LOG_DCAP_DATA portion of the changed data capture IMS log records, if using the IMS Asynchronous Data Capture function. These fields can be set to blank, if the Receiver does not find the required information in the IMS log records.
3. Before calling the RUP, the receiver should set the fields identified with 3:
   - To the value provided by IMS in the INQY ENVIRON output area, when calling your user-written IMS Data Capture exit routine.
   - To blanks or binary zeroes, if using the IMS Asynchronous Data Capture function.
4. **INQERTA**

   Before calling the RUP, the receiver should provide in this field a pointer to a variable-length field. The two first bytes of the variable-length field are a halfword containing the length of the field, and are followed by:

   - In the case where the Data Staging Area of DataPropagator Relational is being fed by the RUP:

  - 8 bytes containing binary zeros
  - 8 bytes containing a commit timestamp in TOD format
  - 16 bytes containing the CVT adjustment (CVTLDTO and CVTLSO) which converts the above TOD timestamp to local time
- In the case where DataPropagator Relational is not being used, a recovery token that must have the same value as that provided by IMS, either:
  - Via the INQERTA pointer, when calling your user-written IMS Data Capture exit routine
  - In the RECOVTKN field of the LOG_DCAP_DATA portion of the changed data capture IMS log record, if using the IMS Asynchronous Data Capture function

Error Handling

When it is called for asynchronous data propagation, the RUP recognizes three types of propagation errors. They are failures caused:

- By deadlocks
- By unavailable resources
- For other reasons (typically mapping errors)

Your receiver must handle propagation failures. If not handled correctly, propagation failures can result in data inconsistencies.

The RUP does not perform rollbacks when it encounters an error. Therefore, design your receiver to generate rollbacks that preserve the concept of the original UOW. This can help you to maintain the correct sequence of updates presented to the RUP.

When the RUP returns a return code of 8 (indicating an error), it writes error messages to the //EKYPRINT data set, the DPROP audit trail, and the optional trace data set //EKYTRACE to help with diagnosis.

Return codes and reason codes: The RUP places the following return (RC) and reason (RSNC) codes in the XPCB when a propagation error occurs:

RC=0, RSNC=0
Propagation completed successfully. For PRs defined with ERROPT=IGNORE, the RUP can return RC=0, RSNC=0 even if propagation failed.

RC=0, RSNC=4
This is a warning. The RUP completed propagation without errors. However, the number of successfully processed PRs is zero. This can occur, for example, if the DPROP directory did not contain a PR defined for the segment type of the changed data.

RC=8, RSNC=4
Propagation failed because of a DB2 deadlock. DB2 performed a rollback for the entire UOW. The receiver can restart processing of the failed UOW.

RC=8, RSNC=8
Propagation failed because of a DB2 deadlock. However, rollback processing for the failing UOW was not performed. The receiver can generate an SQL rollback and restart processing of the failed UOW. This combination of codes is never returned if you are running under IMS.

RC=8, RSNC=12
Propagation failed due to an unavailable resource problem. Rollback processing for the failing UOW was *not* performed. The receiver must generate a rollback and terminate, or Abend. To maintain data consistency, you must solve the unavailable resource problem before restarting processing of the failed UOW.

RC=8, RSNC=16
Propagation failed because of another type of error. Rollback processing for the failing UOW was not performed.

For PRs defined with ERROPT=BACKOUT, the receiver must generate a rollback and terminate, or Abend. To maintain data consistency, you must solve the error before restarting processing of the failed UOW.

For PRs defined with ERROPT=IGNORE, the RUP rarely returns with RC=8, RSNC=16. Instead, it often provides diagnosis information and returns to the receiver without error indications. The RUP still generates error messages to the //EKYPRINT data set and the optional trace data set. The RUP can also write error messages to the audit trail and snaps to the trace data set. The amount of these messages to the audit trail and snaps to the trace data set can be controlled with the MAXERROR value during PR generation.

After writing error messages, the RUP processes any remaining PRs for the same segment type and returns to the receiver with zero return and reason codes.

Calling the Housekeeping Module EKYZ800X

The first call to DPROP within an address space must be an initialization call to the DPROP housekeeping module (EKYZ800X). This call tells DPROP what environment it is in and that it is being called asynchronously.

The last call to DPROP within an address space must be a termination call to the housekeeping module. This tells DPROP to perform its cleanup processing (for example, to close files).

Avoid generating initialization and termination calls frequently because of their significant performance impact.

The housekeeping module must be called according to the standard OS/VS conventions for calling Assembler modules.

Parameters for the initialization Call: For DPROP initialization, call the module with the following two parameters:

Call Function A four-byte character field that contains the string **INIT**

Environment A four-byte character field that describes the environment. DPROP uses this value to determine which language interface to use for SQL calls. The value of the string must be one of the following:

- **IMS** The receiver is running in an IMS environment. DPROP generates its SQL statements through the language interface of the IMS Attach facility.
- **TSO** The receiver is running in a TSO environment. DPROP generates its SQL statements through the language interface of the TSO Attach facility.

CAF The receiver is running in a Call Attach facility (CAF) environment. DPROP generates its SQL statements through the language interface of the CAF Attach.

If running in a CAF environment, the receiver must establish a CAF connection to DB2 before calling the housekeeping module, and close the connection after terminating the housekeeping module. The receiver can establish the connection with CAF CONNECT and OPEN requests; it can close the connection with CAF CLOSE and DISCONNECT requests.

The RUP and the housekeeping module must be called from the task that establishes and closes the CAF connection.

Remember that you must link-edit any DPROP exit routine that generates SQL statements with the SQL language interface of the proper DB2 Attach.

Parameter for the termination Call: For DPROP termination, only the first parameter is required:

Call Function A four-byte character field containing the string **TERM**.

Environment This parameter is optional. The housekeeping module does not use it, but it can be useful for consistency with the initialization call.

Calling the Module: As with the RUP, the receiver must not be link-edited with the housekeeping module. Instead, you must call the module dynamically using the same methods described above.

Return codes: This section describes the return codes from the housekeeping module. After a termination call, the module (EKYZ800X) always returns with a zero in Register 15 (R15).

After an initialization call, the following codes can be returned in R15:

0 DPROP initialization was successful.

4 DPROP initialization failed because a DB2 deadlock condition was encountered. Rollback processing for the failing UOW was performed. The receiver can restart any processing done during the failing UOW, and regenerate the INIT call.

8 DPROP initialization failed because a DB2 deadlock condition was encountered. However, rollback processing was not performed. The receiver can generate an SQL rollback, and restart any processing done during the failing UOW.

12 DPROP initialization failed because of an unavailable resource. Rollback processing for the failing UOW was not performed. The receiver must either generate an SQL rollback and return (if running under TSO/CAF), or Abend.

16 DPROP initialization failed for another type of error. The receiver must either generate an SQL rollback and return (if running under TSO/CAF), or Abend.

Supported Environments and Restrictions

This section describes the environment DPROP supports when your receiver calls DPROP:

- The RUP runs as a DB2 application; your receiver can call it in the following environments:
  - In an IMS/ESA batch or BMP region
  - under TSO foreground or TSO batch
  - In a DB2 CAF environment

  IMS MPP or IFP regions, and CICS environments are not supported. DPROP does not test if it is being called in one of these environments.
- If your receiver is running in a TSO or CAF environment, any DPROP exit routines must not generate DL/I calls, because IMS does not support TSO or CAF.
- DPROP does not support access to remote or distributed DB2 systems. This applies to both synchronous and asynchronous data propagation systems.
- The RUP and the housekeeping module (EKYZ800X) must be link-edited in AMODE 31, and must be called in AMODE 31. Remember that, for maintenance and migration considerations, you must not link-edit your receiver with these modules.
- DPROP must be called by a program running in the ordinary problem-program mode, with PSW protection key 8. DPROP cannot be called in:
  - SVC or SRB mode
  - Cross memory mode
  - Access register mode
  - Authorized mode
  - Any protection key other than 8
- If DPROP is running in a subtask, the attaching mother task must not share any OS/VS virtual storage subpool with the subtask, if it intends to reattach DPROP after termination. Also, in a multiple task environment, higher level OS/VS tasks must not preload DPROP modules.

JCL Requirements

To run the receiver, in a TSO or IMS environment, you must provide:

- The JCL that DB2 and DB2 Attach require (the IMS, TSO, or CAF Attach), and
- The JCL that DPROP requires.

This section describes DPROP's JCL requirements. In addition to the DPROP JCL for //STEPLIB and //EKYRESLB, this includes:

1. An //EKYPRINT DD statement, which the RUP uses to write error messages. It is typically coded as:

   ```
   //EKYPRINT DD SYSOUT=A
   ```

2. An //EKYLOG DD statement or an //EKYTRACE DD statement, which contain information that the RUP usually writes to the IMS log (for example, traces, snaps, or error messages). They are typically coded as:

   ```
   //EKYLOG DD DSN=xxxx,DISP=(,CATLG),
   //                  UNIT=xxx,SPACE=(CYL,(nn,nn))
   //EKYTRACE DD SYSOUT=A
   ```

3. An optional //EKYIN DD statement, which is used to provide a TRACE control statement used to activate the DPROP Trace module. Refer to *IMS DPROP Reference* for the syntax of the TRACE statement.

Binding a DB2 Plan for the Receiver

When your receiver calls DPROP, DPROP generates SQL statements to access DPROP directory tables and to update the propagated tables. Therefore, you need to bind a DB2 plan for running the receiver.

For details on how to perform the bind, see the appropriate *Administrators Guide* for your propagation mode.

Installation Considerations: Asynchronous Data Propagation

If you are using DPROP for both synchronous and asynchronous propagation, you must define two different DPROP systems, each with its own DPROP directory.

When installing each DPROP system, you must define whether the system is used for synchronous or asynchronous data propagation. Specify the type of system in the System Type field of the EKYGP42E installation panel:

SYNC For synchronous propagation
ASYNC For asynchronous propagation

Specifying the type of system is part of the DPROP generation and customization process. Refer to *IMS DPROP Installation Guide* for more details.

The Status Change Utility (SCU)

The Status Change utility supports only the following control statements for an asynchronous DPROP system:

- INIT DPROP=
- DISPLAY STATUS

Because you cannot activate or deactivate PRs in an asynchronous DPROP system, the RUP considers all PRs in such a system as active.

Multiple MVS Images

As explained in the appropriate *Administrators Guide* for your propagation mode, additional restrictions apply if IMS and DB2 reside on different MVS images.

DPROP must execute on the *target* MVS system, the same MVS system that owns the DB2 tables. The DPROP directory must also reside on this system.

MVG and MVGU must run on the target MVS system. They must have access to the DBDLIB of the *source* MVS system, the system on which IMS runs. If the MVS images share DASD, the DBDLIB can reside on the shared storage. Otherwise, an up-to-date copy of the DBDLIB must be provided on the target MVS system.

As described in the appropriate *Administrators Guide* for your propagation mode and *IMS DPROP Reference*, special considerations are required for:

- Use of the CCU

- Combined use of DXT for data extract and DPROP for asynchronous propagation

Database Maintenance

With asynchronous propagation, you can reorganize or repair the DB2 side while you are collecting or storing updates. Then, when you are applying those updates to DB2, you can stop updating the IMS database and perform database maintenance activities on the IMS side. Conversely, when you are collecting updates on the IMS side, you can perform database table space maintenance on the DB2 side.

With asynchronous propagation, out-of-sync conditions are normal, and data can be truly synchronized for only brief periods. Asynchronous propagation gives you some allowance for unavailable resources that is not possible in the synchronous environment. You can generally operate on the IMS and DB2 components independently without having one affect the other.

With asynchronous propagation in IMS online or BMP environments, your stored updates must be backed out with a failing IMS UOW if you use a database (full-function or DEDB) or a message queue for warehousing the propagating updates. The IMS synchronization point manager performs this backout. If you use the IMS Batch Backout utility or dynamic backout in IMS batch jobs, your propagating updates must also be backed out from databases used to store changes intended for the receiving program.

However, if you use the IMS log or an MVS flat file for storing propagating updates, then no backout can occur if failure occurs. You must restore data integrity by eliminating the failed updates from the log or flat file.

Recovering the DPROP Directory

For information on recovering the DPROP directory, see the appropriate *Administrators Guide* for your propagation mode.

Appendix A. Calling the Trace Module

This appendix describes the interface for the optional DPROP trace module from a Propagation exit routine. Some reasons for calling the trace module are also discussed. For more general information about DPROP trace support, refer to *IMS DPROP Diagnosis*.

To complement the RUP's and HUP's trace support, your Propagation exit routine can also call the trace module (module EKYR410X) directly. By activating the DPROP Trace with the appropriate debug level, you can request that the RUP and HUP trace the parameters, control blocks, and other areas involved in calling your exit routine. (For information about debug levels see the *IMS DPROP Diagnosis*.) The RUP and HUP can also trace this information when your exit routine signals a propagation failure by returning a nonzero return code. Therefore, your exit routine does not need to provide the code for tracing this interface information.

Typically, your exit routine can call the trace module for two purposes:

1. To trace updating SQL calls (HR propagation) and IMS calls (RH propagation) that your exit routine creates upon request. If the PICDBLV2 bit is on in the Propagation Interface Control Block (PICB), you are requesting the tracing of SQL calls and IMS Calls.
2. To trace information needed for problem determination. If you have a propagation failure, even if you have not requested tracing, you can snap or trace whatever information you think is needed to solve the problem. When your exit routine returns with an error return code, the RUP snaps or traces all relevant interface information.

The DPROP trace module can trace multiple items with each call. For example, when tracing an SQL call, each DB2 column involved in the call is traced as a separate item. the trace module writes its results to the //EKYTRACE data set, to the //EKYLOG data set, or to the IMS log. To find out how to format and print these records and interpret the trace output, refer to *IMS DPROP Diagnosis*.

Trace Module Interface

Your exit routine must use standard OS/VS linkage conventions when calling the trace module.

Register 1	Points to the parameter list described below
Register 13	Contains the address of the standard save area
Register 14	Contains the return address
Register 15	Contains the entry address of the DPROP trace module

Parameter list

The first parameter in the parameter list pointed to by Register 1 must be the address of the Trace Request Block (TRB). Following this address, the parameter list must include the address of one Trace Element Descriptor (TED) for each item included in the trace. The TRB and TED are described below.

The trace module must be called in AMODE 31, and returns control to your exit routine in AMODE 31.

The sample Propagation exit routine (see Figure 52 on page 190) contains a macro called SETTED. This macro simplifies calling the trace module. You can create a similar macro to use in your system. See Figure 52 on page 190, where the SETTED macro is used in the sample Propagation exit routine.

Trace Request Block (TRB)

Figure 94 on page 345 contains the DSECT for the TRB. Following the DSECT, the fields are described in detail.

The EKYTRB DSECT is provided in the DPROP macro library. Code the EKYTRB macro statement to create the DSECT in your exit routine.

```
                                  1         EKYTRB
                                  2+****************** START OF CONTROL BLOCK SPECIFICATION ***************
                                  3+*                                                                     *
                                  4+*       CONTROL BLOCK NAME:                                           *
                                  5+*         EKYTRB (TRB)                                                *
                                  6+*                                                                     *
                                  7+*       DESCRIPTIVE NAME:                                             *
                                  8+*         DPROP TRACE REQUEST BLOCK (TRB)                             *
                                  9+*               =     =       =                                       *
                                 10+*                                                                     *
                                 11+***********************************************************************
                                 12+*                                                                     *
                                 13+*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
                                 14+*                                                                     *
                                 15+*          5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
                                 16+*          ALL RIGHTS RESERVED.                                       *
                                 17+*                                                                     *
                                 18+*          U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
                                 19+*          USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
                                 20+*          GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
                                 21+*                                                                     *
                                 22+*          LICENSED MATERIALS - PROPERTY OF IBM.                      *
                                 23+*                                                                     *
                                 24+***********************************************************************
                                 25+*                                                                     *
                                 26+*       STATUS: V1 R2 M0                                              *
                                 27+*                                                                     *
                                 28+*       FUNCTION:                                                     *
                                 29+*         A TRB IS USED FOR THE COMMUNICATION BETWEEN A               *
                                 30+*         'PROPAGATION USER EXIT ROUTINE' AND THE DPROP TRACE         *
                                 31+*         FUNCTION.                                                   *
                                 32+*                                                                     *
                                 33+*         WHEN INVOKING THE DPROP-TRACE FUNCTION, THE CALLING         *
                                 34+*         USER EXIT MUST PROVIDE THE TRB AS FIRST CALL-PARAMETER.     *
                                 35+*                                                                     *
                                 36+*         THE TRB PROVIDES INFORMATION ABOUT THE TRACE REQUEST.       *
                                 37+*                                                                     *
                                 38+*       MODULE TYPE= MACRO                                            *
                                 39+*          PROCESSOR= ASSEMBLER H                                     *
                                 40+*                                                                     *
                                 41+*       ACQUIRED BY MODULE INVOKING THE TRACE                         *
                                 42+*                                                                     *
                                 43+*       INNER CONTROL BLOCKS: NONE                                    *
                                 44+*                                                                     *
                                 45+*       MACROS USED FROM MACRO LIBRARY: NONE                          *
                                 46+*                                                                     *
                                 47+*       CHANGE ACTIVITY:                                              *
                                 48+*                      KMP0057  12/13/90                              *
                                 49+*                                                                     *
                                 50+******************** END OF CONTROL BLOCK SPECIFICATION ***************


000000                           54+TRB      DSECT
000000 E3D9C240                  55+TRBEYE   DC   C'TRB '       EYE-CATCHER
000004 00000000                  56+TRBPTD   DC   A(0)          ADDRESS OF THE DPROP-PTD CONTROL BLOCK

                                 58+*********
                                 59+*********                   NAME OF OBJECTS ASSOCIATED WITH THE TRACE
                                 60+*********

000008 4040404040404040          62+TRBTABQ  DC    CL8' '       TABLE-NAME QUALIFIER ASSOC. W. TRACE
000010 4040404040404040          63+TRBTABN  DC    CL18' '      TABLE-NAME ASSOCIATED WITH THE TRACE
000022 4040                      64+         DC    CL2' '
000024 4040404040404040          65+TRBDBN   DC    CL8' '       DBD-NAME ASSOCIATED WITH THE TRACE
00002C 4040404040404040          66+TRBSEGN  DC    CL8' '       SEG-NAME ASSOCIATED WITH THE TRACE
```

Figure 94 (Part 1 of 2). Trace Request Block

```
                                68+*********
                                69+*********                     SOLICITED/UNSOLICITED INDICATION
                                70+*********
000034 40                       72+TRBSOLI  DC   CL1' '          SOLICITED TRACE
                      000E8     73+TRBSOLY  EQU  C'Y'            ...Y: TRACE SOLICITED BY THE USER
                      000D5     74+TRBSOLN  EQU  C'N'            ...N: TRACE NOT SOLICITED BY THE USER

000035 0000000000000000         76+         DC   13X'00'         RESERVED/MUST BE ZERO
                      00042     77+TRBEND   EQU  *
                      00042     78+TRBLEN   EQU  *-TRB           LENGTH OF ONE TRB
                                79          END
```

Figure 94 (Part 2 of 2). Trace Request Block

TRB Field Descriptions

TRBEYE Your exit routine must set this field to **TRB**. The trace module validates its content.

TRBPTD Your exit routine must provide the address of the PTD control block in this field. This PTD address can be found in the PICPTD field of the PICB.

When performing HR propagation, your exit routine must also set the next two fields, which are used in the trace records to identify data objects associated with the trace. DPROP includes the data you provide below, in both the trace record (to allow selective trace formatting), and the formatted trace output.

TRBTABQ The table name qualifier of the table involved in the trace
TRBTABN The unqualified table name of the table involved in the trace

When performing RH propagation, your exit routine must set the next two fields, which are used in the trace records to identify data objects associated with the trace. DPROP includes the data you provide, in both the trace record (to allow selective trace formatting), and the formatted trace output.

TRBDBN The name of the physical IMS database involved in the trace
TRBSEGN The name of the physical IMS segment involved in the trace

Your exit must also set the next field:

TRBSOLI The Propagation exit routine must set this field to determine if the trace was requested by the user. If the user requested it, the exit routine must set this field to **Y**. If the user did not request it, (for example, if errors occurred), the exit routine must set this field to **N**.

Trace Element Descriptor (TED)

This section describes the Trace Element Descriptor (TED). You specify one TED in the keyword list for *each* item you want to trace.

DPROP distinguishes between the following three different types of items that can be traced:

- Header items
- Subheader items
- Data items

The DPROP Trace module formats each of the three types of items differently. In each TED in the parameter list, your exit routine must identify the type of item the TED describes.

DPROP requires that the first TED in the keyword list describe a header item. TEDs describing subheader items are optional; they can be provided to make reading of the formatted trace easier by helping to structure the information presented in the trace output. An exit routine provides one or more TEDs that describe data items to be traced.

Figure 95 is an example of a formatted trace. The figure and the explanations that follow show how the DPROP trace module formats the different item types.

```
*** 15:14:49.88 90.052  PROPAGATING SQL-UPDATE CALL FOR TABLE=PROD.PARTS
    DPR ID   = T096606    IMS ID   = KOEX       USER ID  =
    JOB NAME = T096606X   PSB NAME = KOEPSB3    RECOV TK = 0000000300000005
    DBD NAME = DB1        SEG NAME = SEG1       TAB NAME = PROD.PARTS
    RUP CALL = 00000020   PR ID    = PR0001

.                       SQLCODE:
02F8C94C  00000000                                                    *                                *

...                     COLUMNS IN WHERE CLAUSE:

.                       BRANCH-OFFICE     :
02F95806  F3F3                                                        *33                              *

.                       PART-NBR          :
02F95816  F8F8F4F5 F6F7                                               *884567                          *

...                     PROPAGATED COLUMNS:

.                       MANUFACTURER      :
02F95832  C9C2D440 40404040 40404040 40404040  404040                 *IBM                             *

.                       ZIP-CODE          :
02F95822  C3C140F9 F5F0F3F0                                           *CA 95030                        *

.                       CITY              :
02F95812  E2C1D540 D1D6E2C5 40404040 40404040                         *SAN JOSE                        *

.                       PRICE             :
02F95912  12344567 8F                                                 *                                *
```

Figure 95. Example of Formatted Trace

This is an example of formatted trace. The DPROP Trace module creates it, using the following TEDs:

1. A TED for a header item, which provides a text string that is printed exactly as entered (the text string “PROPAGATING SQL-UPDATE FOR TABLE=PROD.PARTS” at the top of the figure).

   DPROP Trace formatting prefixes the text string of a header item with asterisks, the time, and the date.

   For header items, the DPROP formatting routine prints additional lines with identifying information (DPR ID, IMS ID, and so forth.)

2. A TED for a data item. The second TED consists of the text string “SQLCODE:,” followed on the next print line by the snapped SQL error code.

   Note the difference between TEDs for header items and TEDs for data items:

   - TEDs for header items (and subheader items) provide only a text string.

- TEDs for data items provide both:

  a. A descriptive text string (in the example: “SQLCODE:”) printed on the first print line. DPROP formatting prefixes the text string with a period and some blanks to help identify it.

  b. A virtual storage area to be snapped both in hexadecimal and character/EBCDIC format printed on the following lines. DPROP Trace formatting prefixes each print line with the virtual storage address of the first byte represented in the print line.

3. A TED for a subheader item, consisting of the text string “COLUMNS IN WHERE CLAUSE”

   In this example, the DPROP trace module's caller provides a subheader item to add additional structure to the formatted trace. It identifies the columns used in the WHERE clause of the SQL statement, and columns that the SQL statement propagates.

   DPROP Trace formatting prefixes the text string in a subheader item with three dots and some blanks for easier identification.

4. A TED for a data item (the BRANCH OFFICE).
5. A TED for a data item (the PART NBR).
6. A TED for a subheader item (PROPAGATED COLUMNS).
7. A TED for a data item (the MANUFACTURER).
8. A TED for a data item (the ZIP CODE).
9. A TED for a data item (the CITY).
10. A TED for a data item (the PRICE).

Figure 96 on page 349 shows the DSECT for the Trace Element Descriptors. Field descriptions follow the figure.

The EKYTED DSECT is provided in the DPROP macro library. Code the **EKYTED** macro statement to create the DSECT in your exit routine.

```
                                      1          EKYTED
                                      2+****************** START OF CONTROL BLOCK SPECIFICATION ***************
                                      3+*                                                                     *
                                      4+*       CONTROL BLOCK NAME:                                           *
                                      5+*         EKYTED (TED)                                                *
                                      6+*                                                                     *
                                      7+*       DESCRIPTIVE NAME:                                             *
                                      8+*         DPROP TRACE ELEMENT DESCRIPTOR (TED)                        *
                                      9+*               =     =       =                                       *
                                     10+*                                                                     *
                                     11+***********************************************************************
                                     12+*                                                                     *
                                     13+*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
                                     14+*                                                                     *
                                     15+*          5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
                                     16+*          ALL RIGHTS RESERVED.                                       *
                                     17+*                                                                     *
                                     18+*          U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
                                     19+*          USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
                                     20+*          GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
                                     21+*                                                                     *
                                     22+*          LICENSED MATERIALS - PROPERTY OF IBM.                      *
                                     23+*                                                                     *
                                     24+***********************************************************************
                                     25+*                                                                     *
                                     26+*       STATUS: V1 R2 M0                                              *
                                     27+*                                                                     *
                                     28+*       FUNCTION:                                                     *
                                     29+*         WHEN INVOKING THE DPROP TRACE FUNCTION, THE CALLING         *
                                     30+*         MODULE MUST PROVIDE ONE TED FOR EACH:                       *
                                     31+*          - TRACE-HEADER                                             *
                                     32+*          - TRACE-SUBHEADER                                          *
                                     33+*          - DATA-AREA                                                *
                                     34+*       WHICH SHOULD BE TRACED/SNAPPED.                               *
                                     35+*                                                                     *
                                     36+*       MODULE TYPE= MACRO                                            *
                                     37+*          PROCESSOR= ASSEMBLER H                                     *
                                     38+*                                                                     *
                                     39+*       ACQUIRED BY MODULE INVOKING THE TRACE                         *
                                     40+*                                                                     *
                                     41+*       INNER CONTROL BLOCKS: NONE                                    *
                                     42+*                                                                     *
                                     43+*       MACROS USED FROM MACRO LIBRARY: NONE                          *
                                     44+*                                                                     *
                                     45+*       CHANGE ACTIVITY:                                              *
                                     46+*                      KMP0057  12/13/90                              *
                                     47+*                                                                     *
                                     48+******************** END OF CONTROL BLOCK SPECIFICATION ***************


000000                               52+TED      DSECT
000000 E3C5C440                      53+TEDEYE   DC   C'TED '       EYE-CATCHER
000004 40                            54+TEDTYPE  DC   C' '          TYPE OF TRACE ITEM
                            000C8    55+TEDTYPH  EQU  C'H'          ... HEADER
                            000E2    56+TEDTYPS  EQU  C'S'          ... SUB-HEADER
                            000C4    57+TEDTYPD  EQU  C'D'          ... DATA
000005 40                            58+TEDALIGN DC   C' '          ALIGNMENT FOR SNAP-FORMATTING
                            000D3    59+TEDALIGL EQU  C'L'          ...L    = LEFT ALIGNMENT
                            00040    60+TEDALIGB EQU  C' '          ...BLANK= NO LEFT ALIGNMENT
000006 0000                          61+         DC   XL2'00'       RESERVED
000008 00000000                      62+TEDTXTA  DC   A(0)          PTR TO TEXT-STRING
00000C 00000000                      63+TEDTXTL  DC   F'0'          LENGTH OF TEXT-STRING
000010 00000000                      64+TEDMA    DC   A(0)          VIRTUAL STORAGE ADDR OF AREA TO BE SNAPPED
000014 00000000                      65+TEDALEN  DC   F'0'          LENGTH OF AREA TO BE SNAPPED
000018 00000000                      66+TEDALET  DC   F'0'          ALET OF DATA (MUST BE ZERO IN THIS RELEASE)
00001C 0000000000000000              67+         DC   2F'0'         RESERVED/MUST BE ZERO
```

Figure 96 (Part 1 of 2). Trace Element Descriptor

```
00024    68+TEDEND   EQU  *
00024    69+TEDLEN   EQU  *-TED        LENGTH OF ONE TED
         70          END
```

Figure 96 (Part 2 of 2). Trace Element Descriptor

TED Field Descriptions

TEDEYE
Your exit routine must set this field to **TED**. The trace module validates its content.

TEDTYPE
The type of the item to be traced. DPROP recognizes three types of items:

Header
For a header, your exit routine must set this field to **H**. You must also provide a text string to be used as the header, and store its address in TEDTXTA and its length in TEDTXTL. For a header item, the fields TEDMA, TEDALEN, and TEDALIGN do not apply. Therefore, you do not need to provide values for these fields.

Subheader
For a subheader, your exit routine must set this field to **S**. You must also provide a text string to be used as the subheader, and store its address in TEDTXTA and its length in TEDTXTL. For a subheader item, the fields TEDMA, TEDALEN, and TEDALIGN do not apply. Therefore, you do not need to provide values for these fields.

Data
For data, your exit routine must set this field to **D**. It must store the address of the data item to be traced in TEDMA, and the length in TEDALEN. Depending on the length of the data item, the trace is formatted on one or more print lines.

Your exit routine must also provide a descriptive text string explaining what information is being traced. This text string is printed in the formatted trace output. The address of the text string must be placed in TEDTXTA, and the length in TEDTXTL. When tracing a DB2 column, It is recommended that the text string be the DB2 column name.

Also for a data item, your exit routine must set TEDALIGN to indicate whether the traced area must be left-aligned on the formatted print line.

For more information on headers, subheaders, and data items, and on the format of the trace output, see *IMS DPROP Diagnosis*.

TEDALIGN
This field determines if the first byte of the formatted trace output is aligned to the left of the page. If you want the output to be left-aligned, set this field to **L**. If you do not want the output left-aligned, the field must be blank.

Left-alignment can make the trace output much easier to read, especially when the output length is small and you do not need to locate the area using virtual storage address. DPROP uses left-alignment when tracing SQL calls, and it is recommended that your exit routine use the same convention.

If, however, the traced area is large, or you want to locate traced information using a virtual storage address, do not align the trace output on the left. The output then resembles a storage dump. This can be useful when tracing entire control blocks or work areas. It simplifies location of information when you search using virtual addresses.

To see an example of formatting with left-alignment, refer to Figure 95 on page 347. To see an example of formatting without left-alignment, see *IMS DPROP Diagnosis*.

TEDTXTA The address of the text string that is printed in the formatted trace output.

TEDTXTL The length of the text string that is printed in the formatted trace output.

TEDMA For a data item, the address of the area in storage that is traced.

TEDALEN For a data item, the length of the area in storage that is traced.

Appendix B. Sample Segment Exit Control Blocks

This appendix contains sample Segment exit control blocks which map the existing DPROP interface control blocks. This appendix provides the exit control blocks in three languages:

- COBOL
- PL/I
- C

The Assembler version of the Segment exit control block is shown in Figure 7 on page 28.

Sample Segment Exit Control Block for COBOL

Figure 97 shows an example of the EKYRCDAX control block in COBOL. This control block, called EKYRCDXC, resides in the DPROP Sample Source library (EKYSAMP).

```
000100**************** START OF CONTROL BLOCK SPECIFICATION *********** 00010000
000200*                                                                * 00020000
000300*  CONTROL BLOCK NAME:                                           * 00030000
000400*    EKYRCDXC (DAX)                                              * 00040000
000500*                                                                * 00050000
000600*  DESCRIPTIVE NAME:                                             * 00060000
000700*    DPROP COBOL SEGMENT EXIT INTERFACE BLOCK                    * 00070000
000800*                                                                * 00080000
000900*    COBOL VERSION OF EKYRCDAX                                   * 00090000
001000*                                                                * 00100000
001100****************************************************************** 00110000
001200*                                                                * 00120000
001300*    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".        * 00130000
001400*                                                                * 00140000
001500*    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                      * 00150000
001600*    ALL RIGHTS RESERVED.                                        * 00160000
001700*                                                                * 00170000
001800*    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                   * 00180000
001900*    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY               * 00190000
002000*    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                    * 00200000
002100*                                                                * 00210000
002200*    LICENSED MATERIALS - PROPERTY OF IBM.                       * 00220000
002300*                                                                * 00230000
002400****************************************************************** 00240000
002500*                                                                * 00250000
002600*  STATUS: V1 R2 M0                                              * 00260000
002700*                                                                * 00270000
002800*  FUNCTION:                                                     * 00280000
002900*    THIS IS THE COBOL CONTROL BLOCK USED TO INTERFACE BETWEEN  * 00290000
003000*         - DPROP OR DXT                                         * 00300000
003100*         AND                                                    * 00310000
003200*         - A USER'S SEGMENT EXIT ROUTINE (THESE USER            * 00320000
003300*           EXIT ROUTINES ARE CALLED BY DXT 'USER DATA           * 00330000
003400*           EXIT ROUTINES')                                      * 00340000
003500*                                                                * 00350000
003600*    THERE IS ONE DAX CONTROL BLOCK FOR EACH SEGMENT             * 00360000
003700*    EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT          * 00370000
003800*    IN VIRTUAL STORAGE.                                         * 00380000
003900*    FOR SYNCH PROPAGATION IN MPP REGIONS:                       * 00390000
004000*    - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER        * 00400000
004100*      SUBTASK.                                                  * 00410000
004200*    FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR             * 00420000
004300*    CCU AND DLU PROCESSING, AND FOR ASYNCH PROPAGATION          * 00430000
004400*    (DEPENDING ON HOW AYSNCH PROPAGATION IS IMPLEMENTED):       * 00440000
004500*    - THIS IS THE DURATION OF THE JOBSTEP.                      * 00450000
004600*                                                                * 00460000
004700*----------------------------------------------------------------* 00470000
004800*  IMPORTANT NOTES:                                              * 00480000
004900*  ================                                              * 00490000
005000*  - SINCE THE SAME USER EXIT ROUTINE CAN BE INVOKED BOTH        * 00500000
005100*    BY DPROP AND BY DXT: CHANGES TO THIS CONTROL BLOCK MUST     * 00510000
005200*    BE COORDINATED BETWEEN DPROP DEVELOPMENT AND DXT            * 00520000
005300*    DEVELOPMENT.                                                * 00530000
005400*                                                                * 00540000
005500*  - FIELDS MARKED IN THE COMMENT WITH '**DXT ONLY**'            * 00550000
005600*    HAVE NO MEANING, WHEN THE SEGMENT USER EXIT                 * 00560000
```

Figure 97 (Part 1 of 5). COBOL Interface Control Block for a Segment Exit Routine

```
*    ROUTINE IS INVOKED BY DPROP.                                    *
*                                                                    *
*--------------------------------------------------------------------*
*                                                                    *
*  CHANGE ACTIVITY:                                                  *
*                                                                    *
**************** END OF CONTROL BLOCK SPECIFICATION *************
*
 01      DAX.
*
*--------------------------------------------------------------------*
*   THIS SECTION OF THE CB MAY NOT BE MODIFIED BY EXIT               *
*--------------------------------------------------------------------*
*
  02     DAXPFX.
*            PREFIX OF CONTROL BLOCK
   03    DAXTNAME     PIC X(8).
*            EYE CATCHER: "DVRXCDAX"
   03    DAXRSVD      PIC X(24).
*            RESERVED FOR DXT INTERNAL USE
  02     DAXPFXE.
*            PREFIX EXTENSION
*
   03    DAXCALL      PIC XX.
*            TYPE OF CALL TO EXIT:
*              "NO" - NORMAL CALL, ISSUED TO CONVERT DATA
*                FROM IMS DATABASE FORMAT TO DPROP/DXT FORMAT
*              "RV" - REVERSE CALL, ISSUED TO CONVERT DATA
*                FROM DPROP/DXT FORMAT TO IMS DATABASE FORMAT
*
   03    DAXDATYP     PIC XX.
*            TYPE OF DATA BEING PASSED:
*              "DL" - DL/I DATA
*
   03    DAXFIL       PIC X(32).
*            NAME OF FILE OR PCB FROM WHICH DATA IS BEING PASSED
*
   03    DAXPSB       PIC X(8).
*            NAME OF PSB IF TYPE IS "DL"
*
   03    DAXSEGM      PIC X(32).
*            NAME OF SEGMENT IF TYPE IS "DL"
*             IF CALLER IS DPROP: NAME OF PHYSICAL SEGMENT
*             IF CALLER IS DXT:   NAME OF SEGMENT SPECIFIED IN
*                THE USED DBD (DBD CAN BE PHYSICAL OR LOGICAL)
*
   03    DAXPCBAD     POINTER.
**DXT ONLY** PTR TO PCB IF TYPE IS "DL"
*
   03    DAXPCBLS     POINTER.
**DXT ONLY** PTR TO LIST OF DEM'S PCBS, IF DEM IS A DL/I DEM
*
   03    DAXKFBAD     POINTER.
*            PTR TO SEGMENT'S FULLY CONCAT KEY (IF DL/I).
*            ZERO IF CALLER IS DPROP AND IF 'NOKEY' HAS BEEN
*            SPECIFIED ON EXIT= OF DBDGEN.
```

Figure 97 (Part 2 of 5). COBOL Interface Control Block for a Segment Exit Routine

```
011300*                                                              01130000
011400  03   DAXKFBLN     PIC S9(8) COMP.                            01140000
011500*           LENGTH OF SEGM'S FULLY CONCAT KEY (IF DL/I)        01150000
011600*           ZERO IF CALLER IS DPROP AND IF 'NOKEY' HAS BEEN    01160000
011700*           SPECIFIED ON EXIT= OF DBDGEN.                      01170000
011800*                                                              01180000
011900  03   DAXINLN.                                                01190000
012000*                                                              01200000
012100   04  DAXDLEN      PIC S9(8) COMP.                            01210000
012200*           LENGTH OF IMS DB SEGMENT BUFFER                    01220000
012300*                                                              01230000
012400  03   DAXOUTLN.                                               01240000
012500*                                                              01250000
012600   04  DAXFLEN      PIC S9(8) COMP.                            01260000
012700*           LENGTH OF DPROP SEGMENT BUFFER                     01270000
012800*                                                              01280000
012900  03   DAXSYSPR     POINTER.                                   01290000
013000**DXT ONLY** POINTER TO SYSPRINT DCB                           01300000
013100*                                                              01310000
013200  03   DAXENVT.                                                01320000
013300*           ENVIRONMENT SUBFIELDS                              01330000
013400   04  DAXOPSYS     PIC X(4).                                  01340000
013500*           OPERATING SYSTEM:                                  01350000
013600*             "ESA " IF MVS/ESA                                01360000
013700*                                                              01370000
013800   04  DAXTRANS     PIC X(4).                                  01380000
013900*           DB/DC ENVIRONMENT                                  01390000
014000*                                                              01400000
014100   04  DAXPROGM     PIC X(4).                                  01410000
014200*           CALLING PROGRAM:                                   01420000
014300*             "DXT " IF DXT                                    01430000
014400*             "DPRS" IF DPROP SYNCH PROP                       01440000
014500*             "DPRA" IF DPROP ASYNCH PROP                      01450000
014600*             "DPRC" IF DPROP CCU  PROP                        01460000
014700*             "DPRL" IF DPROP DLU                              01470000
014800*                                                              01480000
014900  03   DAXEXIT      PIC X(8).                                  01490000
015000*           NAME OF THIS EXIT ROUTINE                          01500000
015100*                                                              01510000
015200  03   DAXDBNM      PIC X(8).                                  01520000
015300*           NAME OF IMS DATABASE                               01530000
015400*            IF CALLER IS DPROP: NAME OF PHYSICAL DBD.         01540000
015500*            IF CALLER IS DXT:   NAME OF USED DBD (CAN BE NAME 01550000
015600*                                OF A PHYSICAL OR LOGICAL DBD) 01560000
015700*                                                              01570000
015800  03   DAXDPRPN     PIC X(24).                                 01580000
015900*           RESERVED                                           01590000
016000*                                                              01600000
016100  03   DAXASGNO     PIC S9(8) COMP.                            01610000
016200**DXT ONLY** NUMBER OF DAXASEGS ARRAY ELEMENTS                 01620000
016300*                                                              01630000
016400  03   DAXASEGS     PIC X(12)      OCCURS 15.                  01640000
016500**DXT ONLY** ARRAY OF ANCESTOR SEGMS                           01650000
016600*                                                              01660000
016700  03   DAXRSVD1     PIC X(46).                                 01670000
016800*           RESERVED FOR DXT USE                               01680000
```

Figure 97 (Part 3 of 5). COBOL Interface Control Block for a Segment Exit Routine

```
  03   FILLER  REDEFINES  DAXRSVD1.
*          RESERVED FOR DXT USE
    04   DAXDPRCT     PIC X(4).
*          IF CALLER IS DPROP, EXIT IS CALLED TO PROCESS:
*             "ISRT" - A DL/I OR DB2 INSERT
*             "DLET" - A DL/I OR DB2 DELETE
*             "REPL" - A DL/I OR DB2 REPLACE (AFTER-IMAGE)
*
    04   DAXREPL      PIC X.
*          IF CALLER IS DPROP AND IF DAXDPRCT IS "REPL":
     88  DAXREPLA             VALUE "A".
*            AFTER-REPLACE  IMAGE
     88  DAXREPLB             VALUE "B".
*            BEFORE-REPLACE  IMAGE
*
    04   DAXSEGT      PIC X.
*          IF CALLER IS DPROP, TYPE OF SEGMENT PROCESSED:
     88  DAXSEGTU             VALUE "U".
*            UPDATED IMS SEGMENT
     88  DAXSEGTA             VALUE "A".
*            ANCESTOR OF UPDATED SEGM
     88  DAXSEGTI             VALUE "I".
*            INTERNAL SEGMENT
*
    04   DAXPSUP      PIC X.
*          IF CALLER IS DPROP: DESCRIPTION WHETHER
*            PROPAGATION-SUPPRESSION IS ALLOWED:
     88  DAXPSUPN             VALUE "N".
*            SUPPRESSION NOT ALLOWED
     88  DAXPSUPY             VALUE "Y".
*            SUPPRESSION ALLOWED
*
    04   FILLER       PIC X.
*
    04   DAXISEGM     PIC X(8).
*          IF CALLER IS DPROP AND FOR RH PROPAGATION: NAME OF
*          SEGMENT TO PROCESS. SAME AS PHYSICAL IMS SEGMENT
*          NAME IN DAXSEGM IF NOT MAPPING CASE 3 ENTITY
*          (INTERNAL) SEGMENT IN PROCESS.
*
    04   DAXIDDSB     POINTER.
*          IF CALLER IS DPROP AND FOR RH PROPAGATION: POINTER
*          TO THE BUFFER CONTAINING THE BEFORE-CHANGE IMS DATA-
*          BASE SEGMENT. THIS BUFFER CONTAINS THE BEFORE IMAGE
*          OF THE IMS SEGMENT IF:
*            - DAXDPRCT EQ REPL, OR
*            - DAXDPRCT EQ DLET, OR
*            - DAXSEGT  EQ DAXSEGTI (INTERNAL SEGMENT)
*             OR CONTAINS ALL BINARY ZEROES IN OTHER CASES.
*          BUFFER IS READ ONLY FOR THE EXIT ROUTINE.
*
    04   DAXIDDSL     POINTER.
*          IF CALLER IS DPROP AND FOR RH PROPAGATION: LENGTH
*             OF THE 'BEFORE-CHANGE' IMS DB SEGMENT POINTED-TO
*             BY DAXIDDSB.
*
```

Figure 97 (Part 4 of 5). COBOL Interface Control Block for a Segment Exit Routine

```
    04   FILLER       PIC X(22).
*
*----------------------------------------------------------------*
*  THE NEXT GROUP OF FIELDS MAY BE MODIFIED BY THE EXIT ROUTINE *
*----------------------------------------------------------------*
*
   03    DAXENTRD     PIC X.
*            SET BY EXIT ROUTINE TO "X", INDICATES THAT EXIT
*            HAS BEEN ENTERED
*
   03    DAXINCTL     PIC X.
*            SET BY EXIT ROUTINE TO "X", INDICATES THAT EXIT
*            IS IN CONTROL
*
   03    DAXRETC      PIC S9(8) COMP.
*            RETURN CODE.
    88   DAXRCOK              VALUE 0.
*              0 = NORMAL, OUTPUT DATA RETURNED
    88   DAXRCOKR             VALUE 4.
*              4 = **DXT ONLY***
    88   DAXRCNQ              VALUE 8.
*              8 = IF CALLER IS DPROP: DPROP WILL SUPPRESS
*                     THE PROPAGATION OF THE CHANGED DL/I DATA
*                  IF CALLER IS DXT: DXT SHOULD NOT CONSIDER
*                     DATA TO BE ELIGIBLE FOR EXTRACT
    88   DAXRCERB             VALUE 12.
*             12 = ERROR
*                  - IF CALLER IS DPROP: PROPAGATION FAILURE.
*                       DPROP/RUP WILL GO THROUGH ITS USUAL
*                       ERROR HANDLING LOGIC.
*                  - IF CALLER IS DXT:   DXT SHOULD TERMINATE
    88   DAXRCERD             VALUE 16.
*             16 = ERROR
*                  - IF CALLER IS DPROP: RUP WILL ABEND
*                  - IF CALLER IS DXT:   DXT SHOULD TERMINATE
*                                        DEM EXECUTION
*
   03    DAXSMESG     PIC X(64).
*            TEXT OF MESSAGE PASSED FROM EXIT ROUTINE
*            TO DPROP/DXT. ALL BLANKS MEANS NO MESSAGE.
*            - IF CALLER IS DPROP: MSG WILL BE WRITTEN TO
*                 VARIOUS DESTINATIONS ACCORDING TO USUAL
*                 DPROP/RUP ERROR HANDLING LOGIC IN MESSAGE
*                 EKYR980I OR EKYR981E.
*            - IF CALLER IS DXT: TEXT OF MESSAGE WILL BE
*                 WRITTEN TO SYSPRINT DATA SET IN MESSAGE
*                 DVRA0_50. (UNDERSCORE IS REPLACED BY ONE OF
*                 SEVERAL DIGITS) HAS EFFECT FOR ALL CALLS.
*
   03    DAXDPRPM     PIC X(24).
*            STORAGE RESERVED FOR DATA EXIT
*
   03    DAXRSVD2     PIC X(32).
*            RESERVED FOR DXT USE
   03    DAXSCRT1     PIC X(128).
*            WORK SPACE (SCRATCHPAD) MAY BE USED BY EXIT
*
```

Figure 97 (Part 5 of 5). COBOL Interface Control Block for a Segment Exit Routine

Sample Segment Exit Control Block for PL/I

Figure 98 shows an example of the EKYRCDAX control block in PL/I. This control block, called EKYRCDXP, resides in the (EKYSAMP) library.

```
1/*************** START OF CONTROL BLOCK SPECIFICATION ****************
 *                                                                   *
 * Control Block name:                                               *
 *   EKYRCDXP (DAX)                                                  *
 *                                                                   *
 * Descriptive name:                                                 *
 *    DPROP PL/1 segment exit interface block.                       *
 *                                                                   *
 *    PL/1 version of EKYRCDAX                                       *
 *********************************************************************
 *                                                                   *
 *    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".           *
 *                                                                   *
 *    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                   *
 *    ALL RIGHTS RESERVED.                                           *
 *                                                                   *
 *    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                      *
 *    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                  *
 *    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                       *
 *                                                                   *
 *    LICENSED MATERIALS - PROPERTY OF IBM.                          *
 *                                                                   *
 *********************************************************************
 *                                                                   *
 * STATUS: V1 R2 M0                                                  *
 *                                                                   *
 * Function:                                                         *
 *    This is the PL/1 control block used to interface between       *
 *       - DPROP OR DXT                                              *
 *       and                                                         *
 *       - a user segment exit routine (these user exit routines are *
 *         called by DXT "user data exit routines")                  *
 *                                                                   *
 *    There is one DAX control block for each segment exit routine,  *
 *    lasting for the duration of the exit in virtual storage.       *
 *    For synchronous propagation in MPP regions:                    *
 *       this is the duration of the IMS program controller subtask. *
 *    For synchronous propagation in batch/BMP regions, for CCU and  *
 *    DLU processing, and for asynchronous propagation (depending    *
 *    on how aysnchronous propagation is implemented):               *
 *       this is the duration of the jobstep.                        *
 *********************************************************************
 *                                                                   *
 * Important note:                                                   *
 *    - Fields marked in the comment with '***** DXT only *****' have *
 *      no meaning, when the segment user exit routine is invoked by *
 *      DPROP.                                                       *
 *********************************************************************
 *                                                                   *
 * Change activity:                                                  *
 *    None                                                           *
 **************** END OF CONTROL BLOCK SPECIFICATION *****************/
1DECLARE 1 DAX BASED(DAX_POINTER),
 /********************************************************************
  * This section of the control block may not be modified by exit    *
  ********************************************************************/
    2 DAXPFX,                  /* Prefix of control block (32 bytes)  */
       3 DAXTNAME CHAR(8),       /* eye catcher ("DVRXCDAX")          */
       3 DAXRSVD  CHAR(24),      /* reserved for DXT internal use     */
```

Figure 98 (Part 1 of 5). PL/I Interface Control Block for a Segment Exit Routine

```
   2 DAXPFXE,                /* Prefix extension (448 bytes)       */
      3 DAXCALL  CHAR(2),       /* Type of call to exit:
                                   "NO" - normal call, ISSUED TO
                                    convert data from DL/I IO-area
                                    format to DPROP/DXT format.
                                   "RV" - reverse call  issued to
                                    convert data from DPROP/DXT
                                    format to DL/I IOarea format.
                   **DXT only**   "RE" - return call issued by DXT.
                   **DXT only**   "ED" - end-of-data call issued by
                                    DXT.                           */

      3 DAXDATYP CHAR(2),       /* Type of data being passed:
                                   "DL" - DL/I data
                   **DXT only**   "PS" - physical sequential data
                   **DXT only**   "VK" - VSAM KSDS data
                   **DXT only**   "VE" - VSAM ESDS data
                   **DXT only**   "GD" - GDI RECRD data            */

      3 DAXFIL   CHAR(32),      /* Name of file or PCB from which
                                   data is being passed            */

      3 DAXPSB   CHAR(8),       /* Name of PSB if type is "DL"      */

      3 DAXSEGM  CHAR(32),      /* Name of segment if type is "DL".
                                   If caller is DPROP:  name of
                                   physical segment.
                                   If caller is DXT:  name of segment
                                   specified in the used DBD (DBD can
                                   be physical or logical).        */

      3 DAXPCBAD POINTER,       /* ***** DXT only ***** pointer to
                                   PCB if type is "DL"             */

      3 DAXPCBLS POINTER,       /* ***** DXT only ***** Pointer to
                                   list of DEM's PCBs, if DEM is a
                                   DL/I DEM.                       */

      3 DAXKFBAD POINTER,       /* Pointer to segment's fully
                                   concatenated key (if DL/I).
                                   Zero if caller is DPROP and if
                                   "NOKEY" has been specified on
                                   "EXIT=" of DBDGEN.              */
1     3 DAXKFBLN FIXED BIN(31), /* Length of segm's fully concatenated
                                   concatenated key (if DL/I).
                                   Zero if caller is dprop and if
                                   "NOKEY" has been specified on
                                   "EXIT=" of DBDGEN.              */

      3 DAXINLN,                /* Length of input segment/record
                                   passed to segment exit routine.  */
         4 DAXDLEN FIXED BIN(31),  /* Alternate name, refers to length
                                      of DL/I ioarea format buffer.*/

      3 DAXOUTLN,               /* Length of output segment/record to
                                   be built by segment exit routine.*/
         4 DAXFLEN FIXED BIN(31),  /* Alternate name, refers to
                                     length of DPROP format buffer. */

      3 DAXSYSPR POINTER,       /* ***** DXT only ***** pointer
```

Figure 98 (Part 2 of 5). PL/I Interface Control Block for a Segment Exit Routine

```
                                to sysprint DCB               */

  3 DAXENVT,                 /* Environment subfields (12 bytes) */
     4 DAXOPSYS CHAR(4),        /* operating system:
                                   "ESA " if MVS/ESA
                                   **DXT only**   "XA  " if MVS/XA
                                   **DXT only**   "MVS " if MVS  */
     4 DAXTRANS CHAR(4),        /* DB/DC environment:
                                   "BAT " if IMS BATCH/BMP
                                   "MPP " if IMS MPP
                                   "IFP " if Fast Path
                                   "CICS" if CICS
                                   "    " if none of the above   */
     4 DAXPROGM CHAR(4),        /* Calling program:
                                   "DXT " if DXT
                                   "DPRS" if DPROP SYNCH PROP
                                   "DPRA" if DPROP ASYNCH PROP
                                   "DPRC" if DPROP CCU PROP
                                   "DPRL" if DPROP DLU           */

  3 DAXEXIT  CHAR(8),        /* Name of this exit routine       */

  3 DAXDBNM  CHAR(8),        /* Name of IMS data base.
                                If caller is DPROP:
                                   name of physical DBD.
                                If caller is DXT:
                                   name of used dbd (can be name
                                   of a physical or logical DBD) */

  3 DAXDPRPN CHAR(24),       /* Reserved                        */

  3 DAXASGNO FIXED BIN(31),  /* ***** DXT only ***** number
                                of DAXASEGS array elements      */

  3 DAXASEGS(15) CHAR(12),   /* ***** DXT only ***** array
                                of ancestor segments            */
1 3 DAXDPRCT CHAR(4),        /* If caller is DPROP,
                                exit is called to process:
                                "ISRT" - a DL/I or DB2 insert
                                "DLET" - a DL/I or DB2 delete
                                "REPL" - a DL/I or
                                    DB2 replace (after-image)    */

  3 DAXREPL  CHAR(1),        /* If caller is DPROP and
                                if DAXDPRCT is "REPL":
                                   "A" - after replace
                                   "B" - before replace         */

  3 DAXSEGT  CHAR(1),        /* If caller is DPROP,
                                type of segment processed:
                                   "U" - updated IMS segment
                                   "A" - ancestor of updated seg
                                   "I" - internal segment       */

  3 DAXPSUP  CHAR(1),        /* If caller is DPROP:  description
                                whether propagation-suppression is
                                allowed:
                                   "N" - suppression not allowed
                                   "Y" - suppression is allowed  */
```

Figure 98 (Part 3 of 5). PL/I Interface Control Block for a Segment Exit Routine

```
      3 FILL01   CHAR(1),         /* Reserved                          */
      3 DAXISEGM CHAR(8),         /* If caller is DPROP and for RH
                                     propagation: name of segment to
                                     process.  Same as physical IMS
                                     segment name in DAXSEGM if not
                                     mapping case 3 entity (internal)
                                     segment in process.               */

      3 DAXIDDSB POINTER,         /* If caller is DPROP and for RH
                                     propagation: pointer to DL/I DB
                                     segment buffer.
                                     This buffer contains contains the
                                     before image of the IMS segment if:
                                        - DAXDPRCT equals REPL, or DLET
                                          or
                                        - DAXSEGT equals DAXSEGTI
                                          (internal seg)
                                     else contains all binary zeroes
                                     in other cases.
                                     Buffer is read only for
                                     the exit routine.                 */

      3 DAXIDDSL POINTER,         /* If caller is DPROP and for RH
                                     propagation: length of the
                                     'before-change' IMS DB segment
                                     pointed-to by DAXIDDSB.           */
      3 FILL22 CHAR(22),          /* Filler                            */
1     /****************************************************************
       *The next group of fields may be modified by the exit routine.*
       ****************************************************************/
      3 DAXENTRD CHAR(1),         /* Set by exit to "X" indicating
                                     that the exit has been entered.  */
      3 DAXINCTL CHAR(1),         /* Set by exit to "X" indicating
                                     that exit is in control.          */
      3 DAXRETC  FIXED BIN(31),   /* Return code.
                                     0 = normal, output data returned.
                                     4 = **DXT ONLY***
                                     8 = If caller is DPROP:
                                         Propagation of the DL/I
                                         changed data will be
                                         suppressed.
                                         If caller is DXT:
                                         DXT should not consider data
                                         to be eligible for extract.
                                    12 = ERROR
                                         If caller is DPROP:
                                            Propagation failure.
                                            DPROP/RUP will go through
                                            its usual error handling
                                            logic.
                                         If caller is DXT:
                                            DXT should terminate.

                                    16 = ERROR
                                         If caller is DPROP:
                                         RUP will abend.
                                         If caller is DXT:
                                         DXT should terminate DEM
                                         execution.                    */
```

Figure 98 (Part 4 of 5). PL/I Interface Control Block for a Segment Exit Routine

```
      3 DAXSMESG CHAR(64),       /* Text of message passed from exit
                                    routine to DPROP/DXT.
                                    All blanks means no message.

                                    If caller is DPROP:
                                    Message will be written to various
                                    destinations according to usual
                                    DPROP/RUP error handling logic in
                                    message EKYR980I or EKYR981E.

                                    If caller is DXT:  text of message
                                    will be written to SYSPRINT dataset
                                    in message DVRA0_50, (underscore is
                                    replaced by one of several digits)
                                    has effect for all calls.        */

      3 DAXDPRPM CHAR(24),       /* Storage reserved for data exit.  */
      3 DAXRSVD2 CHAR(32),       /* Reserved for DXT use.            */
      3 DAXSCRT1 CHAR(128);      /* Work space (scratchpad), may be
                                    used by the exit as desired.     */
```

Figure 98 (Part 5 of 5). PL/I Interface Control Block for a Segment Exit Routine

Sample Segment Exit Control Block for C

Figure 99 shows an example of the EKYRCDAX control block in C. This control block, called EKYRCDXK, resides in the (EKYSAMP) library.

```
/*************** START OF CONTROL BLOCK SPECIFICATION *****************
*                                                                    *
* Control Block name:                                                *
*    EKYRCDXK (DAX)                                                  *
*                                                                    *
* Descriptive name:                                                  *
*    DPROP C language segment exit interface block.                  *
*    C language version of EKYRCDAX                                  *
**********************************************************************
*     THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".           *
*                                                                    *
*     5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                   *
*     ALL RIGHTS RESERVED.                                           *
*                                                                    *
*     U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                      *
*     USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                  *
*     GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                       *
*                                                                    *
*     LICENSED MATERIALS - PROPERTY OF IBM.                          *
**********************************************************************
*                                                                    *
* STATUS: V1 R2 M0                                                   *
*                                                                    *
* Function:                                                          *
*    This is the C language control block used to interface between *
*       - DPROP OR DXT                                               *
*       and                                                          *
*       - a user's segment exit routine (these user exit routines    *
*          are called by dxt 'user data exit routines')              *
*                                                                    *
*    There is one DAX control block for each segment exit routine,   *
*    lasting for the duration of the exit in virtual storage.        *
*    For synchronous propagation in MPP regions:                     *
*    -  this is the duration of the IMS program controller subtask.  *
*    For synchronous propagation in batch/BMP regions, for CCU and   *
*    DLU processing, and for asynchronous propagation (depending     *
*    on how asynchronous propagation is implemented):                *
*    -  this is the duration of the jobstep.                         *
**********************************************************************
* Important notes:                                                   *
*    Since the same user exit routine can be invoked both by DPROP   *
*    and by DXT: changes to this control block must be coordinated   *
*    between DPROP development and DXT development.                  *
*                                                                    *
*    Fields marked in the comment with '***** DXT only *****' have   *
*    no meaning, when the segment user exit routine is invoked by    *
*    DPROP.                                                          *
**********************************************************************
*                                                                    *
* Change activity:                                                   *
*    None                                                            *
**************** END OF CONTROL BLOCK SPECIFICATION *******************/

#pragma page(1)

typedef
struct
{                              /* DAX                                */
/*********************************************************************
 * This section of the control block may not be modified by exit     *
 *********************************************************************/
```

Figure 99 (Part 1 of 5). C Interface Control Block for a Segment Exit Routine

```
                                /* DAXPFX                             */
                                /* Control block prefix (32 bytes)   */
  unsigned char daxtname[8];    /* Eye catcher ("dvrxcdax")           */
  unsigned char daxrsvd[24];    /* Reserved for DXT internal use      */
 /*******************************************************************/
                                /* DAXPFXE                            */
                                /* Prefix extension (448 bytes)       */
  unsigned char daxcall[2];     /* Type of call to exit:
                                   "NO" - normal call, ISSUED TO
                                   convert data from DL/I IO-area
                                   format to DPROP/DXT format.
                                   "RV" - reverse call  issued to
                                   convert data from DPROP/DXT
                                   format to DL/I IOarea format.
                    **DXT only**  "RE" - return call issued by DXT.
                    **DXT only**  "ED" - end-of-data call issued by
                                    DXT.                               */

  unsigned char daxdatyp[2];    /* Type of data being passed:
                                   "DL" - DL/I data
                    **DXT only**  "PS" - physical sequential data
                    **DXT only**  "VK" - VSAM KSDS data
                    **DXT only**  "VE" - VSAM ESDS data
                    **DXT only**  "GD" - GDI RECRD data                */

  unsigned char daxfil[32];     /* Name of file or PCB from which
                                   data is being passed                */

  unsigned char daxpsb[8];      /* Name of PSB if type is "DL"         */

  unsigned char daxsegm[32];    /* Name of segment if type is "DL".
                                   If caller is DPROP:  name of
                                   physical segment.
                                   If caller is DXT:  name of segment
                                   specified in the used DBD (DBD can
                                   be physical or logical).            */

          char *daxpcbad;       /* ***** DXT only ***** pointer to
                                   PCB if type is "DL"                 */

          char *daxpcbls;       /* ***** DXT only ***** pointer to
                                   list of DEM's PCBs, if DEM is a
                                   DL/I DEM.                           */
#pragma page(1)
          char *daxkfbad;       /* Pointer to segment's fully
                                   concatenated key (if DL/I).
                                   Zero if caller is DPROP and if
                                   "NOKEY" has been specified on
                                   "EXIT=" of DBDGEN.                  */

          long daxkfbln;        /* Length of segm's fully concatenated
                                   concatenated key (if DL/I).
                                   Zero if caller is dprop and if
                                   "NOKEY" has been specified on
                                   "EXIT=" of DBDGEN.                  */
/*********************************************************************/
                                /* DAXINLN                            */
                                /* Length of input segment/record
                                   passed to segment exit routine.    */
          long daxdlen;         /* alternate name, refers to length
```

Figure 99 (Part 2 of 5). C Interface Control Block for a Segment Exit Routine

```
                                    of DL/I ioarea format buffer.     */
/*********************************************************************/
                                 /* DAXOUTLN                          */
                                 /* Length of output segment/record to
                                    be built by segment exit routine. */
           long daxflen;         /* Alternate name, refers to length
                                    of DPROP format buffer.           */
/*********************************************************************/
           char *daxsyspr;       /* ***** DXT only ***** pointer
                                    to sysprint DCB                   */
/*********************************************************************/
                                 /* DAXENVT                           */
                                 /* Environment subfields (12 bytes)  */
  unsigned char daxopsys[4];     /* Operating system:
                                       "ESA " if MVS/ESA
                                       **DXT only**   "XA  " if MVS/XA
                                       **DXT only**   "MVS " if MVS   */
  unsigned char daxtrans[4];     /* DB/DC environment:
                                       "BAT " if IMS BATCH/BMP
                                       "MPP " if IMS MPP
                                       "IFP " if Fast Path
                                       "CICS" if CICS
                                       "    " if none of the above    */
  unsigned char daxprogm[4];     /* Calling program:
                                       "DXT " if DXT
                                       "DPRS" if DPROP SYNCH PROP
                                       "DPRA" if DPROP ASYNCH PROP
                                       "DPRC" if DPROP CCU PROP
                                       "DPRL" if DPROP DLU            */
/*********************************************************************/
#pragma page(1)

  unsigned char daxexit[8];     /* Name of this exit routine          */

  unsigned char daxdbnm[8];     /* Name of IMS data base.
                                   If caller is DPROP:
                                      name of physical DBD.
                                   If caller is DXT:
                                      name of used DBD (can be name
                                      of a physical or logical DBD)   */

  unsigned char daxdprpn[24];    /* Reserved                          */

           long daxasgno;        /* ***** DXT only ***** number
                                    of DAXASEGS array elements        */

           char daxasegs[15][12];       /* ***** DXT only ***** array
                                                of ancestor segments  */

  unsigned char daxdprct[4];     /* If caller is DPROP,
                                    exit is called to process:
                                       "ISRT" - a DL/I or DB2 insert
                                       "DLET" - a DL/I or DB2 delete
                                       "REPL" - a DL/I or
                                           DB2 replace (after-image)  */

  unsigned char daxrepl;         /* If caller is DPROP and
                                    if DAXDPRCT is "REPL":
                                       "A" - after replace
                                       "B" - before replace           */
```

Figure 99 (Part 3 of 5). C Interface Control Block for a Segment Exit Routine

```
  unsigned char daxsegt;         /* If caller is DPROP,
                                    type of segment processed:
                                       "U" - updated IMS segment
                                       "A" - ancestor of updated seg
                                       "I" - internal segment        */

  unsigned char daxpsup;         /* If caller is DPROP: description
                                    whether propagation-suppression is
                                    allowed:
                                       "N" - suppression not allowed
                                       "Y" - suppression is allowed   */
  unsigned char fill01;          /* Reserved                          */
  unsigned char daxisegm[8];     /* If caller is DPROP and for RH
                                    propagation: name of segment to
                                    process.  Same as physical IMS
                                    segment name in DAXSEGM if not
                                    mapping case 3 entity (internal)
                                    segment in process.               */
#pragma page(1)

           char *daxiddsb;       /* If caller is DPROP and for RH
                                    propagation: pointer to DL/I DB
                                    segment buffer.
                                    This buffer contains contains the
                                    before image of the IMS segment if:
                                       - DAXDPRCT equals REPL, or DLET
                                         or
                                       - DAXSEGT equals DAXSEGTI
                                         (internal seg)
                                    else contains all binary zeroes
                                    in other cases.
                                    Buffer is read only for
                                    the exit routine.                 */
           char *daxiddsl;       /* If caller is DPROP and for RH
                                    propagation: length of the
                                    'before-change' IMS DB segment
                                    pointed-to by DAXIDDSB.           */
  unsigned char fill22[22];      /* Filler                            */

 /*********************************************************************
  *The next group of fields may be modified by the exit routine.      *
  *********************************************************************/
  unsigned char daxentrd;        /* Set by exit to "x" indicating
                                    that the exit has been entered.  */
  unsigned char daxinctl;        /* Set by exit to "x" indicating
                                    that exit is in control.         */
           long daxretc;         /* Return code.
                                    0 = normal, output data returned.
                                    4 = **DXT ONLY***
                                    8 = If caller is DPROP:
                                        Propagation of the DL/I
                                        changed data will be
                                        suppressed.
                                        If caller is DXT:
                                        DXT should not consider data
                                        to be eligible for extract.
                                   12 = ERROR
                                        If caller is DPROP:
                                           Propagation failure.
                                           DPROP/RUP will go through
                                           its usual error handling
                                           logic.
```

Figure 99 (Part 4 of 5). C Interface Control Block for a Segment Exit Routine

```
                                        If caller is DXT:
                                           DXT should terminate.

                                   16 = ERROR
                                        If caller is DPROP:
                                        RUP will abend.
                                        If caller is DXT:
                                        DXT should terminate DEM
                                        execution.                      */
#pragma page(1)

  unsigned char daxsmesg[64];    /* Text of message passed from exit
                                    routine to DPROP/DXT.
                                    All blanks means no message.

                                    If caller is DPROP:
                                    Message will be written to various
                                    destinations according to usual
                                    DPROP/RUP error handling logic in
                                    message EKYR980I or EKYR981E.

                                    If caller is DXT:  text of message
                                    will be written to SYSPRINT dataset
                                    in message DVRA0_50, (underscore is
                                    replaced by one of several digits)
                                    has effect for all calls.         */

  unsigned char daxdprpm[24];    /* Storage reserved for data exit.   */
  unsigned char daxrsvd2[32];    /* Reserved for DXT use.             */
  unsigned char daxscrt1[128];   /* Work space (scratchpad), may be
                                    used by the exit as desired.      */
 } EKYRCDAX;

#pragma page(1)
```

Figure 99 (Part 5 of 5). C Interface Control Block for a Segment Exit Routine

Appendix C. Sample Field Exit Control Blocks

This appendix contains sample Field exit control blocks which map the existing DPROP interface control blocks. This appendix provides the exit control blocks in three languages:

- COBOL
- PL/I
- C

Figure 28 on page 115 shows the Assembler version of the Field exit control block.

Sample Field Exit Control Block for COBOL

Figure 100 shows an example of the EKYRCUDT control block in COBOL. This control block, called EKYRCUDC, resides in the DPROP Sample Source library (EKYSAMP).

```
000100*************** START OF CONTROL BLOCK SPECIFICATION *********** 00010000
000200*                                                               * 00020000
000300* CONTROL BLOCK NAME:                                           * 00030000
000400*    EKYRCUDC (UDT)                                             * 00040000
000500*                                                               * 00050000
000600*  DESCRIPTIVE NAME:                                            * 00060000
000700*    DPROP COBOL FIELD EXIT INTERFACE                           * 00070000
000800*                                                               * 00080000
000900*    COBOL VERSION OF EKYRCUDT                                  * 00090000
001000*                                                               * 00100000
001100**************************************************************** 00110000
001200*                                                               * 00120000
001300*    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       * 00130000
001400*                                                               * 00140000
001500*    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                    * 00150000
001600*    ALL RIGHTS RESERVED.                                       * 00160000
001700*                                                               * 00170000
001800*    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  * 00180000
001900*    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              * 00190000
002000*    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   * 00200000
002100*                                                               * 00210000
002200*    LICENSED MATERIALS - PROPERTY OF IBM.                      * 00220000
002300*                                                               * 00230000
002400**************************************************************** 00240000
002500*                                                               * 00250000
002600*  STATUS: V1 R2 M0                                             * 00260000
002700*                                                               * 00270000
002800*  FUNCTION:                                                    * 00280000
002900*    THIS IS THE COBOL CONTROL BLOCK USED TO INTERFACE BETWEEN  * 00290000
003000*         - DPROP OR DXT                                        * 00300000
003100*         AND                                                   * 00310000
003200*         - A USER'S FIELD EXIT ROUTINE (THESE USER             * 00320000
003300*           EXIT ROUTINES ARE CALLED BY DXT 'USER DATA TYPE     * 00330000
003400*           EXIT ROUTINES')                                     * 00340000
003500*                                                               * 00350000
003600*    THERE IS ONE CONTROL BLOCK FOR EACH FIELD                  * 00360000
003700*    EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT         * 00370000
003800*    IN VIRTUAL STORAGE.                                        * 00380000
003900*    FOR SYNCH PROPAGATION IN MPP REGIONS:                      * 00390000
004000*    - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER       * 00400000
004100*      SUBTASK.                                                 * 00410000
004200*    FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR            * 00420000
004300*    ASYNCH PROPAGATION, AND FOR CCU PROCESSING:                * 00430000
004400*    - THIS IS THE DURATION OF THE JOBSTEP.                     * 00440000
004500*                                                               * 00450000
004600*---------------------------------------------------------------* 00460000
004700*  IMPORTANT NOTES:                                             * 00470000
004800*  ===============                                              * 00480000
004900*                                                               * 00490000
005000*  - SINCE THE SAME USER EXIT ROUTINE CAN BE INVOKED BOTH       * 00500000
005100*    BY DPROP AND BY DXT: CHANGES TO THIS CONTROL BLOCK MUST    * 00510000
```

Figure 100 (Part 1 of 4). COBOL Interface Control Block for a Field Exit Routine

```
*    BE COORDINATED BETWEEN DPROP DEVELOPMENT AND DXT          *
*    DEVELOPMENT.                                              *
*                                                              *
*--------------------------------------------------------------*
*                                                              *
*  CHANGE ACTIVITY:                                            *
*                                                              *
**************** END OF CONTROL BLOCK SPECIFICATION *************
*
 01      EKYRCUDC.
*
*--------------------------------------------------------------*
* THIS SECTION OF THE CB MAY NOT BE MODIFIED BY THE EXIT.      *
*--------------------------------------------------------------*
*
  02     UDTPFX.
   03    UDTTNAME     PIC X(8).
*            NAME OF CONTROL BLOCK. MAPS TO DVRXCUDT
   03    FILLER       PIC X(24).
*            RESERVED FOR DXT USE
*
  02     UDTPFXE.
*            PREFIX EXTENSION
*
   03    UDTPNMOD.
    04   UDTCALL      PIC X(2).
*            TYPE OF CALL TO EXIT...
     88 UDTCSRTG    VALUE "ST".
*            "SRC --> TRG" CALL ISSUED BY DXT AND BY DPROP
*            DURING HR MAPPING. EXIT SHOULD CONVERT THE DATA
*            FROM THE USER FORMAT TO THE DPROP FORMAT.
     88 UDTCTGSR    VALUE "TS".
*            "TRG --> SRC" CALL ISSUED BY DPROP DURING RH
*            MAPPING. EXIT SHOULD CONVERT DATA FROM THE
*            DPROP FORMAT TO THE USER FORMAT.
     88 UDTCDEFN    VALUE "DF".
*            **NOT** ISSUED BY DPROP.
*            DEFINITION CALL ISSUED BY DXT-UIM FOR EACH DATATYPE.
*            EXIT CAN VALIDATE REQUEST AND RETURN REQUIRED VALUES
*
    04   FILLER       PIC X(2).
*            RESERVED FOR DXT USE
    04   UDTENVRN.
*            ENVIRONMENTAL INFORMATION
     05  UDTOPSYS     PIC X(4).
*            OPERATING SYSTEM CALLING PROGRAM IS EXECUTING IN:
     88 UDTOSMVS    VALUE "MVS ".
*            INDICATES DXT IS RUNNING IN MVS/370 ENVIRONMENT.
     88 UDTOSXA     VALUE "XA  ".
*            INDICATES DXT IS RUNNING IN MVS/XA ENVIRONMENT.
     88 UDTOSESA    VALUE "ESA ".
```

Figure 100 (Part 2 of 4). COBOL Interface Control Block for a Field Exit Routine

```
010300*           INDICATES DXT IS RUNNING IN MVS/ESA ENVIRONMENT.     01030000
010400     05  UDTTRANS     PIC X(4).                                  01040000
010500*           DB/DC ENVIRONMENT: 'BAT ' IF IMS BATCH/BMP           01050000
010600*                              'MPP ' IF IMS MP                  01060000
010700*                              'IFP ' IF FAST PATH               01070000
010800*                              'CICS' IF CICS                    01080000
010900*                              '    ' IF NONE OF ABOVE           01090000
011000     05  UDTPROGM     PIC X(4).                                  01100000
011100*           CALLING PROGRAM:   'DXT ' IF DXT                     01110000
011200*                              'DPRS' IF DPROP SYNCH PROP        01120000
011300*                              'DPRA' IF DPROP ASYNCH PROP       01130000
011400*                              'DPRC' IF DPROP CCU PROCESSING    01140000
011500*                              'DPRL' IF DPROP DLU               01150000
011600    04   UDTEXIT      PIC X(8).                                  01160000
011700*           NAME OF THE USER EXIT                                01170000
011800    04   UDTPCBLS      PIC X(4).                                 01180000
011900    04   UDTDPRP1      PIC X(24).                                01190000
012000*---------------------------------------------------------------- 01200000
012100* THIS SECTION CONTAINS DATA PERTINENT TO THE SOURCE FIELD       01210000
012200*---------------------------------------------------------------- 01220000
012300    04   UDTSTYPE     PIC X(2).                                  01230000
012400*           SOURCE DATA TYPE VALUE                               01240000
012500    04   UDTSBYTI     PIC X(1).                                  01250000
012600*           LENGTH INDICATOR FOR USER FORMAT (DXT ONLY)          01260000
012700     88  UDTSBYIN     VALUE "N".                                 01270000
012800*           LENGTH OF USER FORMAT RESIDES WITH THE DEFINITION.   01280000
012900     88  UDTSBYIV     VALUE "V".                                 01290000
013000*           LENGTH OF USER FORMAT VARIES AND MUST BE RETURNED    01300000
013100*           AT "DEFINITION" TIME.                                01310000
013200    04   FILLER       PIC X(1).                                  01320000
013300*           RESERVED FOR DXT USE                                 01330000
013400    04   UDTSBYTV     PIC S9(4)  COMP.                           01340000
013500*           LENGTH OF FIELD IN USER FORMAT.                      01350000
013600    04   UDTSSCLI     PIC X(1).                                  01360000
013700*           SCALE INDICATOR FOR USER FORMAT (DXT ONLY)           01370000
013800     88  UDTSSCLN     VALUE "N".                                 01380000
013900*           SCALE OF USER FORMAT RESIDES WITH THE DEFINITION.    01390000
014000     88  UDTCSCLV     VALUE "V".                                 01400000
014100*           SCALE OF USER FORMAT VARIES AND MUST BE RETURNED     01410000
014200*           AT "DEFINITION" TIME.                                01420000
014300    04   FILLER       PIC X(3).                                  01430000
014400*           RESERVED FOR DXT USE                                 01440000
014500    04   UDTSSCLV     PIC S9(4)  COMP.                           01450000
014600*           VALUE OF SCALE IN USER FORMAT                        01460000
014700*                                                                01470000
014800*---------------------------------------------------------------- 01480000
014900* THIS SECTION CONTAINS DATA PERTINENT TO THE TARGET FIELD       01490000
015000*---------------------------------------------------------------- 01500000
015100    04   UDTTTYPE     PIC X(2).                                  01510000
015200*           DATA TYPE OF DPROP FORMAT                            01520000
015300    04   UDTTBYTI     PIC X(1).                                  01530000
```

Figure 100 (Part 3 of 4). COBOL Interface Control Block for a Field Exit Routine

```
      *           LENGTH INDICATOR FOR DPROP FORMAT (DXT ONLY)
            88  UDTTBYIN     VALUE "N".
      *           LENGTH OF DPROP FORMAT RESIDES WITH THE DEFINITION.
            88  UDTTBYIV     VALUE "V".
      *           LENGTH OF DPROP FORMAT VARIES AND MUST BE RETURNED
      *           AT "DEFINITION" TIME.
           04   FILLER       PIC X(1).
      *           RESERVED FOR DXT USE
           04   UDTTBYTV     PIC S9(4)  COMP.
      *           LENGTH OF FIELD IN DPROP FORMAT
           04   UDTTSCLI     PIC X(1).
      *           SCALE INDICATOR FOR DPROP FORMAT (DXT ONLY)
            88  UDTTSCLN     VALUE "N".
      *           SCALE OF DPROP FORMAT RESIDES WITH THE DEFINITION.
            88  UDTTSCLV     VALUE "V".
      *           SCALE OF DPROP FORMAT VARIES AND MUST BE RETURNED
      *           AT "DEFINITION" TIME.
           04   FILLER       PIC X(3).
      *           RESERVED FOR DXT USE
           04   UDTTSCLV     PIC S9(4)  COMP.
      *           VALUE OF SCALE IN DPROP FORMAT
      *-----------------------------------------------------------------
      * THIS SECTION IS THE COMMUNICATIONS AREA BETWEEN THE EXIT
      *   AND DPROP/DXT.
      *-----------------------------------------------------------------
          03    UDTXICOM.
      *           DEFINE A COMMUNICATIONS AREA
           04   UDTDPRP2     PIC X(24).
      *           RESERVED
           04   UDTSCRT1     PIC X(128).
           04   UDTXITWS     REDEFINES UDTSCRT1 PIC X(128).
      *           USER EXIT WORK AREA
           04   UDTENTRD     PIC X(1).
      *           'ENTERED' FLAG - SET TO X BY EXIT TO INDICATE
      *           THAT DATA TYPE ROUTINE HAS BEEN ENTERED.
           04   UDTINCTL     PIC X(1).
      *           'IN-CONTROL' FLAG - SET TO X BY EXIT TO INDICATE
      *           THAT DATA TYPE ROUTINE IS IN CONTROL.
           04   UDTNULLT     PIC X(1).
      *             DATA RETURNED FROM EXIT IS NULL.
              88  UDTNULLY     VALUE "Y".
      *             RETURN DATA IS NULL.
              88  UDTNULLN     VALUE "N".
      *             RETURNED DATA IS NOT NULL.
           04   FILLER       PIC X(1).
      *           RESERVED
           04   UDTXRETC     PIC S9(8)  COMP.
      *           USER EXIT RETURN CODE
      *             0     - SUCCESSFUL COMPLETION
      *             OTHER - ERROR ENCOUNTERED
      *                     IF CALLER IS DPROP:
      *                       4 ---> RUP WILL USE IST USUAL
      *                              ERROR HANDLING LOGIC.
      *                      ⅛4 ---> RUP ABENDS
           04   UDTXMESG     PIC X(64).
      *           USER EXIT MESSAGE TEXT INSERTED INTO DPROP/DXT
      *           MESSAGE. IF CALLER IS DPROP, TEXT WILL BE
      *           INSERTED INTO MSG EKYR970I/EKYR971E.
```

Figure 100 (Part 4 of 4). COBOL Interface Control Block for a Field Exit Routine

Sample Field Exit Control Block for PL/I

Figure 101 shows an example of the EKYRCUDT control block in PL/I. This control block, called EKYRCUDP, resides in the (EKYSAMP) library.

```
1/******************* START OF CONTROL BLOCK SPECIFICATION ***********
  *                                                                 *
  * Control Block name:                                             *
  *    EKYRCUDP                                                     *
  *                                                                 *
  * Descriptive name:                                               *
  *    DPROP PL/1 field exit interface.                             *
  *                                                                 *
  *    PL/1 version of EKYRCUDT.                                    *
  *                                                                 *
  *******************************************************************
  *                                                                 *
  *    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".         *
  *                                                                 *
  *    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                 *
  *    ALL RIGHTS RESERVED.                                         *
  *                                                                 *
  *    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                    *
  *    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                *
  *    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                     *
  *                                                                 *
  *    LICENSED MATERIALS - PROPERTY OF IBM.                        *
  *                                                                 *
  *******************************************************************
  *                                                                 *
  * Status: V1 R2 M0                                                *
  *                                                                 *
  * Function:                                                       *
  *    This is the PL/1 control block used to interface between     *
  *       - DPROP or DXT                                            *
  *         and                                                     *
  *       - a user's field exit routine (these user exit routines   *
  *         are called by DXT 'user data type exit routines')       *
  *                                                                 *
  *    There is one control block for each field exit routine,      *
  *    lasting for the duration of the exit in virtual storage.     *
  *    For synchronous propagation in MPP regions:                  *
  *    - this is the duration of the IMS program controller subtask. *
  *    For synchronous propagation in Batch/BMP regions, for        *
  *    asynchronous propagation, and for CCU processing:            *
  *    - this is the duration of the jobstep.                       *
  *******************************************************************
  *  Important Notes:                                               *
  *                                                                 *
  *     Since the same user exit routine can be invoked both by     *
  *     DPROP and by DXT: changes to this control block must be     *
  *     coordinated between DPROP development and DXT development.  *
  *                                                                 *
  *******************************************************************
  *                                                                 *
  * Change activity:                                                *
  *    None.                                                        *
  ******************* END OF CONTROL BLOCK SPECIFICATION *************/
1DECLARE UDT_PTR POINTER;
 DECLARE 1 EKYRCUDP BASED (UDT_PTR),
 /*******************************************************************
  * This section of the control block may not be modified by the exit.*
  *******************************************************************/
    2 UDTPFX,                 /* DXT prefix (32 bytes)            */
       3 UDTTNAME CHAR(8),        /* Name of block, "DVRXCUDT"        */
       3 UDTXADDR POINTER,        /* Address of loaded routine        */
```

Figure 101 (Part 1 of 4). PL/I Interface Control Block for a Field Exit Routine

```
      3 FILL20   CHAR(20),       /* Reserved for DXT use            */

   2 UDTPFXE,                 /* Prefix extension (300 bytes)       */
      3 UDTPNMOD,                /* (76 bytes)                      */

         4 UDTCALL  CHAR(2),
            /*********************************************************
             * Type of call to exit:                                 *
             * "DF" - This is *** NOT *** issued by DPROP.  But is a *
             * definition call issued by DXT-UIM for each datatype.  *
             * Exit can validate request and return required values. *
             * "ST" - Source->target call issued by DXT and by DPROP *
             * during HR mapping. Exit should convert data from the  *
             * DL/I IOarea format to DPROP supported target datatype.*
             * "TS" - Target->source call issued by DPROP during RH  *
             * mapping.Exit should convert data from DPROP supported *
             * datatype to DL/I IOarea format not issued by DXT.     *
          ***********************************************************/

         4 FILL02   CHAR(2),        /* Reserved for DXT use         */
1        4 UDTENVRN,                /* Environmental information
                                       (12 bytes)                   */
            5 UDTOPSYS CHAR(4),
            /*********************************************************
             * Operating system calling program is executing in:     *
             * When UDTOPSYS = "MVS ".                               *
             * This indicates DXT is running in MVS/370 environment. *
             * When UDTOPSYS = "XA  ".                               *
             * This indicates DXT is running in MVS/XA environment.  *
             * When UDTOPSYS = "ESA ".                               *
             * This indicates DXT is running in MVS/ESA environment. *
             *********************************************************/

            5 UDTTRANS CHAR(4),
            /*********************************************************
             * DB/DC environment: 'BAT ' if IMS Batch/BMP            *
             *                    'MPP ' if IMS MPP                  *
             *                    'IFP ' if Fast Path                *
             *                    'CICS' if CICS                     *
             *                    '    ' if none of the above.       *
             *********************************************************/

            5 UDTPROGM CHAR(4),
            /*********************************************************
             * Calling program:   'DXT ' if DXT                      *
             *                    'DPRS' if DPROP synchronous PROP   *
             *                    'DPRA' if DPROP asynchronous PROP  *
             *                    'DPRC' if DPROP CCU processing     *
             *                    'DPRL' if DPROP DLU                *
             *********************************************************/

         4 UDTEXIT  CHAR(8),        /* Name of the user exit        */

         4 UDTPCBLS POINTER,        /* *** DXT only ***
                                    /* Address list of all PCB
                                       addresses if DLI environment  */
         4 UDTDPRP1 CHAR(24),       /* Additional work space        */
1        /***********************************************************
          * This section contains data pertinent to the source field *
          ***********************************************************/
```

Figure 101 (Part 2 of 4). PL/I Interface Control Block for a Field Exit Routine

```
          4 UDTSTYPE CHAR(2),        /* Source data type value       */

          4 UDTSBYTI CHAR(1),
             /********************************************************
              * Source bytes indicator (DXT only).                   *
              * "N" - indicates value resides with the definition.   *
              * "V" - indicates value is returned at "DEF" call to UIM*
              *********************************************************/

          4 FILL01A  CHAR(1),        /* Reserved for DXT use         */

          4 UDTSBYTV FIXED BIN(15),  /* Number of source bytes       */

          4 UDTSSCLI CHAR(1),
             /********************************************************
              * Source scale indicator (DXT only).                   *
              * "N" - indicates value resides with the definition.   *
              * "V" - indicates value is returned at "DEF" call to UIM*
              *********************************************************/

          4 FILL03A  CHAR(3),        /* Reserved for DXT use         */

          4 UDTSSCLV FIXED BIN(15),  /* Value of source scale        */
1         /************************************************************
           * This section contains data pertinent to the target field *
           ************************************************************/
          4 UDTTTYPE CHAR(2),        /* Target data type value       */

          4 UDTTBYTI CHAR(1),
             /********************************************************
              * Target bytes indicator (DXT only).                   *
              * "N" - indicates value resides with the definition.   *
              * "V" - indicates value is returned at "DEF" call to UIM*
              *********************************************************/

          4 FILL01B  CHAR(1),        /* Reserved for dxt use         */

          4 UDTTBYTV FIXED BIN(15),  /* Number of target bytes       */

          4 UDTTSCLI CHAR(1),
             /********************************************************
              * Target scale indicator (DXT only).                   *
              * "N" - indicates value resides with the definition.   *
              * "V" - indicates value is returned at "DEF" call to UIM*
              *********************************************************/

          4 FILL03B  CHAR(3),        /* Reserved for DXT use         */

          4 UDTTSCLV FIXED BIN(15),  /* Value of target scale        */
1     /****************************************************************
       * This section is the communications area between the exit and *
       * DPROP/DXT.                                                   *
       ****************************************************************/
       3 UDTXICOM,                /* Define a communications area
                                     (224 bytes)                      */
          4 UDTDPRP2 CHAR(24),       /* Reserved                     */

          4 UDTSCRT1 CHAR(128),
```

Figure 101 (Part 3 of 4). PL/I Interface Control Block for a Field Exit Routine

```
4 UDTENTRD CHAR(1),
   /*********************************************************
    * 'entered' flag - set to "X" by exit to indicate that  *
    * data type routine has been entered.                   *
    *********************************************************/

4 UDTINCTL CHAR(1),
   /*********************************************************
    * 'in-control' flag - set to "X" by exit to indicate    *
    * data type routine is in control.                      *
    *********************************************************/

4 UDTNULLT CHAR(1),
   /*********************************************************
    * "Y" - indicates data returned from exit is NULL.      *
    * "N" - indicates data returned from exit is NOT NULL   *
    *********************************************************/

4 FILL01C  CHAR(1),       /* Reserved                       */

4 UDTXRETC FIXED BIN(31),
   /*********************************************************
    * User exit return code:
    * 0 - successful completion else error encountered.     *
    * if caller is DPROP:                                   *
    *  4 ---> RUP will use its usual  error handling logic. *
    * >4 ---> RUP ABENDS.                                   *
    *********************************************************/

4 UDTXMESG CHAR(64);
   /*********************************************************
    * user exit message text inserted into DPROP/DXT        *
    * message if caller is DPROP, text will be inserted     *
    * into message EKYR970I/EYKR971E.                       *
    *********************************************************/
```

Figure 101 (Part 4 of 4). PL/I Interface Control Block for a Field Exit Routine

Sample Field Exit Control Block for C

Figure 102 shows an example of the EKYRCUDT control block in C. This control block, called EKYRCUDK, resides in the (EKYSAMP) library.

```
/******************** START OF CONTROL BLOCK SPECIFICATION ***********
 *                                                                  *
 * Control Block name:                                              *
 *    EKYRCUDK                                                      *
 *                                                                  *
 * Descriptive name:                                                *
 *    DPROP C language field exit interface.                        *
 *                                                                  *
 *    C language version of EKYRCUDT.                               *
 *                                                                  *
 ********************************************************************
 *                                                                  *
 *    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".          *
 *                                                                  *
 *    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                  *
 *    ALL RIGHTS RESERVED.                                          *
 *                                                                  *
 *    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                     *
 *    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                 *
 *    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                      *
 *                                                                  *
 *    LICENSED MATERIALS - PROPERTY OF IBM.                         *
 *                                                                  *
 ********************************************************************
 *                                                                  *
 * Status: V1 R2 M0                                                 *
 *                                                                  *
 * Function:                                                        *
 *    This is the C control block used to interface between         *
 *       - DPROP or DXT                                             *
 *         and                                                      *
 *       - a user's field exit routine (these user exit routines    *
 *         are called by DXT 'user data type exit routines')        *
 *                                                                  *
 *    There is one control block for each field exit routine,       *
 *    lasting for the duration of the exit in virtual storage.      *
 *    For synchronous propagation in MPP regions:                   *
 *    - this is the duration of the IMS program controller subtask. *
 *    For synchronous propagation in Batch/BMP regions, for         *
 *    asynchronous propagation, and for CCU processing:             *
 *    - this is the duration of the jobstep.                        *
 ********************************************************************
 *                                                                  *
 * Change activity:                                                 *
 *    None.                                                         *
 ******************** END OF CONTROL BLOCK SPECIFICATION *************/

#pragma page(1)

typedef
struct                     /* EKYRCUDT                             */
{
 /*******************************************************************
  * This section of the control block may not be modified by the exit.*
  *******************************************************************/
                              /* DXT prefix (32 bytes) udtpfx      */
  unsigned char udttname[8];     /* Name of block, "DVRXCUDT"      */
           char *udtxaddr;       /* Address of loaded routine      */
  unsigned char fill20[20];      /* Reserved for DXT use           */
```

Figure 102 (Part 1 of 4). C Interface Control Block for a Field Exit Routine

```
/**********************************************************************/
                             /* Prefix extension (300 bytes) udtpfxe */
                                /* (76 bytes) udtpnmod               */
 unsigned char udtcall[2];
          /**********************************************************
           * Type of call to exit:                                   *
           * "DF" - This is *** NOT *** issued by DPROP.  But is a   *
           * definition call issued by DXT-UIM for each datatype.    *
           * Exit can validate request and return required values.   *
           * "ST" - Source->target call issued by DXT and by DPROP   *
           * during HR mapping. Exit should convert data from the    *
           * DL/I IOarea format to DPROP supported target datatype.  *
           * "TS" - Target->source call issued by DPROP during RH    *
           * mapping.Exit should convert data from DPROP supported   *
           * datatype to DL/I IO area format not issued by DXT.      *
           **********************************************************/
 unsigned char fill02[2];       /* Reserved for DXT use              */
                                /* Environmental information
                                      (12 bytes) udtenvrn            */
 unsigned char udtopsys[4];
          /**********************************************************
           * Operating system calling program is executing in:       *
           * When UDTOPSYS = "MVS ".                                 *
           * This indicates DXT is running in MVS/370 environment.   *
           * When UDTOPSYS = "XA  ".                                 *
           * This indicates DXT is running in MVS/XA environment.    *
           * When UDTOPSYS = "ESA ".                                 *
           * This indicates DXT is running in MVS/ESA environment.   *
           **********************************************************/
 unsigned char udttrans[4];
          /**********************************************************
           * DB/DC environment: 'BAT ' if IMS Batch/BMP              *
           *                    'MPP ' if IMS MPP                    *
           *                    'IFP ' if Fast Path                  *
           *                    'CICS' if CICS                       *
           *                    '    ' if none of the above.         *
           **********************************************************/

#pragma page(1)

 unsigned char udtprogm[4];
          /**********************************************************
           * Calling program:   'DXT ' if DXT                        *
           *                    'DPRS' if DPROP synchronous PROP     *
           *                    'DPRA' if DPROP asynchronous PROP    *
           *                    'DPRC' if DPROP CCU processing       *
           *                    'DPRL' if DPROP DLU                  *
           **********************************************************/
 unsigned char udtexit[8];       /* Name of the user exit            */
          char *udtpcbls;        /* *** DXT only ***
                                    Address list of all PCB
                                    addresses if DLI environment     */
 unsigned char udtdprp1[24];     /* Additional work space            */
         /***********************************************************
          * This section contains data pertinent to the source field *
          ***********************************************************/
 unsigned char udtstype[2];      /* source data type value           */
 unsigned char udtsbyti;
```

Figure 102 (Part 2 of 4). C Interface Control Block for a Field Exit Routine

```
        /************************************************************
         * Source bytes indicator (DXT only).                       *
         * "N" - indicates value resides with the definition.       *
         * "V" - indicates value is returned at "DEF" call to UIM   *
         ************************************************************/
  unsigned char fill01a;         /* Reserved for DXT use            */
  short         udtsbytv;        /* number of source bytes          */
  unsigned char udtsscli;
        /************************************************************
         * Source scale indicator (DXT only).                       *
         * "N" - indicates value resides with the definition.       *
         * "V" - indicates value is returned at "DEF" call to UIM   *
         ************************************************************/
  unsigned char fill03a[3];      /* Reserved for DXT use            */
  short         udtssclv;        /* Value of source scale           */

        /************************************************************
         * This section contains data pertinent to the target field *
         ************************************************************/
  unsigned char udtttype[2];     /* Target data type value      */
  unsigned char udttbyti;
        /************************************************************
         * Target bytes indicator (DXT only).                       *
         * "N" - indicates value resides with the definition.       *
         * "V" - indicates value is returned at "DEF" call to UIM.  *
         ************************************************************/

  unsigned char fill01b;         /* Reserved for DXT use            */
  short         udttbytv;        /* Number of target bytes          */

#pragma page(1)

  unsigned char udttscli;
        /************************************************************
         * Target scale indicator (DXT only).                       *
         * "N" - indicates value resides with the definition.       *
         * "V" - indicates value is returned at "DEF" call to UIM.  *
         ************************************************************/
  unsigned char fill03b[3];      /* Reserved for DXT use            */
  short         udttsclv;        /* Value of target scale           */

    /****************************************************************
     * This section is the communications area between the exit and *
     * DPROP/DXT.                                                   *
     ****************************************************************/
                                 /* Define a communications area
                                    (224 bytes) udtxicom            */
  unsigned char udtdprp2[24];    /* Reserved                    */
  unsigned char udtscrt1[128];
  unsigned char udtentrd;
        /************************************************************
         * 'entered' flag - set to "X" by exit to indicate that     *
         * data type routine has been entered.                      *
         ************************************************************/
  unsigned char udtinctl;
        /************************************************************
         * 'in-control' flag - set to "X" by exit to indicate       *
         * data type routine is in control.                         *
         ************************************************************/
```

Figure 102 (Part 3 of 4). C Interface Control Block for a Field Exit Routine

```
  unsigned char udtnullt;
          /************************************************************
           * "Y" - indicates data returned from exit is NULL.        *
           * "N" - indicates data returned from exit is NOT NULL     *
           ************************************************************/
  unsigned char fill01c;          /* Reserved                       */
  long          udtxretc;
          /************************************************************
           * User exit return code:                                  *
           * 0 - successful completion else error encountered.       *
           * if caller is DPROP:                                     *
           *  4 ---> RUP will use its usual  error handling logic.   *
           * >4 ---> RUP ABENDS.                                     *
           ************************************************************/
  unsigned char udtxmesg[64];
          /************************************************************
           * user exit message text inserted into DPROP/DXT          *
           * message if caller is DPROP, text will be inserted       *
           * into message EKYR970I/EYKR971E.                         *
           ************************************************************/
 } EKYRCUDT;

#pragma page(1)
```

Figure 102 (Part 4 of 4). C Interface Control Block for a Field Exit Routine

Appendix D. Sample Propagation Exit Control Blocks

This appendix contains sample Propagation exit control blocks which map the existing DPROP interface control blocks. This appendix provides the exit control blocks in three languages:

- COBOL
- PL/I
- C

Chapter 4, "Propagation Exit Routines" on page 153 shows the Assembler version of the Propagation exit control blocks.

Sample Propagation Exit Control Blocks for COBOL

Figure 103 shows an example of the Propagation Exit control blocks in COBOL. These control blocks, called EKYRCPCC, EKYRCDLC, EKYHCHCC, and EKYHCQ2C reside in the DPROP Sample Source library (EKYSAMP).

COBOL Propagation Exit Interface (PIC)

Figure 103 shows a COBOL Propagation Exit Interface.

```
****************** START OF CONTROL BLOCK SPECIFICATION ************
*                                                                  *
*  CONTROL BLOCK NAME:                                             *
*    EKYRCPCC (PIC)                                                *
*                                                                  *
*  DESCRIPTIVE NAME:                                               *
*    DPROP COBOL PROPAGATION EXIT INTERFACE                        *
*                                                                  *
*    COBOL VERSION OF EKYRCPIC                                     *
*                                                                  *
********************************************************************
*                                                                  *
*    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".          *
*                                                                  *
*    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                  *
*    ALL RIGHTS RESERVED.                                          *
*                                                                  *
*    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                     *
*    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                 *
*    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                      *
*                                                                  *
*    LICENSED MATERIALS - PROPERTY OF IBM.                         *
*                                                                  *
********************************************************************
*                                                                  *
*  STATUS: V1 R2 M0                                                *
*                                                                  *
*  FUNCTION:                                                       *
*    THIS IS THE COBOL CONTROL BLOCK USED TO INTERFACE BETWEEN      *
*         - DPROP                                                  *
*         AND                                                      *
*         - A USER'S PROPAGATION  EXIT ROUTINE                     *
*                                                                  *
*    THERE IS ONE CONTROL BLOCK FOR EACH EXIT PROPAGATION          *
*    EXIT ROUTINE, LASTING FOR THE DURATION OF THE EXIT            *
*    IN VIRTUAL STORAGE.                                           *
*    FOR SYNCH PROPAGATION IN MPP REGIONS:                         *
*    - THIS IS THE DURATION OF THE IMS PROGRAM CONTROLLER          *
*      SUBTASK.                                                    *
*    FOR SYNCH PROPAGATION IN BATCH/BMP REGIONS, FOR               *
*    ASYNCH PROPAGATION, AND FOR CCU PROCESSING:                   *
*    - THIS IS THE DURATION OF THE JOBSTEP.                        *
*                                                                  *
*  CHANGE ACTIVITY:                                                *
*                                                                  *
****************** END OF CONTROL BLOCK SPECIFICATION **************
*
 01      EKYRCPCC.
*
*------------------------------------------------------------------*
*  THIS SECTION CONTAINS INFORMATION PROVIDED BY DPROP TO THE      *
```

Figure 103 (Part 1 of 5). COBOL Propagation Exit Interface

```
005200*  INVOKED EXIT AT ENTRY TO CALL. THIS SECTION MUST NOT BE      * 00520000
005300*  MODIFIED BY THE EXIT.                                        * 00530000
005400*---------------------------------------------------------------* 00540000
005500*                                                                 00550000
005600  02     PICEYE       PIC X(8).                                   00560000
005700*            EYE CATCHER                                          00570000
005800  02     PICEXIT      PIC X(8).                                   00580000
005900*            NAME OF THE EXIT ROUTINE                             00590000
006000  02     PICCALL      PIC XX.                                     00600000
006100*            TYPE OF CALL TO EXIT                                 00610000
006200*            ... HR = HIERARCHICAL TO RELATIONAL                  00620000
006300*            ... RH = RELATIONAL TO HIERARCHICAL                  00630000
006400  02     PICDBLEV     PIC X.                                      00640000
006500*            DEBUG LEVEL IN EFFECT                                00650000
006600*            HEX'02' : EXTERNAL TRACE OF PROPAGATING              00660000
006700*                      SQL STATEMENTS AND DL/I CALLS              00670000
006800  02     FILLER       PIC X.                                      00680000
006900*            RESERVED                                             00690000
007000  02     PICPTD       POINTER.                                    00700000
007100*            ADDRESS OF DPROP PTD                                 00710000
007200  02     PICPRID      PIC X(8).                                   00720000
007300*            PRID                                                 00730000
007400  02     PICPRSET     PIC X(8).                                   00740000
007500*            PRSET-ID                                             00750000
007600  02     PICPRTST     PIC X(26).                                  00760000
007700*            PR TIMESTAMP                                         00770000
007800  02     FILLER       PIC XX.                                     00780000
007900*            RESERVED                                             00790000
008000  02     PICPCBLA     PIC X(8).                                   00800000
008100*            PCB LABEL AS SPECIFIED ON PR                         00810000
008200  02     FILLER       PIC X(56).                                  00820000
008300*            RESERVED                                             00830000
008400  02     PICOPSYS     PIC X(4).                                   00840000
008500*            OPERATING SYSTEM                                     00850000
008600*            ...'ESA ': MVS/ESA                                   00860000
008700  02     PICTRANS     PIC X(4).                                   00870000
008800*            IMS REGION TYPE                                      00880000
008900*            ...'MPP ': MPP REGION                                00890000
009000*            ...'IFP ': IMS FAST PATH REGION                      00900000
009100*            ...'BMP ': IMS BMP REGION                            00910000
009200*            ...'BAT ': IMS BATCH REGION                          00920000
009300*            ...'    ': IF NONE OF ABOVE                          00930000
009400  02     PICPROGM     PIC X(4).                                   00940000
009500*            CALLING PROGRAM                                      00950000
009600*            ...'DPRS': DPROP SYNCH PROPAGATION                   00960000
009700*            ...'DPRA': DPROP ASYNCH PROPAGATION                  00970000
009800  02     FILLER       PIC X(12).                                  00980000
009900*            RESERVED FOR DPROP                                   00990000
010000*---------------------------------------------------------------* 01000000
010100*  THIS SECTION IS USED BY THE EXIT TO PROVIDE                  * 01010000
010200*  INFORMATION TO DPROP                                         * 01020000
```

Figure 103 (Part 2 of 5). COBOL Propagation Exit Interface

```
010300*--------------------------------------------------------------* 01030000
010400*                                                                 01040000
010500  02     PICENTRD     PIC X.                                      01050000
010600*            SET BY EXIT ROUTINE TO C'X', INDICATES               01060000
010700*            THAT EXIT HAS BEEN ENTERED                           01070000
010800  02     PICINCTL     PIC X.                                      01080000
010900*            SET BY EXIT ROUTINE TO C'X', INDICATES               01090000
011000*            THAT EXIT IS IN CONTROL                              01100000
011100*                                                                 01110000
011200***  RETURN CODE AND ERROR MESSAGE                                01120000
011300*                                                                 01130000
011400  02     PICXRETC     PIC S9(4)  COMP.                            01140000
011500*            RETURN CODE                                          01150000
011600*            ...4: SQL ERROR. SQL ERROR CODE IS IN THE FIELD      01160000
011700*                  SQLCODE OF THE SQLCA                           01170000
011800*            ...8: DLI ERROR. AIBRETRN, AIBREASN AND DL/I         01180000
011900*                  STATUS CODE IN PCB POINTED BY AIBRSA1          01190000
012000*            ..12: ERROR OTHER THAN SQL ERROR:                    01200000
012100*                  SOME RESOURCES NOT AVAILABLE                   01210000
012200*            ..16: ERROR OTHER THAN SQL ERROR:                    01220000
012300*                  NOT A RESOURCE AVAILABILITY PROBLEM.           01230000
012400*            ..20: SHOULD NOT OCCUR/SHOULD ABEND                  01240000
012500  02     PICXMESG.                                                01250000
012600*            USER EXIT ERROR/WARNING MESSAGE                      01260000
012700*            DPROP WILL WRITE THE MESSAGE TO VARIOUS              01270000
012800*            DESTINATIONS ACCORDING TO USUAL DPROP/RUP            01280000
012900*            ERROR HANDLING LOGIC.                                01290000
013000   03    PICXML1.                                                 01300000
013100*            1ST MESSAGE LINE                                     01310000
013200   04    PICXMSGI     PIC X(8).                                   01320000
013300*            ...8 BYTES MESSAGE ID                                01330000
013400    04   PICXMSGB     PIC X.                                      01340000
013500*            ...ONE BLANK                                         01350000
013600    04   PICXMTXT     PIC X(61).                                  01360000
013700*            ...61 TEXT BYTES IN 1ST MESSAGE LINE                 01370000
013800   03    PICXML2      PIC X(70).                                  01380000
013900*            2ND MESSAGE LINE                                     01390000
014000   03    PICXML3      PIC X(70).                                  01400000
014100*            3RD MESSAGE LINE                                     01410000
014200   03    PICXML4      PIC X(70).                                  01420000
014300*            4TH MESSAGE LINE                                     01430000
014400  02     FILLER       PIC X(12).                                  01440000
014500*            RESERVED FOR DPROP                                   01450000
014600*                                                                 01460000
014700***      NAME OF OBJECTS ASSOCIATED WITH ERROR                    01470000
014800*                                                                 01480000
014900  02     PICDBN       PIC X(8).                                   01490000
015000*            DBDNAME ASSOCIATED WITH THE ERROR                    01500000
015100  02     PICSEGN      PIC X(8).                                   01510000
015200*            SEG NAME ASSOCIATED WITH THE ERROR                   01520000
015300  02     PICTABQ      PIC X(8).                                   01530000
```

Figure 103 (Part 3 of 5). COBOL Propagation Exit Interface

```
*            TABLE NAME QUALIFIER ASSOC. W. ERROR
  02     PICTABN      PIC X(18).
*            TABLE NAME ASSOCIATED WITH THE ERROR
  02     FILLER       PIC X(14).
*            RESERVED FOR DPROP
*----------------------------------------------------------------*
*  EXIT WORK AREA                                                *
*                                                                *
*  THE EXIT WORK AREA CAN BE USED TO SAVE INFORMATION ACROSS     *
*  CALLS TO THE EXIT (E.G. TO SAVE THE ADDRESSES OF GETMAINED    *
*  AREAS ACROSS CALLS TO THE EXIT.                               *
*----------------------------------------------------------------*
*
  02     FILLER       PIC X(4).
*           4 BYTES FOR DOUBLE WORD ALIGNMENT (IN ASM: DS 0D)
  02     PICSWORK     PIC X(256).
*            WORK AREA FOR THE EXIT
  02     FILLER       PIC X(16).
*            RESERVED FOR DPROP
*----------------------------------------------------------------*
*  SQL COMMUNICATION AREA (SQLCA).                               *
*                                                                *
*  THIS SQLCA IS NOT USED BY COBOL EXITS                         *
*----------------------------------------------------------------*
*
  02     PICSQLCA     PIC X(136).
  02     FILLER       PIC X(16).
*
*----------------------------------------------------------------*
*  DLI APPLICATION INTERFACE BLOCK (AIB)                         *
*                                                                *
*  THE EXIT SHOULD USE THIS AIB FOR ITS DLI CALL. BEFORE FIRST   *
*  CALL, DPROP INITS AIBID, AIBLEN, AIBRSNM1 AND AIBSFUNC FIELDS*
*----------------------------------------------------------------*
*
  02     PICAIB.
*
   03    AIBID        PIC X(8).
*            EYECATCHER
   03    AIBLEN       PIC S9(8)  COMP.
*            DFSAIB ALLOCATED LENGTH
   03    AIBSFUNC     PIC X(8).
*            SUBFUNCTION CODE
   03    AIBRSNM1     PIC X(8).
*            RESOURCE NAME 1
   03    AIBRSNM2     PIC X(8).
*            RESOURCE NAME 2
   03    FILLER       PIC X(8).
*            RESERVED
   03    AIBOALEN     PIC S9(8)  COMP.
*            OUTPUT AREA LENGTH (MAX)
```

Figure 103 (Part 4 of 5). COBOL Propagation Exit Interface

```
020500   03    AIBOAUSE     PIC S9(8)  COMP.                              02050000
020600*            OUTPUT AREA LENGTH (USED)                              02060000
020700   03    FILLER       PIC X(12).                                    02070000
020800*            RESERVED                                               02080000
020900   03    AIBRETRN     PIC S9(8)  COMP.                              02090000
021000*            RETURN CODE                                            02100000
021100   03    AIBREASN     PIC S9(8)  COMP.                              02110000
021200*            REASON CODE                                            02120000
021300   03    FILLER       PIC X(4).                                     02130000
021400*            RESERVED                                               02140000
021500   03    AIBRSA1      POINTER.                                      02150000
021600*            RESOURCE ADDRESS 1                                     02160000
021700   03    AIBRSA2      POINTER.                                      02170000
021800*            RESOURCE ADDRESS 2                                     02180000
021900   03    AIBRSA3      POINTER.                                      02190000
022000*            RESOURCE ADDRESS 3                                     02200000
022100   03    FILLER       PIC X(40).                                    02210000
022200*            RESERVED                                               02220000
022300*                                                                   02230000
022400  02     FILLER       PIC X(16).                                    02240000
022500*                                                                   02250000
022600*---------------------------------------------------------------* 02260000
```

Figure 103 (Part 5 of 5). COBOL Propagation Exit Interface

COBOL DL/I Capture Interface (XPCB and XSDB)

Figure 104 shows a COBOL DL/I capture interface.

```
****************** START OF CONTROL BLOCK SPECIFICATION ***********
*                                                                 *
* CONTROL BLOCK NAME:                                             *
*    EKYRCDLC                                                     *
*                                                                 *
*  DESCRIPTIVE NAME:                                              *
*    DPROP RUP: COBOL DL/I CAPTURE INTERFACE                      *
*                                                                 *
*    COBOL VERSION OF EKYRCDL1                                    *
*                                                                 *
*******************************************************************
*                                                                 *
*    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".         *
*                                                                 *
*    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                 *
*    ALL RIGHTS RESERVED.                                         *
*                                                                 *
*    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                    *
*    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                *
*    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                     *
*                                                                 *
*    LICENSED MATERIALS - PROPERTY OF IBM.                        *
*                                                                 *
*******************************************************************
*                                                                 *
*  STATUS: V1 R2 M0                                               *
*                                                                 *
*  FUNCTION:                                                      *
*    EKYRCDLC IS A COPYAREA PROVIDING DESCRIPTIONS FOR THE        *
*    INTERFACE-AREAS USED TO COMMUNICATE BETWEEN                  *
*       - DL/I CHANGED DATA CAPTURE                               *
*       - THE EKYRUP00 DL/I CHANGED DATA EXIT ROUTINE             *
*    EKYRCDLC GENERATES DESCRIPTIONS OF FOLLOWING AREAS:          *
*       - THE DL/I XPCB                                           *
*       - THE DL/I XSDB                                           *
*       - THE DL/I DBPCB                                          *
*                                                                 *
*  CHANGE ACTIVITY:                                               *
*                                                                 *
****************** END OF CONTROL BLOCK SPECIFICATION *************
*
*-----------------------------------------------------------------*
*   E X T E N D E D   D A T A  B A S E   P C B  --  X P C B       *
*-----------------------------------------------------------------*
*
 01      XPCB.
*
  02     XPCBEYE      PIC X(4).
*            "XPCB" EYECATCHER
  02     XPCBVER      PIC XX.
*            XPCB VERSION INDICATOR
```

Figure 104 (Part 1 of 4). COBOL DL/I Capture Interface

```
005200  02     XPCBREL      PIC XX.                                    00520000
005300*            XPCB RELEASE INDICATOR                              00530000
005400  02     XPCBEXIT     PIC X(8).                                  00540000
005500*            SEGMENT USER EXIT NAME                              00550000
005600  02     XPCBRC       PIC S9(4)  COMP.                           00560000
005700*            RETURN-CODE                                         00570000
005800  02     XPCBRSNC     PIC S9(4)  COMP.                           00580000
005900*            REASON-CODE                                         00590000
006000  02     XPCBDBD      PIC X(8).                                  00600000
006100*            PHYSICAL DATA BASE NAME                             00610000
006200  02     XPCBVERA     POINTER.                                   00620000
006300*            ADDRESS OF DBD VERSION ID                           00630000
006400  02     XPCBSEG      PIC X(8).                                  00640000
006500*            PHYSICAL SEGMENT NAME                               00650000
006600  02     XPCBCALL     PIC X(4).                                  00660000
006700*            "CALL FUNCTION" DEFINED BY IMS/ESA                  00670000
006800*            ISRT: INSERT                                        00680000
006900*            REPL: REPLACE                                       00690000
007000*            DLET: DELETE                                        00700000
007100*            CASC: CASCADING DELETE                              00710000
007200*            DLLP: NOW ALSO DELETED FROM LOGICAL PATH            00720000
007300  02     XPCBPCALL    PIC X(4).                                  00730000
007400*            "PHYSICAL UPDATE TYPE" DEFINED BY IMS               00740000
007500*            ISRT: INSERT                                        00750000
007600*            REIN: RE-INSERT VIA LOGICAL PATH                    00760000
007700*            REPL: REPLACE                                       00770000
007800*            DLET: DELETE                                        00780000
007900*            DLPP: DELETED ONLY FROM PHYSICAL PATH               00790000
008000  02     FILLED       PIC X(4).                                  00800000
008100*            RESERVED                                            00810000
008200  02     XPCBPCBA     POINTER.                                   00820000
008300*            ADDRESS OF DB PCB                                   00830000
008400  02     XPCBPCBN     PIC X(8).                                  00840000
008500*            NAME OF DB PCB                                      00850000
008600  02     XPCBINQA     POINTER.                                   00860000
008700*            ADDRESS OF "INQY" OUTPUT                            00870000
008800  02     XPCBIOPA     POINTER.                                   00880000
008900*            ADDRESS OF I/O PCB                                  00890000
009000  02     FILLER       PIC S9(4)  COMP.                           00900000
009100*            RESERVED                                            00910000
009200  02     XPCBCKEYL    PIC S9(4)  COMP.                           00920000
009300*            LENGTH OF CONCATENATED KEY                          00930000
009400  02     XPCBCKEYA    POINTER.                                   00940000
009500*            ADDRESS OF CONCATENATED KEY                         00950000
009600  02     XPCBXSDBD    POINTER.                                   00960000
009700*            ADDRESS OF XSDB FOR DATA                            00970000
009800  02     XPCBXSDBB    POINTER.                                   00980000
009900*            ADDRESS OF XSDB FOR REPL DATA                       00990000
010000  02     XPCBXSDBP    POINTER.                                   01000000
010100*            ADDRESS OF XSDB FOR PATH DATA                       01010000
010200  02     FILLER       PIC X(12).                                 01020000
```

Figure 104 (Part 2 of 4). COBOL DL/I Capture Interface

```
*           RESERVED
  02     XPCBEXIWP    POINTER.
*           ADDRESS OF 256-BYTE AREA RESERVED FOR EXIT
  02     FILLER       PIC X(8).
*           RESERVED
  02     XPCBTIMST    PIC X(8).
*           TIMESTAMP OF CALL
  02     FILLER       PIC X(4).
*           RESERVED
*------------------------------------------------------------*
*  E X T E N D E D   S E G M E N T   D A T A -- X S D B     *
*------------------------------------------------------------*
*
 01      XSDB.
*
  02     XSDBEYE      PIC X(4).
*           "XSDB" EYECATCHER
  02     XSDBVER      PIC XX.
*           XSDB VERSION INDICATOR
  02     XSDBREL      PIC XX.
*           XSDB RELEASE INDICATOR
  02     XSDBNXSDB    POINTER.
*           NEXT XSDB POINTER
  02     XSDBDBD      PIC X(8).
*           PHYSICAL DATA BASE NAME
  02     XSDBSEG      PIC X(8).
*           PHYSICAL SEGMENT NAME
  02     XSDBPHP      PIC X.
*           PHYSICAL PATH ACCESSIBILITY
   88    XSDBPHPY     VALUE "Y".
*           ...SEGM ACCESSIBLE VIA PHYSICAL PATH
   88    XSDBPHPN     VALUE "N".
*           ...SEGM NOT ACCESSIBLE VIA PH. PATH
  02     FILLER       PIC X(3).
*
  02     XSDBSEGLV    PIC S9(4)  COMP.
*           SEGMENT DATA BASE LEVEL
  02     XSDBKEYL     PIC S9(4)  COMP.
*           LENGTH OF PHYSICAL KEY
  02     XSDBKEYA     POINTER.
*           ADDRESS OF PHYSICAL KEY
  02     FILLER       PIC XX.
*           RESERVED
  02     XSDBSEGL     PIC S9(4)  COMP.
*           LENGTH OF SEGMENT DATA
  02     XSDBSEGA     POINTER.
*           ADDRESS OF SEGMENT DATA
  02     FILLER       PIC X(12).
*
*------------------------------------------------------------*
* D A T A   B A S E   P C B                                  *
```

Figure 104 (Part 3 of 4). COBOL DL/I Capture Interface

```
      *---------------------------------------------------------------*
      *
       01      DBPCB.
      *
        02     DBPCBDBD     PIC X(8).
      *            DBD NAME
        02     DBPCBLEV     PIC XX.
      *            LEVEL FEEDBACK
        02     DBPCBSTC     PIC XX.
      *            STATUS CODES (RETURNED TO USER)
        02     DBPCBPRO     PIC X(4).
      *            PROCESSING OPTIONS
        02     DBPCBPFX     PIC S9(8)  COMP.
      *            PREFIX ADDRESS
        02     DBPCBSFD     PIC X(8).
      *            SEGMENT NAME FEEDBACK
        02     DBPCBMKL     PIC S9(8)  COMP.
      *            CURRRENT LENGTH OF KFBA OR GSAM FEEDBACK AREA
        02     DBPCBNSS     PIC S9(8)  COMP.
      *            NO OF SENSITIVE SEGMENTS IN PCB
      *
        02     DBPCBKFD     PIC X.
      *            KEY FEEDBACK AREA (MAY BE 256 BYTES LONG)
      *
      *---------------------------------------------------------------*
```

Figure 104 (Part 4 of 4). COBOL DL/I Capture Interface

COBOL HUP Exit Communication Block (HEC)

Figure 105 shows a COBOL HUP exit communication block.

```
**************** START OF CONTROL BLOCK SPECIFICATION ***********
*                                                                *
*  CONTROL BLOCK NAME:                                           *
*    EKYHCHCC (HEC)                                              *
*                                                                *
*  DESCRIPTIVE NAME:                                             *
*    DPROP COBOL HUP EXIT COMMUNICATION BLOCK                    *
*                                                                *
*    COBOL VERSION OF EKYHCHEC                                   *
*                                                                *
******************************************************************
*                                                                *
*    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".        *
*                                                                *
*    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                *
*    ALL RIGHTS RESERVED.                                        *
*                                                                *
*    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                   *
*    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY               *
*    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                    *
*                                                                *
*    LICENSED MATERIALS - PROPERTY OF IBM.                       *
*                                                                *
******************************************************************
*                                                                *
*  STATUS: V1 R2 M0                                              *
*                                                                *
*  FUNCTION:                                                     *
*    THIS IS THE COBOL CONTROL BLOCK USED TO PASS INFORMATION    *
*    GOT BY DPROP FROM THE DB2 CHANGED DATA CAPTURE EXIT         *
*    (USING IFI CALLS) TO THE PROPAGATION EXIT ROUTINE.          *
*                                                                *
*    THE HEC IS NEWLY BUILD FOR EACH EXIT CALL AND DOES          *
*    CONTAIN DATA TO BE RETAINED BEETWEEN EXIT CALLS.            *
*                                                                *
*  CHANGE ACTIVITY:                                              *
*                                                                *
**************** END OF CONTROL BLOCK SPECIFICATION *************
*
 01      HEC.
*
  02     HECEYE       PIC X(8).
*            EYE CATCHER
  02     FILLER       PIC X(8).
*            RESERVED
*
*------ POINTERS TO IFI HEADER AREAS
*
  02     HECQWHS      POINTER.
*            ADDRESS OF THE DB2 IFI STANDARD HEADER AREA
  02     HECQWHC      POINTER.
```

Figure 105 (Part 1 of 2). COBOL HUP Exit Communication Block

```
005200*            ADDRESS OF THE DB2 IFI CORRELATION DATA AREA        00520000
005300*                                                                00530000
005400*------ POINTERS TO CDC DATA AREAS                               00540000
005500*                                                                00550000
005600  02     HECCDCDD     POINTER.                                   00560000
005700*            ADDRESS OF CDC DATA DESCRIPT.                       00570000
005800  02     HECCDCDA     POINTER.                                   00580000
005900*            ADDRESS OF CDC DATA ROW: ONLY DATA FOR ISRT/DLET    00590000
006000*            OR CONTAINS THE AFTER IMAGE FOR UPDATE OPERATIONS   00600000
006100  02     HECCDCDB     POINTER.                                   00610000
006200*            ADDRESS OF CDC DATA ROW. ZERO FOR ISRT AND DLET     00620000
006300*            OR BEFORE IMAGE OF ROW FOR UPDATE OPERATIONS        00630000
006400*                                                                00640000
006500*------ RETURN CODE FROM IFI CALL                                00650000
006600*                                                                00660000
006700  02     HECRARC2     PIC S9(8)  COMP.                           00670000
006800*            IFCRC2 REASON CODE                                  00680000
006900*                                                                00690000
007000*------ DBDNAME/SEGNAME/PCBLABEL AREA (MAPPED BY HECDSLDS BELOW)  00700000
007100*                                                                00710000
007200  02     HECDBSLA     POINTER.                                   00720000
007300*            ADDRESS OF DBD/SEG/PCBLABEL AREA (HECDSLDS)         00730000
007400  02     HECDBSLN     PIC S9(8)  COMP.                           00740000
007500*            NUMBER OF ENTRIES IN HECDSLDS                       00750000
007600*                                                                00760000
007700*------ RESERVED SPACE AND CB SIZE                               00770000
007800*                                                                00780000
007900  02     HECRESV2     PIC X(16).                                 00790000
008000*                                                                00800000
008100*--------------------------------------------------------------* 00810000
008200*  FOR PROPAGATION EXIT ROUTINES ONLY, THE HECDBSLA FIELD       * 00820000
008300*  POINTS TO AN AREA. THIS AREA CONTAINS 24 BYTE ENTRIES        * 00830000
008400*  (IN TOP TO BOTTOM HIERARCHY) WHICH WAS DEFINED TO DPROP      * 00840000
008500*  FOR THE PR IN PROCESS. THE NUMBER OF ENTRIES IN THIS LIST    * 00850000
008600*  IS CONTAINED IN THE HECDBSLN FIELD.                          * 00860000
008700*--------------------------------------------------------------* 00870000
008800*                                                                00880000
008900 01      HECDSLDS.                                               00890000
009000*            ENTRY FOR DBD/SEG/PCBLABEL                          00900000
009100  02     HECDSELM     OCCURS 1.                                  00910000
009200*                                                                00920000
009300   03    HECDBDNM     PIC X(8).                                  00930000
009400*            DBD NAME                                            00940000
009500   03    HECSEGNM     PIC X(8).                                  00950000
009600*            SEGMENT NAME                                        00960000
009700   03    HECPCBNM     PIC X(8).                                  00970000
009800*            PCB LABEL NAME                                      00980000
009900*--------------------------------------------------------------* 00990000
```

Figure 105 (Part 2 of 2). COBOL HUP Exit Communication Block

COBOL IFC Copyarea for IFCIDS 0185

Figure 106 shows a COBOL IFC copyarea for IFCIDS 0185.

```
***************** START OF CONTROL BLOCK SPECIFICATION ***********
*                                                                *
*  CONTROL BLOCK NAME:                                           *
*    EKYHCQ2C                                                    *
*                                                                *
*  DESCRIPTIVE NAME:                                             *
*    DPROP COBOL IFC COPYAREA FOR IFCIDS 0185                    *
*                                                                *
*    COBOL VERSION OF A PORTION OF ASM MACRO DSNDQW02            *
*                                                                *
******************************************************************
*                                                                *
*    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".        *
*                                                                *
*    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                *
*    ALL RIGHTS RESERVED.                                        *
*                                                                *
*    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                   *
*    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY               *
*    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                    *
*                                                                *
*    LICENSED MATERIALS - PROPERTY OF IBM.                       *
*                                                                *
******************************************************************
*                                                                *
*  STATUS: V1 R2 M0                                              *
*                                                                *
*  FUNCTION:                                                     *
*    COPYAREA FOR ICF EVENTS.                                    *
*                                                                *
*    QW0185 IS WRITTEN FOR READS REQUESTS FOR IFCID 185.         *
*    IT CONTAINS A HEADER SECTION WHICH IS FOLLOWED BY           *
*    A DATA SECTION.                                             *
*                                                                *
*    THE DATA PORTION OF QW0185 BEGINS WITH FIELD:               *
*         - QW0185DD, IF QW0185TP=S                              *
*         OR                                                     *
*         - QW0185DR, IF QW0185TP=D                              *
*                                                                *
*                                                                *
*    IF QW0185TP = S, THE DATA PORTION CONSISTS OF FOUR          *
*       FIELDS FOLLOWED BY AN ARBITRARY NUMBER OF OCCURRENCES    *
*       OF THE QW0185VR STRUCTURE.                               *
*                                                                *
*    IF QW0185TP = D, THE DATA PORTION CONSISTS OF:              *
*       ---> THE DATA ROW, IF QW0185RC = 0                       *
*     OR                                                         *
*       ---> AN ERROR MESSAGE, OTHERWISE.                        *
*                                                                *
***************** END OF CONTROL BLOCK SPECIFICATION *************
*
```

Figure 106 (Part 1 of 4). COBOL IFC Copyarea for IFCIDS 0185

```
 01      QW0185       PIC X.
*
*---------------------------------------------------------------*
*                                                               *
*  QW0185A IS THE STRUCTURE CONTAINING THE TABLE DESCRIPTION    *
*  IN ITS DATA PORTION (QW0185TP=S).                            *
*                                                               *
*  THE DATA PORTION (QW0185DD) CONSISTS OF 4 FIELDS FOLLOWED    *
*  AN ARBITRARY NUMBER OF OCCURRENCES OF THE QW0185VR STRUCTURE *
*                                                               *
*---------------------------------------------------------------*
*
 01      QW0185A  REDEFINES  QW0185.
*
  02     QW0185LN     PIC S9(8)  COMP.
*            LENGTH OF TOTAL DB2CDC DATA
  02     QW0185TP     PIC X.
*            TYPE OF CONTROL BLOCK
   88    QW0185DS                     VALUE "S".
*              DB2CDC TABLE DESCRIPTION
   88    QW0185DO                     VALUE "D".
*              DB2CDC DATA ROW
  02     FILLER       PIC XXX.
*            RESERVED
  02     QW0185RC     PIC X(4).
*            REASON CODE DESCRIBING ERROR FOR THIS DATA PORTION
  02     QW0185QT.
*            QUALIFIED TABLE NAME
   03    QW0185CR     PIC X(8).
*            CREATOR OF TABLE (AUTH ID)
   03    QW0185TB     PIC X(18).
*            TABLE NAME
  02     QW0185TS     PIC X(10).
*            TIMESTAMP OF TABLE DESCRIPTION FROM CATALOG
  02     QW0185TL     PIC X(10).
*            TIMESTAMP OF LOG BUFFER CI WHEN IT IS EXTERNALIZED
*              OR WHEN THE BUFFER IS INITIALIZED
  02     QW0185UR     PIC X(8).
*            RBA OF THE FIRST LOG RECORD FOR THIS UNIT OF WORK.
  02     QW0185LR     PIC X(8).
*            RBA OF LOG RECORD FROM WHICH THIS DB2CDC DATA ROW
  02     FILLER       PIC XX.
*            OPERATION CODE, NOT USED WHEN QW0185TP=S.
  02     QW0185RI     PIC XX.
*            OPERATION CODE QUALIFIER.
   88    QW0185RE                     VALUE "RI".
*            RESULT OF A REFERENTIAL CONSTRAINT ENFORCEMENT OF
*              A DELETE SET NULL OR CASCADE, IF QW0185TP = "D".
  02     FILLER       PIC X(6).
*            RESERVED
*
```

Figure 106 (Part 2 of 4). COBOL IFC Copyarea for IFCIDS 0185

```
*-----------------------  DATA PORTION  ------------------------*
*
 02     QW0185DD.
*
  03    QW0185ID     PIC X(8).
*           EYE CATCHER = "CDCDD   "
  03    QW0185BC     PIC S9(8)  COMP.
*           LENGTH OF THE QW0185DD SECTION
  03    QW0185NO     PIC S9(4)  COMP.
*           TOTAL NUMBER OF OCCURRENCES OF QW0185VR
  03    QW0185LD     PIC S9(4)  COMP.
*           NUMBER OF COLS DESCRIBED BY OCCURRENCES OF QW0185VR
  03    QW0185VR                  OCCURS 1.
*           DESCRIBES A COLUMN IN A CAPTURED TABLE
   04   QW0185ST     PIC S9(4)  COMP.
*           TELLS THE DATA TYPE OF THE COLUMN AND WHETHER
*             IT HAS AN ASSOCIATED INDICATOR VARIABLE
   04   QW0185LE     PIC S9(4)  COMP.
*           DEFINES THE EXTERNAL LG OF A VALUE FROM THE COLUMN
   04   QW0185SD     PIC S9(8)  COMP.
*           CONTAINS THE CCSID
   04   QW0185SI     PIC S9(8)  COMP.
*           OFFSET OF THIS COLUMN INTO THE DATA ROW
   04   FILLER  REDEFINES  QW0185SI.
    05  FILLER       PIC XX.
    05  QW0185SX     PIC S9(4)  COMP.
*           OFFSET IS IN A HALF WORD
   04   QW0185SN.
*           LENGTH OF NAME AND NAME OF THE COLUMN
    05  QW0185NL     PIC S9(4)  COMP.
*           LENGTH OF COLUMN NAME
    05  QW0185CN     PIC X(30).
*           NAME OF COLUMN
*
*---------------------------------------------------------------*
*                                                               *
*  QW0185B IS THE STRUCTURE CONTAINING THE DATA ROW OR ERROR    *
*  MESSAGE IN ITS DATA PORTION (QW0185TP=D).                    *
*                                                               *
*  THE DATA PORTION (QW0185DR) CONSISTS OF:                     *
*      - THE DATA ROW, IF QW0185RC = 0                          *
*    OR                                                         *
*      - AN ERROR MESSAGE, OTHERWISE.                           *
*                                                               *
*---------------------------------------------------------------*
*
01      QW0185B  REDEFINES  QW0185.
*
 02     FILLER       PIC X(74).
*           REDEFINITION OF 74 BYTES OF THE HEADER PORTION
 02     QW0185PC     PIC XX.
```

Figure 106 (Part 3 of 4). COBOL IFC Copyarea for IFCIDS 0185

```
015400*           OPERATION CODE, USED WHEN QW0185TP=D.                01540000
015500*           IT HAS ONE OF THE FOLLOWING VALUES:                  01550000
015600   88    QW0185IN                     VALUE "IN".                01560000
015700*             INSERT                                             01570000
015800   88    QW0185UB                     VALUE "UB".                01580000
015900*             UPDATE BEFORE IMAGE                                01590000
016000   88    QW0185UA                     VALUE "UA".                01600000
016100*             UPDATE AFTER IMAGE                                 01610000
016200   88    QW0185DE                     VALUE "DE".                01620000
016300*             DELETE                                             01630000
016400  02     FILLER       PIC X(8).                                  01640000
016500*           REDEFINITION OF 8 BYTES MORE.                        01650000
016600*                                                                01660000
016700*-----------------------  DATA PORTION  ------------------------* 01670000
016800*                                                                01680000
016900  02   QW0185DR       PIC X        OCCURS 1.                     01690000
017000*           DATA ROW OR ERROR MESSAGE                            01700000
017100*                                                                01710000
017200*---------------------------------------------------------------* 01720000
```

Figure 106 (Part 4 of 4). COBOL IFC Copyarea for IFCIDS 0185

Sample Propagation Exit Control Blocks for PL/I

Figure 107 on page 398 shows an example of the Propagation Exit control block in PL/I. These control blocks, called EKYRCPCP, EKYRCDLP, EKYHCHCP, and EKYHCQ2P, reside in the (EKYSAMP) library.

PL/I Propagation Exit Interface (PIC)

Figure 107 on page 398 shows a PL/I Propagation exit interface.

```
1/******************** START OF CONTROL BLOCK SPECIFICATION ***********
  *                                                                   *
  * Control Block name:                                               *
  *    EKYRCPCP                                                       *
  *                                                                   *
  *    Descriptive name:                                              *
  *       DPROP PL/1 propagation exit interface.                      *
  *                                                                   *
  *       PL/1 version of EKYRCPIC.                                   *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  *        THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
  *                                                                   *
  *        5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
  *        ALL RIGHTS RESERVED.                                       *
  *                                                                   *
  *        U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
  *        USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
  *        GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
  *                                                                   *
  *        LICENSED MATERIALS - PROPERTY OF IBM.                      *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  * Status: V1 R2 M0                                                  *
  *                                                                   *
  * Function:                                                         *
  *    This is the PL/1 control block used to interface between       *
  *       - DPROP                                                     *
  *         and                                                       *
  *       - a user's propagation  exit routine                        *
  *                                                                   *
  *    There is one control block for each exit propagation exit      *
  *    routine, lasting for the duration of the exit in virtual       *
  *    storage.                                                       *
  *                                                                   *
  *    For synchronous propagation in MPP regions:                    *
  *       - this is the duration of the IMS program controller        *
  *         subtask.                                                  *
  *                                                                   *
  *    For synchronous propagation in batch/BMP regions, for          *
  *    asynchronous propagation, and for CCU processing:              *
  *       - this is the duration of the jobstep.                      *
  *                                                                   *
  * Change activity:                                                  *
  *    None                                                           *
  ******************** END OF CONTROL BLOCK SPECIFICATION *************/
1DECLARE EKYRCPCP_PTR POINTER;
 DECLARE 1 EKYRCPCP BASED(EKYRCPCP_PTR),
 /*********************************************************************
  * This section contains information provided by DPROP to the       *
  * invoked exit at entry to call.  This section MUST NOT be modified *
  * by the exit.                                                      *
  *********************************************************************/
    2 PICEYE   CHAR(8),        /* Eye catcher ("EKYRCPIC")            */
    2 PICEXIT  CHAR(8),        /* Name of the exit routine            */
    2 PICCALL  CHAR(2),        /* Type of call to exit:
                                     HR = hierarchical to relational
                                     RH = relational to hierarchical  */
```

Figure 107 (Part 1 of 3). PL/I Propagation Exit Interface

```
   2 PICDBLEV CHAR(1),        /* Debug level in effect.  Hex'02':
                                 external trace of propagating
                                 SQL statements and DL/I calls.      */

   2 FILL01   CHAR(1),        /* Reserved                            */
   2 PICPTD   POINTER,        /* Address of DPROP PTD                */
   2 PICPRID  CHAR(8),        /* PR-ID                               */
   2 PICPRSET CHAR(8),        /* PRSET-ID                            */
   2 PICPRTST CHAR(26),       /* PR timestamp                        */
   2 FILL02   CHAR(2),        /* Reserved                            */
   2 PICPCBLA CHAR(8),        /* PCB label as specified on PR        */
   2 FILL56   CHAR(56),       /* Reserved                            */
   2 PICOPSYS CHAR(4),        /* Operating system. 'ESA ': MVS/ESA   */

   2 PICTRANS CHAR(4),        /* IMS region type:
                                    'MPP ' = MPP region
                                    'IFP ' = IMS fast path region
                                    'BMP ' = IMS BMP region
                                    'BAT ' = IMS batch region
                                    '    ' = none of above           */

   2 PICPROGM CHAR(4),        /* calling program
                                  'DPRS' - DPROP synch propagation
                                  'DPRA' - DPROP asynch propagation  */

   2 FILL12A  CHAR(12),       /* Reserved for DPROP                  */
1  /*****************************************************************
    * This section is used by exit to provide information to DPROP   *
    *****************************************************************/
   2 PICENTRD CHAR(1),        /* Set by exit routine to 'X', to
                                 indicate that exit has been entered */
   2 PICINCTL CHAR(1),        /* Set by exit routine to 'X', to
                                 indicate that exit is in control    */

   /*****************************************************************
    *              Return code and error message                    *
    *****************************************************************/
   2 PICXRETC FIXED BIN(15),  /* Return code:
                                 4 = SQL error.  SQL error code is in
                                     the field SQLCODE of the SQLCA.
                                 8 = DLI error. AIBRETRN, AIBREASN
                                     and DL/I status code in PCB
                                     pointed by AIBRSA1.
                                12 = error other than SQL error,
                                     some resources not available.
                                16 = error other than SQL error, not
                                     a resource availability problem.
                                20 = should not occur/should abend.  */

   2 PICXMESG,                /* User exit error/warning message DPROP
                                 will write the message to various
                                 destinations according to the usual
                                 DPROP/RUP error handling logic.
                                 (280 byte area)                     */
      3 PICXML1,                 /* 1st message line (70 text bytes) */
         4 PICXMSGI CHAR(8),        /* 8 byte message ID             */
         4 PICXMSGB CHAR(1),        /* one blank                     */
         4 PICXMTXT CHAR(61),       /* 61 text bytes                 */

      3 PICXML2  CHAR(70),       /* 2nd message line (70 text bytes) */
```

Figure 107 (Part 2 of 3). PL/I Propagation Exit Interface

```
      3 PICXML3  CHAR(70),       /* 3rd message line (70 text bytes) */
      3 PICXML4  CHAR(70),       /* 4th message line (70 text bytes) */

   2 FILL12B  CHAR(12),       /* Reserved for DPROP               */

   /******************************************************************
    *       Names of objects associated with error                  *
    ******************************************************************/
   2 PICDBN   CHAR(8),        /* DBD name                         */
   2 PICSEGN  CHAR(8),        /* Segment name                     */
   2 PICTABQ  CHAR(8),        /* Table name qualifier             */
   2 PICTABN  CHAR(18),       /* Table name                       */
   2 FILL14   CHAR(14),       /* Reserved for DPROP               */
1  /******************************************************************
    *  Exit Work Area                                               *
    *     The exit work area can be used to save information across  *
    *     calls to the exit (e.g. to save the addresses of getmained *
    *     areas across calls to the exit).                          *
    ******************************************************************/
   2 FILLFLO  FLOAT BIN(0),   /* for double word alignment
                                 (in ASM: DS 0D)                  */
   2 PICSWORK CHAR(256),      /* Work area for the exit           */
   2 FILL16A  CHAR(16),       /* Reserved for DPROP               */

   /******************************************************************
    * SQL communication area (SQLCA).                               *
    ******************************************************************/
   2 PICSQLCA CHAR(136),
   2 FILL16B  CHAR(16),

   /******************************************************************
    * DLI application interface block (AIB).                        *
    * The exit should use this AIB for its DLI call. Before first    *
    * call, DPROP inits AIBID, AIBLEN, AIBRSNM1 and AIBSFUNC fields. *
    ******************************************************************/
   2 PICAIB,                  /*  AIB initialized by DPROP         */
      3 AIBID    CHAR(8),        /* Eyecatcher                    */
      3 AIBLEN   FIXED BIN(31),  /* DFSAIB ALLOCATED LENGTH       */
      3 AIBSFUNC CHAR(8),        /* Subfunction code              */
      3 AIBRSNM1 CHAR(8),        /* Resource name 1               */
      3 AIBRSNM2 CHAR(8),        /* Resource name 2               */
      3 FB1(2)   FIXED BIN(31),  /* Reserved                      */
      3 AIBOALEN FIXED BIN(31),  /* Output area length (max)      */
      3 AIBOAUSE FIXED BIN(31),  /* Output area length (used)     */
      3 FB2(2)   FIXED BIN(31),  /* Reserved                      */
      3 FIXB(2)  FIXED BIN(15),  /* Reserved                      */
      3 AIBRETRN FIXED BIN(31),  /* Return code                   */
      3 AIBREASN FIXED BIN(31),  /* Reason code                   */
      3 FB3      FIXED BIN(31),  /* Reserved                      */
      3 AIBRSA1  POINTER,        /* Resource address 1            */
      3 AIBRSA2  POINTER,        /* Resource address 2            */
      3 AIBRSA3  POINTER,        /* Resource address 3            */
      3 FB4(10)  FIXED BIN(31),  /* Reserved                      */

   2 FB5(4)   FIXED BIN(31);  /* Reserved                         */
```

Figure 107 (Part 3 of 3). PL/I Propagation Exit Interface

PL/I (RUP) DL/I Capture Interface

Figure 108 shows a PL/I DL/I capture interface.

```
1/******************** START OF CONTROL BLOCK SPECIFICATION ***********
  *                                                                   *
  * Control Block name:                                               *
  *    EKYRCDLP                                                       *
  *                                                                   *
  * Descriptive name:                                                 *
  *    DPROP RUP: PL/1 DL/I capture interface.                        *
  *                                                                   *
  *    PL/1 VERSION OF EKYRCDL1.                                      *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  *        THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
  *                                                                   *
  *        5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
  *        ALL RIGHTS RESERVED.                                       *
  *                                                                   *
  *        U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
  *        USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
  *        GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
  *                                                                   *
  *        LICENSED MATERIALS - PROPERTY OF IBM.                      *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  * STATUS: V1 R2 M0                                                  *
  *                                                                   *
  * FUNCTION:                                                         *
  *                                                                   *
  *    EKYRCDLP is an include library member providing descriptions   *
  *    for the interface-areas used to communicate between            *
  *       - DL/I changed data capture                                 *
  *       - the EKYRUP00 DL/I changed data exit routine               *
  *                                                                   *
  *    EKYRCDLP contains descriptions of following areas:             *
  *       - the DL/I XPCB                                             *
  *       - the DL/I XSDB                                             *
  *       - the DL/I DBPCB                                            *
  *                                                                   *
  * CHANGE ACTIVITY:                                                  *
  *    None                                                           *
  ******************** END OF CONTROL BLOCK SPECIFICATION *************/
1/*********************************************************************
  *     E x t e n d e d   D a t a   B a s e   P C B  --  X P C B      *
  *********************************************************************/
 DECLARE XPCB_POINTER POINTER;
 DECLARE 1 XPCB BASED (XPCB_POINTER),
    2 XPCBEYE   CHAR(4),       /* "XPCB" eyecatcher                   */
    2 XPCBVER   CHAR(2),       /* XPCB version indicator              */
    2 XPCBREL   CHAR(2),       /* XPCB release indicator              */
    2 XPCBEXIT  CHAR(8),       /* Segment user exit name              */
    2 XPCBRC    FIXED BIN(15), /* Return-code                         */
    2 XPCBRSNC  FIXED BIN(15), /* Reason-code                         */
    2 XPCBDBD   CHAR(8),       /* Physical Data Base name             */
    2 XPCBVERA  POINTER,       /* Address of DBD version ID           */
    2 XPCBSEG   CHAR(8),       /* Physical segment name               */
    2 XPCBCALL  CHAR(4),       /* "Call Function" defined by IMS/ESA
                                  ISRT: Insert
                                  REPL: Replace
                                  DLET: Delete
```

Figure 108 (Part 1 of 3). PL/I (RUP) DL/I Capture Interface

```
                              CASC: Cascading delete
                              DLLP: now also deleted from
                                    logical path              */
   2 XPCBPCALL CHAR(4),      /* "Physical Update Type" defined by IMS
                              ISRT: Insert
                              REIN: Re-insert via logical path
                              REPL: Replace
                              DLET: Delete
                              DLPP: Deleted only from
                                    physical path             */
   2 FILL      CHAR(4),      /* Reserved                      */
   2 XPCBPCBA  POINTER,      /* Address of DB PCB             */
   2 XPCBPCBN  CHAR(8),      /* Name of DB PCB                */
   2 XPCBINQA  POINTER,      /* Address of "INQY" output      */
   2 XPCBIOPA  POINTER,      /* Address of I/O PCB            */
   2 FILLER    FIXED BIN(15), /* Reserved                     */
   2 XPCBCKEYL FIXED BIN(15), /* Length of concatenated key   */
   2 XPCBCKEYA POINTER,      /* Address of concatenated key   */
   2 XPCBXSDBD POINTER,      /* Address of XSDB for data      */
   2 XPCBXSDBB POINTER,      /* Address of XSDB for REPL data */
   2 XPCBXSDBP POINTER,      /* Address of XSDB for path data */
   2 XPCBFIL1  FIXED BIN(31), /* Reserved                     */
   2 XPCBFIL2  FIXED BIN(31), /* Reserved                     */
   2 XPCBFIL3  FIXED BIN(31), /* Reserved                     */
   2 XPCBEXIWP POINTER,      /* Address of 256-byte area reserved
                              for exit                        */
   2 XPCBFIL4  FIXED BIN(31), /* Reserved                     */
   2 XPCBFIL5  FIXED BIN(31), /* Reserved                     */
   2 XPCBTIMST CHAR(8),      /* Timestamp of call             */
   2 XPCBFIL6  FIXED BIN(31); /* Reserved                     */
1/*******************************************************************
 *      E x t e n d e d   S e g m e n t   D a t a -- X S D B      *
 *******************************************************************/
 DECLARE XSDB_POINTER POINTER;
 DECLARE 1 XSDB BASED(XSDB_POINTER),
   2 XSDBEYE   CHAR(4),      /* "XSDB" eyecatcher             */
   2 XSDBVER   CHAR(2),      /* XSDB version indicator        */
   2 XSDBREL   CHAR(2),      /* XSDB release indicator        */
   2 XSDBNXSDB POINTER,      /* Next XSDB pointer             */
   2 XSDBDBD   CHAR(8),      /* Physical data base name       */
   2 XSDBSEG   CHAR(8),      /* Physical segment name         */
   2 XSDBPHP   CHAR(1),      /* Physical path accessibility
                              If value is "y" then segment is
                              accessible via physical path.
                              If value is "N" then segment is not
                              accessible via physical path.   */
   2 FILLER    CHAR(3),      /* Reserved                      */
   2 XSDBSEGLV FIXED BIN(15), /* Segment data base level      */
   2 XSDBKEYL  FIXED BIN(15), /* Length of physical key       */
   2 XSDBKEYA  POINTER,      /* Address of physical key       */
   2 XSDBFIL1  FIXED BIN(15), /* Reserved                     */
   2 XSDBSEGL  FIXED BIN(15), /* Length of segment data       */
   2 XSDBSEGA  POINTER,      /* Address of segment data       */
   2 XSDBFIL2  FIXED BIN(31), /* Reserved                     */
   2 XSDBFIL3  FIXED BIN(31), /* Reserved                     */
   2 XSDBFIL4  FIXED BIN(31); /* Reserved                     */
```

Figure 108 (Part 2 of 3). PL/I (RUP) DL/I Capture Interface

```
1/*********************************************************************
 *                 D a t a   B a s e   P C B                         *
 *********************************************************************/
 DECLARE 01 DBPCB,
    2 DBPCBDBD  CHAR(8),       /* DBD name                            */
    2 DBPCBLEV  CHAR(2),       /* Level feedback                      */
    2 DBPCBSTC  CHAR(2),       /* Status codes (returned to user)     */
    2 DBPCBPRO  CHAR(4),       /* Processing options                  */
    2 DBPCBPFX  FIXED BIN(31), /* Prefix address                      */
    2 DBPCBSFD  CHAR(8),       /* Segment name feedback               */
    2 DBPCBMKL  FIXED BIN(31), /* Currrent length of KFBA or
                                  GSAM feedback area                  */
    2 DBPCBNSS  FIXED BIN(31), /* Number of sensitive segments in
                                  the PCB                             */
    2 DBPCBKFD  CHAR(0);       /* Key feedback area                   */
```

Figure 108 (Part 3 of 3). PL/I (RUP) DL/I Capture Interface

PL/I HUP Exit Communication Block

Figure 109 shows a PL/I HUP exit communication block.

```
1/******************** START OF CONTROL BLOCK SPECIFICATION ***********
  *                                                                   *
  * Control Block name:                                               *
  *    EKYHCHCP (HEC)                                                 *
  *                                                                   *
  * Descriptive name:                                                 *
  *    DPROP PL/I HUP exit communication block.                       *
  *                                                                   *
  *    PL/I version of EKYHCHEC.                                      *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  *        THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
  *                                                                   *
  *        5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
  *        ALL RIGHTS RESERVED.                                       *
  *                                                                   *
  *        U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
  *        USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
  *        GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
  *                                                                   *
  *        LICENSED MATERIALS - PROPERTY OF IBM.                      *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  * Status: V1 R2 M0                                                  *
  *                                                                   *
  * Function:                                                         *
  *    This is the PL/I control block used to pass information        *
  *    received by DPROP from the DB2 changed data capture exit       *
  *    (using IFI calls) to the propagation exit routine.             *
  *                                                                   *
  *    The HEC is newly built for each exit call and does contain     *
  *    data to be retained between exit calls.                        *
  *********************************************************************
  *                                                                   *
  * Change activity:                                                  *
  *    None                                                           *
  ******************** END OF CONTROL BLOCK SPECIFICATION *************/
1DECLARE HEC_POINTER POINTER;
 DECLARE 1 HEC BASED(HEC_POINTER),
    2 HECEYE   CHAR(8),        /* Eye catcher ("EKY HEC ")           */
    2 HECRESV1 CHAR(8),        /* Reserved                           */

    /* Pointers to IFI header areas                                  */
    2 HECQWHS  POINTER,        /* Addr DB2 IFI standard header area  */
    2 HECQWHC  POINTER,        /* Addr DB2 IFI correlation data area */

    /* Pointers to CDC data areas                                    */
    2 HECCDCDD POINTER,        /* Address of CDC data description
                                  (always passed to exit)            */
    2 HECCDCDA POINTER,        /* Address of CDC data row:
                                  (always passed to exit)
                                     - only data for INSERT/DELETE
                                     - OR contains the after image
                                       for UPDATE operations         */
    2 HECCDCDB POINTER,        /* Address of CDC data row:
                                  - Zero for INSERT and DELETE
                                  - Otherwise, BEFORE  image of row
                                    for UPDATE operations.           */
```

Figure 109 (Part 1 of 2). PL/I HUP Exit Communication Block

```
   /* Return code from IFI call                                        */
   2 HECRARC2 FIXED BIN(31),  /* IFCRC2 reason code                    */

   /* DBDname/SEGname/PCBlabel area (mapped by HECDSLDS below)         */
   2 HECDBSLA POINTER,        /* Address of DBD/SEG/PCBlabel area
                                 HECDSLDS                              */
   2 HECDBSLN FIXED BIN(31),  /* Number of entries in HECDSLDS         */

   /* Reserved space                                                   */
   2 HECRESV2 CHAR(16);


/*********************************************************************
 * For propagation exit routines only, the HECDBSLA field points to  *
 * an area (for DB2 subexit routines this field is zero).  This area *
 * contains 24 byte entries (in top to bottom hierarchy) which were  *
 * defined to DPROP for the PR in process.  The number of entries in *
 * this list is contained in the HECDBSLN field.                     *
 *********************************************************************/
DECLARE HECDSLDS_PTR POINTER;
DECLARE 1 HECDSLDS(1) BASED(HECDSLDS_PTR),
                              /* Entry for DBD/SEG/PCB label           */
   2 HECDBDNM     CHAR(8),    /* DBD name                              */
   2 HECSEGNM     CHAR(8),    /* SEGMENT name                          */
   2 HECPCBNM     CHAR(8);    /* PCB label name                        */

/*********************************************************************/
```

Figure 109 (Part 2 of 2). PL/I HUP Exit Communication Block

PL/I IFC Copyarea for IFCIDS 0185

Figure 110 shows a PL/I IFC copyarea for IFCIDS 0185.

```
1/**************** START OF CONTROL BLOCK SPECIFICATION ***************
  *                                                                   *
  *  Control Block name:                                              *
  *    EKYHCQ2P                                                       *
  *                                                                   *
  *  Descriptive name:                                                *
  *    DPROP PL/1 IFC copyarea for IFCIDS 0185                        *
  *                                                                   *
  *    PL/1 VERSION OF A PORTION OF ASM MACRO DSNDQW02                *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  *    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".           *
  *                                                                   *
  *    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                   *
  *    ALL RIGHTS RESERVED.                                           *
  *                                                                   *
  *    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                      *
  *    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                  *
  *    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                       *
  *                                                                   *
  *    LICENSED MATERIALS - PROPERTY OF IBM.                          *
  *                                                                   *
  *********************************************************************
  *                                                                   *
  *  STATUS: V1 R2 M0                                                 *
  *                                                                   *
  *  FUNCTION:                                                        *
  *    Copyarea for IFC events.                                       *
  *                                                                   *
  *    QW0185 is written for reads requests for IFCID 185.            *
  *    It contains a header section which is followed by              *
  *    a data section.                                                *
  *                                                                   *
  *    The data portion of QW0185 begins with field:                  *
  *         - QW0185DD, if QW0185TP=S                                 *
  *         or                                                        *
  *         - QW0185DR, if QW0185TP=D                                 *
  *                                                                   *
  *                                                                   *
  *    If QW0185TP = S, the data portion consists of four             *
  *       fields followed by an arbitrary number of occurrences       *
  *       of the QW0185VR structure.                                  *
  *                                                                   *
  *    If QW0185TP = D, the data portion consists of:                 *
  *       ---> the data row, if QW0185RC = 0                          *
  *     or                                                            *
  *       ---> an error message, otherwise.                           *
  **************** END OF CONTROL BLOCK SPECIFICATION *****************/
1/********************************************************************
  *                                                                   *
  *  QW0185A is the structure containing the table description        *
  *  in its data portion (QW0185TP=S).                                *
  *                                                                   *
  *  The data portion (QW0185DD) consists of 4 fields followed        *
  *  an arbitrary number of occurrences of the QW0185VR structure     *
  *                                                                   *
  *********************************************************************/

 DECLARE HECCDCDD_PTR POINTER;
```

Figure 110 (Part 1 of 3). PL/I IFC Copyarea for IFCIDS 0185

```
DECLARE 1 QW0185A BASED(HECCDCDD_PTR),

   2 QW0185LN     FIXED BIN(31), /* Length of total db2cdc data      */
   2 QW0185TP     CHAR(1),       /* type of control block            *
                                  * S - DB2CDC table description     *
                                  * D - DB2CDC data row              */
   2 FILLER1      CHAR(3),       /* Reserved                         */
   2 QW0185RC     CHAR(4),       /* Reason code describing error     *
                                  * for this data portion            */
   2 QW0185QT,                   /* qualified table name             */
     3 QW0185CR   CHAR(8),       /* Creator of table (Auth id)       */
     3 QW0185TB   CHAR(18),      /* table name                       */
   2 QW0185TS     CHAR(10),      /* Timestamp of table               *
                                  * description from catalog         */
   2 QW0185TL     CHAR(10),      /* Timestamp of log buffer ci       *
                                  * when it is externalized or       *
                                  * when the buffer is initialized   */
   2 QW0185UR     CHAR(8),       /* RBA of the first log record      *
                                  * for this unit of work.           */
   2 QW0185LR     CHAR(8),       /* RBA of log record from           *
                                  * which this DB2CDC data row       */
   2 FILLER2      CHAR(2),       /* Operation code,                  *
                                  * not used when QW0185TP=S.        */
   2 QW0185RI     CHAR(2),       /* operation code qualifier.        *
                                  * RI - result of a referential     *
                                  *      constraint enforcement of   *
                                  *      a delete set null or        *
                                  *      cascade, if QW0185TP = "D". */
                                 /*                                  */
   2 FILLER3      CHAR(6),       /* Reserved                         */
1/************************  DATA PORTION  ****************************/

   2 QW0185DD,
     3 QW0185ID   CHAR(8),       /* Eye catcher = "CDCDD   "         */
     3 QW0185BC   FIXED BIN(31), /* Length of the QW0185DD section   */
     3 QW0185NO   FIXED BIN(15), /* Total number of occurrences      *
                                  * of QW0185VR                      */
     3 QW0185LD   FIXED BIN(15), /* Number of cols described by      *
                                  * occurrences of QW0185VR          */
     3 QW0185VR(1),              /* Describes a column in            *
                                  * a captured table                 */
      4 QW0185ST  FIXED BIN(15), /* Tells the data type of           *
                                  * the column and whether           *
                                  * it has an associated             *
                                  * indicator variable               */
      4 QW0185LE  FIXED BIN(15), /* Defines the external lg          *
                                  * of a value from the column       */
      4 QW0185SD  FIXED BIN(31), /* contains the CCSID               */
      4 QW0185SI,                /* offset of this column            *
                                  * into the data row                */
       5 FILLER   CHAR(2),
       5 QW0185SX FIXED BIN(15), /* Offset is in a half word         */
      4 QW0185SN,                /* Length of name and               *
                                  * name of the column               */
       5 QW0185NL FIXED BIN(15), /* Length of column name            */
       5 QW0185CN CHAR(30);      /* Name of column                   */
```

Figure 110 (Part 2 of 3). PL/I IFC Copyarea for IFCIDS 0185

```
1/*******************************************************************
  *                                                                 *
  *  QW0185B is the structure containing the data row or error      *
  *  message in its data portion (QW0185TP=D).                      *
  *                                                                 *
  *  The data portion (QW0185DR) consists of:                       *
  *      - the data row, if QW0185RC = 0                            *
  *    otherwise                                                    *
  *      - an error message.                                        *
  *                                                                 *
  *******************************************************************/

 DECLARE HECCDCDA_PTR POINTER;
 DECLARE 1 QW0185B BASED(HECCDCDA_PTR),
    2 FILLER5      CHAR(74),      /* Redefinition of 74 bytes        *
                                   * of the header portion           */
    2 QW0185PC     CHAR(2),       /* Operation code,                 *
                                   * used when QW0185TP=d.           *
                                   * IN - Insert                     *
                                   * UB - Update before image        *
                                   * UA - Update after image         *
                                   * DE - Delete                     */
    2 FILLER6      CHAR(8),



 /************************  DATA PORTION  ****************************/

    2 QW0185DR(1)  CHAR(1);       /* Data row or error message       */
```

Figure 110 (Part 3 of 3). PL/I IFC Copyarea for IFCIDS 0185

Sample Propagation Exit Control Blocks for C

Figure 111 on page 410 shows an example of the Propagation Exit control blocks in C. These control blocks, called EKYRCPCK, EKYRCDLK, EKYHCHCK, and EKYHCQ2K, reside in the (EKYSAMP) library.

C Propagation Exit Interface (PIC)

Figure 111 on page 410 shows a C Propagation exit interface.

```
/******************** START OF CONTROL BLOCK SPECIFICATION ***********
 *                                                                  *
 * Control Block name:                                              *
 *    EKYRCPCK                                                      *
 *                                                                  *
 *    Descriptive name:                                             *
 *       DPROP C propagation exit interface.                        *
 *                                                                  *
 *       C version of EKYRCPIC.                                     *
 *                                                                  *
 ********************************************************************
 *                                                                  *
 *        THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".      *
 *                                                                  *
 *        5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.              *
 *        ALL RIGHTS RESERVED.                                      *
 *                                                                  *
 *        U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                 *
 *        USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY             *
 *        GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                  *
 *                                                                  *
 *        LICENSED MATERIALS - PROPERTY OF IBM.                     *
 *                                                                  *
 ********************************************************************
 *                                                                  *
 * Status: V1 R2 M0                                                 *
 *                                                                  *
 * Function:                                                        *
 *    This is the PL/1 control block used to interface between      *
 *       - DPROP                                                    *
 *         and                                                      *
 *       - a user's propagation  exit routine                       *
 *                                                                  *
 *    There is one control block for each exit propagation exit     *
 *    routine, lasting for the duration of the exit in virtual      *
 *    storage.                                                      *
 *                                                                  *
 *    For synchronous propagation in MPP regions:                   *
 *       - this is the duration of the IMS program controller       *
 *         subtask.                                                 *
 *                                                                  *
 *    For synchronous propagation in batch/BMP regions, for         *
 *    asynchronous propagation, and for CCU processing:             *
 *       - this is the duration of the jobstep.                     *
 *                                                                  *
 * Change activity:                                                 *
 *    None                                                          *
 ******************** END OF CONTROL BLOCK SPECIFICATION *************/

#pragma page(1)

typedef struct                /* PICXMESG                           */
{                             /* User exit error/warning message DPROP
                                 will write the message to various
                                 destinations according to the usual
                                 DPROP/RUP error handling logic.
                                 (280 byte area)                    */
                              /* 1st message line (70 text bytes)   */
   unsigned char picxmsgi[8];  /* 8 byte message id                 */
   unsigned char picxmsgb;     /* one blank                         */
   unsigned char picxmtxt[61]; /* 61 text bytes                     */
} PICXML1;
```

Figure 111 (Part 1 of 4). C Propagation Exit Interface

```
typedef struct
{
   PICXML1       picxml1;
   unsigned char picxml2[70];  /* 2nd message line (70 text bytes)    */
   unsigned char picxml3[70];  /* 3rd message line (70 text bytes)    */
   unsigned char picxml4[70];  /* 4th message line (70 text bytes)    */
} PICXMESG;

#pragma page(1)

/*********************************************************************
 * DLI application interface block (AIB).                            *
 * The exit should use this AIB for its DLI call. Before first call, *
 * DPROP inits AIBID, AIBLEN, AIBRSNM1 and AIBSFUNC fields.          *
 *********************************************************************/
typedef struct                 /* PICAIB                             */
{                              /* AIB initialized by DPROP           */
   unsigned char aibid[8];     /* Eyecatcher                         */
   long          aiblen;       /* DFSAIB allocated length            */
   unsigned char aibsfunc[8];  /* Subfunction code                   */
   unsigned char aibrsnm1[8];  /* Resource name 1                    */
   unsigned char aibrsnm2[8];  /* Resource name 2                    */
   long          fb1[2];       /* Reserved                           */
   long          aiboalen;     /* Output area length (max)           */
   long          aiboause;     /* Output area length (used)          */
   long          fb2[2];       /* Reserved                           */
   short         fixb[2];      /* Reserved                           */
   long          aibretrn;     /* Return code                        */
   long          aibreasn;     /* Reason code                        */
   long          fb3;          /* Reserved                           */
            char *aibrsa1;     /* Resource address 1                 */
            char *aibrsa2;     /* Resource address 2                 */
            char *aibrsa3;     /* Resource address 3                 */
   long          fb4[10];      /* Reserved                           */
} PICAIB;

#pragma page(1)

typedef struct
{                              /* EKYRCPIC                           */
 /********************************************************************
  * This section contains information provided by DPROP to the       *
  * invoked exit at entry to call.  This section MUST NOT be modified *
  * by the exit.                                                     *
  ********************************************************************/
    unsigned char piceye[8];   /* Eye catcher ("EKYRCPIC")           */
    unsigned char picexit[8];  /* Name of the exit routine           */
    unsigned char piccall[2];  /* Type of call to exit:
                                     HR = hierarchical to relational
                                     RH = relational to hierarchical  */

    unsigned char picdblev;    /* Debug level in effect. hex'02':
                                  external trace of propagating
                                  SQL statements and DL/I calls.      */

    unsigned char fill01;      /* Reserved                           */
             char *picptd;     /* Address of DPROP PTD               */
    unsigned char picprid[8];  /* PR-ID                              */
    unsigned char picprset[8]; /* PRset-ID                           */
    unsigned char picprtst[26];/* PR timestamp                       */
```

Figure 111 (Part 2 of 4). C Propagation Exit Interface

```
    unsigned char fill02[2];   /* Reserved                              */
    unsigned char picpcbla[8]; /* PCB label as specified on PR          */
    unsigned char fill56[56];  /* Reserved                              */
    unsigned char picopsys[4]; /* Operating system. ESA : MVS/ESA       */

    unsigned char pictrans[4]; /* IMS region type:
                                    'MPP ' = MPP region
                                    'IFP ' = IMS fast path region
                                    'BMP ' = IMS BMP region
                                    'BAT ' = IMS batch region
                                    '    ' = none of above              */

    unsigned char picprogm[4]; /* Calling program:
                                   'DPRS' - DPROP synch propagation
                                   'DPRA' - DPROP asynch propagation */

    unsigned char fill12a[12]; /* Reserved for DPROP                    */

#pragma page(1)

    /*******************************************************************
     * This section is used by exit to provide information to DPROP   *
     *******************************************************************/
    unsigned char picentrd;    /* Set by exit routine to 'X', to
                                  indicate that exit has been entered */
    unsigned char picinctl;    /* Set by exit routine to 'X', to
                                  indicate that exit is in control    */

    /*******************************************************************
     *             Return code and error message                      *
     *******************************************************************/
    short         picxretc;    /* Return code:
                                  4 = SQL error.  SQL error code is in
                                      the field SQLCODE of the SQLCA.
                                  8 = DLI error. AIBRETRN, AIBREASN
                                      and DL/I status code in PCB
                                      pointed by AIBRSA1.
                                 12 = error other than SQL error,
                                      some resources not available.
                                 16 = error other than SQL error, not
                                      a resource availability problem.
                                 20 = should not occur/should abend.  */

    PICXMESG picxmesg;
    unsigned char fill12b[12]; /* Reserved for DPROP                    */

    /*******************************************************************
     *       Names of objects associated with error                   *
     *******************************************************************/
    unsigned char picdbn[8];   /* DBD name                              */
    unsigned char picsegn[8];  /* Segment name                          */
    unsigned char pictabq[8];  /* Table name qualifier                  */
    unsigned char pictabn[18]; /* Table name                            */
    unsigned char fill14[14];  /* Reserved for DPROP                    */

#pragma page(1)
```

Figure 111 (Part 3 of 4). C Propagation Exit Interface

```
   /****************************************************************
    *  Exit Work Area                                              *
    *     The exit work area can be used to save information across  *
    *     calls to the exit (e.g. to save the addresses of getmained *
    *     areas across calls to the exit).                         *
    *****************************************************************/
   unsigned char floater[4];
   unsigned char picswork[256];/* Work area for the exit            */
   unsigned char fill16a[16]; /* Reserved for DPROP                */

   /****************************************************************
    * SQL communication area (SQLCA).                              *
    *****************************************************************/
   struct sqlca  picsqlca;
   unsigned char fill16b[16];
   PICAIB        picaib;

   long          fb5[4];      /* Reserved                          */
 } EKYRCPIC;

#pragma page(1)
```

Figure 111 (Part 4 of 4). C Propagation Exit Interface

C (RUP) DL/I Capture Interface

Figure 112 on page 414 shows a C DL/I capture interface.

```
/******************** START OF CONTROL BLOCK SPECIFICATION ***********
 *                                                                  *
 * Control Block name:                                              *
 *    EKYRCDLK                                                      *
 *                                                                  *
 * Descriptive name:                                                *
 *    DPROP RUP: C DL/I capture interface.                          *
 *                                                                  *
 *    C VERSION OF EKYRCDL1.                                        *
 *                                                                  *
 ********************************************************************
 *                                                                  *
 *        THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".      *
 *                                                                  *
 *        5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.              *
 *        ALL RIGHTS RESERVED.                                      *
 *                                                                  *
 *        U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                 *
 *        USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY             *
 *        GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                  *
 *                                                                  *
 *        LICENSED MATERIALS - PROPERTY OF IBM.                     *
 *                                                                  *
 ********************************************************************
 *                                                                  *
 * STATUS: V1 R2 M0                                                 *
 *                                                                  *
 * FUNCTION:                                                        *
 *                                                                  *
 *    EKYRCDLI is a copyarea providing descriptions for the         *
 *    interface-areas used to communicate between                   *
 *       - DL/I changed data capture                                *
 *       - the EKYRUP00 DL/I changed data exit routine              *
 *                                                                  *
 *    EKYRCDLI generates descriptions of following areas:           *
 *       - the DL/I XPCB                                            *
 *       - the DL/I XSDB                                            *
 *       - the DL/I DBPCB                                           *
 *                                                                  *
 * CHANGE ACTIVITY:                                                 *
 *    None                                                          *
 ******************** END OF CONTROL BLOCK SPECIFICATION *************/

#pragma page(1)

 typedef struct
 {
    unsigned char segikey[6];
    unsigned char segidat1[7];
    unsigned char segidat2[4];
    unsigned char segidat3[8];
 } SEGI;

 /*********************************************************************
 *     E x t e n d e d   S e g m e n t   D a t a -- X S D B         *
 *********************************************************************/
 typedef struct             /* XSDB                                  */
 {
    unsigned char xsdbeye[4];  /* "XSDB" eyecatcher                  */
    unsigned char xsdbver[2];  /* XSDB version indicator             */
```

Figure 112 (Part 1 of 3). C (RUP) DL/I Capture Interface

```
   unsigned char xsdbrel[2];  /* XSDB release indicator             */
          char *xsdbnxsdb;  /* Next XSDB pointer                  */
   unsigned char xsdbdbd[8];  /* Physical data base name            */
   unsigned char xsdbseg[8];  /* Physical segment name              */
   unsigned char xsdbphp;     /* Physical path accessibility
                                 If value is "Y" then segment is
                                 accessible via physical path.
                                 If value is "N" then segment is not
                                 accessible via physical path.      */
   unsigned char filler[3];   /* Reserved                           */
   short         xsdbseglv;   /* Segment data base level            */
   short         xsdbkeyl;    /* Length of physical key             */
          char *xsdbkeya;   /* Address of physical key            */
   short         xsdbfil1;    /* Reserved                           */
   short         xsdbsegl;    /* Length of segment data             */
   SEGI          *xsdbsega;   /* Address of segment data            */
   long          xsdbfil2;    /* Reserved                           */
   long          xsdbfil3;    /* Reserved                           */
   long          xsdbfil4;    /* Reserved                           */
 } XSDB;

#pragma page(1)

 /*********************************************************************
  *    E x t e n d e d   D a t a  B a s e   P C B -- X P C B     *
  *********************************************************************/

 typedef struct            /* XPCB                                */
 {
   unsigned char xpcbeye[4];  /* "XPCB" eyecatcher                  */
   unsigned char xpcbver[2];  /* XPCB version indicator             */
   unsigned char xpcbrel[2];  /* XPCB release indicator             */
   unsigned char xpcbexit[8]; /* Segment user exit name             */
   short         xpcbrc;      /* Return-code                        */
   short         xpcbrsnc;    /* Reason-code                        */
   unsigned char xpcbdbd[8];  /* Physical data base name            */
          char *xpcbvera;   /* Address of DBD version ID          */
   unsigned char xpcbseg[8];  /* Physical segment name              */
   unsigned char xpcbcall[4];
                              /* "Call function" defined by IMS/ESA
                                 ISRT: Insert
                                 REPL: Replace
                                 DLET: Delete
                                 CASC: Cascading delete
                                 DLLP: now also deleted from
                                       logical path                 */
   unsigned char xpcbpcall[4];    /* "Physical update type" defined
                                    by IMS
                                 ISRT: Insert
                                 REIN: Re-insert via logical path
                                 REPL: Replace
                                 DLET: Delete
                                 DLPP: Deleted only from
                                       physical path                */
   unsigned char fill[4];     /* Reserved                           */
          char *xpcbpcba;   /* Address of DB PCB                  */
   unsigned char xpcbpcbn[8]; /* Name of DB PCB                     */
          char *xpcbinqa;   /* Address of "INQY" output           */
          char *xpcbiopa;   /* Address of I/O PCB                 */
   short         filler;      /* Reserved                           */
   short         xpcbckeyl;   /* Length of concatenated key         */
```

Figure 112 (Part 2 of 3). C (RUP) DL/I Capture Interface

```
          char *xpcbckeya;  /* Address of concatenated key        */
   XSDB        *xpcbxsdbd;  /* Address of XSDB for data           */
          char *xpcbxsdbb;  /* Address of XSDB for REPL data      */
          char *xpcbxsdbp;  /* Address of XSDB for path data      */
   long         xpcbfil1;   /* Reserved                           */
   long         xpcbfil2;   /* Reserved                           */
   long         xpcbfil3;   /* Reserved                           */
          char *xpcbexiwp;  /* Address of 256-byte area reserved
                               for exit                           */
   long         xpcbfil4;   /* Reserved                           */
   long         xpcbfil5;   /* Reserved                           */
   unsigned char xpcbtimst[8];/* Timestamp of call                */
   long         xpcbfil6;   /* Reserved                           */
} XPCB;

#pragma page(1)

/*********************************************************************
 *                  D a t a   B a s e   P C B                       *
 *********************************************************************/
typedef struct
{                          /* DBPCB                              */
   unsigned char DBPCBDBD[8]; /* DBD name                        */
   unsigned char DBPCBLEV[2]; /* Level feedback                  */
   unsigned char DBPCBSTC[2]; /* Status codes (returned to user) */
   unsigned char DBPCBPRO[4]; /* Processing options              */
   long         DBPCBPFX;   /* Prefix address                    */
   unsigned char DBPCBSFD[8]; /* Segment name feedback           */
   long         DBPCBMKL;   /* Currrent length of KFBA or
                               GSAM feedback area                 */
   long         DBPCBNSS;   /* Number of sensitive segments in
                               the PCB                            */
   unsigned char DBPCBKFD;  /* Key feedback area                  */
} DBPCB;

#pragma page(1)
```

Figure 112 (Part 3 of 3). C (RUP) DL/I Capture Interface

C HUP Exit Communication Block

Figure 113 on page 417 shows a C HUP exit communication block.

```
/******************** START OF CONTROL BLOCK SPECIFICATION ***********
 *                                                                   *
 * Control Block name:                                               *
 *    EKYHCHCK (HEC)                                                 *
 *                                                                   *
 * Descriptive name:                                                 *
 *    DPROP C HUP exit communication block.                          *
 *                                                                   *
 *    C version of EKYHCHEC.                                         *
 *                                                                   *
 *********************************************************************
 *                                                                   *
 *        THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".       *
 *                                                                   *
 *        5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.               *
 *        ALL RIGHTS RESERVED.                                       *
 *                                                                   *
 *        U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                  *
 *        USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY              *
 *        GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   *
 *                                                                   *
 *        LICENSED MATERIALS - PROPERTY OF IBM.                      *
 *                                                                   *
 *********************************************************************
 *                                                                   *
 * Status: V1 R2 M0                                                  *
 *                                                                   *
 * Function:                                                         *
 *    This is the PL/I control block used to pass information        *
 *    received by DPROP from the DB2 changed data capture exit       *
 *    (using IFI calls) to the propagation exit routine.             *
 *                                                                   *
 *    The HEC is newly built for each exit call and does contain     *
 *    data to be retained between exit calls.                        *
 *********************************************************************
 *                                                                   *
 * Change activity:                                                  *
 *    None                                                           *
 ******************** END OF CONTROL BLOCK SPECIFICATION *************/

#pragma page(1)

/*********************************************************************
 * For propagation exit routines only, the HECDBSLA field points to  *
 * an area (for DB2 subexit routines this field is zero).  This area *
 * contains 24 byte entries (in top to bottom hierarchy) which were  *
 * defined to DPROP for the PR in process.  The number of entries in *
 * this list is contained in the HECDBSLN field.                     *
 *********************************************************************/
typedef struct                /* HECDSLDS1                         */
{                             /* Entry for DBD/SEG/PCB label       */
   unsigned char hecdbdnm[8];   /* DBD name                        */
   unsigned char hecsegnm[8];   /* Segment name                    */
   unsigned char hecpcbnm[8];   /* PCB label name                  */
}  HECDSLDS1;

typedef struct                /* HECDSLDS1 array                   */
{
   HECDSLDS1 hecdslds[30];
}  HECDSLDS;
```

Figure 113 (Part 1 of 2). C HUP Exit Communication Block

```
typedef struct
{                               /* HEC                                  */
   unsigned char heceye[8];     /* Eye catcher ("EKY HEC ")             */
   unsigned char hecresv1[8];   /* Reserved                             */

   /* Pointers to IFI header areas                                      */
            char *hecqwhs;      /* Addr DB2 ifi standard header area    */
            char *hecqwhc;      /* Addr DB2 ifi correlation data area   */

   /* Pointers to CDC data areas                                        */
            void *heccdcdd;     /* Address of CDC data description
                                   (always passed to exit)              */
            void *heccdcda;     /* Address of CDC data row:
                                   (always passed to exit)
                                    - only data for INSERT/DELETE
                                    - OR contains the after image
                                      for UPDATE operations             */
            char *heccdcdb;     /* Address of CDC data row:
                                   - Zero for INSERT and DELETE
                                   - Otherwise, BEFORE  image of row
                                     for UPDATE operations.             */

   /* Return code from IFI call                                         */
   long          hecrarc2;      /* IFCRC2 reason code                   */

   /* DBDname/SEGname/PCBlabel area (mapped by HECDSLDS below)          */
   HECDSLDS      *hecdbsla;     /* Address of DBD/SEG/PCBlabel area
                                   HECDSLDS                             */
   long           hecdbsln;     /* Number of entries in hecdslds        */

   /* Reserved space                                                    */
   unsigned char  hecresv2[16]; /* reserved                             */
} HEC;

/*********************************************************************/
#pragma page(1)
```

Figure 113 (Part 2 of 2). C HUP Exit Communication Block

C IFC Copyarea for IFCIDS 0185

Figure 114 on page 419 shows a C IFC copyarea for IFCIDS 0185.

```
/***************** START OF CONTROL BLOCK SPECIFICATION ***************
 *                                                                    *
 *  Control Block name:                                               *
 *    EKYHCQ2K                                                        *
 *                                                                    *
 *  Descriptive name:                                                 *
 *    DPROP C IFC copyarea for IFCIDS 0185                            *
 *                                                                    *
 *    C VERSION OF A PORTION OF ASM MACRO DSNDQW02                    *
 *                                                                    *
 **********************************************************************
 *                                                                    *
 *    THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM".            *
 *                                                                    *
 *    5685-124 (C) COPYRIGHT IBM CORP. 1989, 1992.                    *
 *    ALL RIGHTS RESERVED.                                            *
 *                                                                    *
 *    U.S. GOVERNMENT USERS RESTRICTED RIGHTS -                       *
 *    USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY                   *
 *    GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                        *
 *                                                                    *
 *    LICENSED MATERIALS - PROPERTY OF IBM.                           *
 *                                                                    *
 **********************************************************************
 *                                                                    *
 *  STATUS: V1 R2 M0                                                  *
 *                                                                    *
 *  FUNCTION:                                                         *
 *    Copyarea for IFC events.                                        *
 *                                                                    *
 *    QW0185 is written for reads requests for IFCID 185.             *
 *    It contains a header section which is followed by               *
 *    a data section.                                                 *
 *                                                                    *
 *    The data portion of QW0185 begins with field:                   *
 *         - QW0185DD, if QW0185TP=S                                  *
 *         or                                                         *
 *         - QW0185DR, if QW0185TP=D                                  *
 *                                                                    *
 *                                                                    *
 *    If QW0185TP = S, the data portion consists of four              *
 *       fields followed by an arbitrary number of occurrences        *
 *       of the QW0185VR structure.                                   *
 *                                                                    *
 *    If QW0185TP = D, the data portion consists of:                  *
 *       ---> the data row, if QW0185RC = 0                           *
 *     or                                                             *
 *       ---> an error message, otherwise.                            *
 *                                                                    *
 **************** END OF CONTROL BLOCK SPECIFICATION *****************/

#pragma page(1)

typedef struct                    /* Describes a column in           *
                                   * a captured table                */
{
   short       qw0185st;          /* Tells the data type of          *
                                   * the column and whether          *
                                   * it has an associated            *
                                   * indicator variable              */
```

Figure 114 (Part 1 of 3). C IFC Copyarea for IFCIDS 0185

```
   short       qw0185le;          /* Defines the external lg         *
                                   * of a value from the column      */
   long        qw0185sd;          /* Contains the ccsid              */
   char        filler[2];
   short       qw0185sx;          /* Offset of this column           *
                                   * into the data row               */
   short       qw0185nl;          /* Length of column name           */
   char        qw0185cn[30];      /* Name of column                  */
} QW0185VR;

#pragma page(1)

/**********************************************************************
 *                                                                    *
 *  QW0185A is the structure containing the table description        *
 *  in its data portion (QW0185TP=S).                                 *
 *                                                                    *
 *  The data portion (QW0185DD) consists of 4 fields followed        *
 *  an arbitrary number of occurrences of the QW0185VR structure     *
 *                                                                    *
 **********************************************************************/

typedef struct
{

   long        qw0185ln;          /* Length of total db2cdc data     */
   char        qw0185tp;          /* Type of control block           *
                                   * S - DB2CDC table description    *
                                   * D - DB2CDC data row             */
   char        filler1[3];        /* Reserved                        */
   char        qw0185rc[4];       /* Reason code describing error    *
                                   * for this data portion           */
   char        qw0185cr[8];       /* Creator of table (auth id)      */
   char        qw0185tb[18];      /* Table name                      */
   char        qw0185ts[10];      /* Timestamp of table              *
                                   * description from catalog        */
   char        qw0185tl[10];      /* Timestamp of log buffer CI      *
                                   * when it is externalized or      *
                                   * when the buffer is initialized  */
   char        qw0185ur[8];       /* RBA of the first log record     *
                                   * for this unit of work.          */
   char        qw0185lr[8];       /* RBA of log record from          *
                                   * which this DB2CDC data row      */
   char        filler2[2];        /* Operation code,                 *
                                   * not used when QW0185TP=S.       */
   char        qw0185ri[2];       /* operation code qualifier.       *
                                   * RI - result of a referential    *
                                   *      constraint enforcement of  *
                                   *      a delete set null or       *
                                   *      cascade, if QW0185TP = "D". */
                                  /*                                 */
   char        filler3[6];        /* reserved                        */

 /************************  DATA PORTION  ****************************/

                                  /* Qualified table name            */
   char        qw0185id[8];       /* Eye catcher = "CDCDD   "        */
   long        qw0185bc;          /* Length of the QW0185DD section  */
   short       qw0185no;          /* Total number of occurrences     *
                                   * of QW0185VR                     */
```

Figure 114 (Part 2 of 3). C IFC Copyarea for IFCIDS 0185

```
   short       qw0185ld;          /* Number of cols described by      *
                                   * occurrences of QW0185VR          */
   QW0185VR    qw0185vr[1];
} QW0185A;

#pragma page(1)

 /*********************************************************************
  *                                                                   *
  *  QW0185B is the structure containing the data row or error        *
  *  message in its data portion (QW0185TP=D).                        *
  *                                                                   *
  *  The data portion (QW0185DR) consists of:                         *
  *      - the data row, if QW0185RC = 0                              *
  *    or                                                             *
  *      - an error message, otherwise.                               *
  *                                                                   *
  *********************************************************************/

typedef struct
{
   char        filler5[74];       /* Redefinition of 74 bytes         *
                                   * of the header portion            */
   char        qw0185pc[2];       /* Operation code,                  *
                                   * used when QW0185TP=D.            *
                                   * IN - Insert                      *
                                   * UB - Update before image         *
                                   * UA - Update after image          *
                                   * DE - Delete                      */
   char        filler6[8];

 /***********************  DATA PORTION  ****************************/

   char        qw0185dr[60];      /* Data row or error message        */
} QW0185B;

#pragma page(1)
```

Figure 114 (Part 3 of 3). C IFC Copyarea for IFCIDS 0185

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, Program, or service may be used. Any functionally equivalent product, Program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors.Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

fl (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. fl Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This publication is intended to help you administer IMS DataPropagator, hereafter called IMS DPROP.

This publication also documents general-use programming interface and associated guidance information provided by IMS DPROP.

General-use programming interfaces allow the customer to write programs that obtain the services of IMS DPROP.

General-use programming interface and associated guidance information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Notice

This chapter documents general-use programming interface and associated guidance information.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AD/Cycle
AT
CICS
CICS/ESA
CICS/MVS
COBOL/370
Database 2
DataPropagator
DataRefresher
DB2
DFSMS
DXT
IBM
IMS
IMS Client Server/2
IMS/ESA
Information Warehouse
Language Environment
MVS
MVS/ESA
OS/390
QMF
RACF
SAA
z/OS

Bibliography

The IMS DataPropagator for z/OS Version 3 Release 1 Library

Order Number	Book Title
GC27-1216	Administrators Guide for Log Asynchronous Propagation
GC27-1217	Administrators Guide for MQSeries Asynchronous Propagation
GC27-1215	Administrators Guide for Synchronous Propagation
SC27-1544	Concepts
GC27-1214	Customization Guide
GC27-1209	Diagnosis
GC27-1211	An Introduction
GC27-1212	Installation Guide
GC27-1213	Messages and Codes
GC27-1210	Reference
GC27-1208	Licensed Program Specification

Other Books Referenced in This Book

The following books are referred to in this book or might be helpful in understanding customization tasks:

- *DB2 Administration Guide*, SC26-4888
- *DB2 Messages and Codes*, SC26-4892
- *DB2 SQL Reference*, SC26-4890
- *DXT Writing Exit Routines*, SC26-4636
- *IMS/ESA General Information*, GC26-3068
- *IMS/ESA Customization Guide*, SC26-3064-00
- *IMS/ESA Administration Guide: Database Manager*, SC26-3065-00
- *IMS/ESA Application Programming: DL/I Calls*, SC26-3062-00
- *IMS/ESA Utilities Reference: Database Manager*, SC26-4627-00
- *IMS/ESA Utilities Reference: System*, SC26-4629-00
- *OS/390 Language Environment Programming Reference*, SC26-4841
- *IBM SAA AD/Cycle PL/I MVS & VM Language Reference*, SC26-3114
- Remote Recovery Data Facility Program Description and Operation, LY37-3710-03
- OS/390 Language Environment Programming Guide, SC26-4818

Glossary of Terms and Abbreviations

A

abort record. An IMS DataPropagator propagation log record (38*nn* or 5938), indicating that the associated unit of work will not be committed by IMS and should not be propagated to DB2. *Compare with commit record*.

ACB. Application control block. Located in IMS.

ACDC. Asynchronous changed data capture.

| **Apply Program**. A component of IMS MQ-DPROP that reads the MQSeries messages containing the changed data and passes it to the RUP. RUP transforms the changed data into relational format and updates the DB2 target tables.

Archive utility. A utility that filters out propagation log records from the records written to the IMS logs and writes them to Changed Data Capture data sets (CDCDSs).

asynchronous changed data capture. An IMS function that captures the changes needed for IMS DPROP asynchronous propagation and saves them on the IMS logs. The function is mandatory for IMS DPROP asynchronous propagation and is either implemented by an SPE (IMS 3.1) or built into the program (subsequent releases of IMS).

asynchronous propagation. The propagation of data at a later time, not within the same unit of work as the update call.

Audit Extract utility. An IMS DPROP utility that inserts the IMS DPROP audit records written to SMF into the IMS DPROP audit table.

AUDU. Audit Extract utility.

B

Batch Log data set. A data set that an IMS batch job uses to store propagation log records needed for IMS DPROP asynchronous propagation.

C

CAF. Call attach facility.

CCU. Consistency Check utility.

CDCDS. Changed Data Capture data sets.

CDCDS Registration utility. An IMS DPROP asynchronous propagation utility that registers new CDCDS to DBRC.

CDCDS Unregistration utility. An IMS DPROP asynchronous propagation utility that deletes CDCDS entries from DBRC.

CDU. CDCDS Unregistration utility.

| **CEC**. central electronics complex.

Changed Data Capture data set (CDCDS). The data sets that the archive utility uses to store the IMS DPROP asynchronous propagation log records filtered during the archive process. CDCDSs contain only the propagation log records. These log records are used by the Selector in place of the corresponding SLDSs, that contain all IMS changes.

Changed Data Capture exit routine. See DB2 Changed Data Capture exit routine

Changed Data Capture function. See DB2 Changed Data Capture function.

commit record. An IMS DPROP asynchronous propagation log record (9928, 37*nn*, 41*nn*, or 5937) indicating that the associated unit of work has been committed by IMS and should be propagated to DB2. *Compare with abort record*.

concatenated key. See “IMS concatenated key” and “conceptual concatenated key.”

conceptual concatenated key. The conceptual concatenated key of a segment consists of the concatenated keys of the segment's immediate physical parent and physical ancestors. Unlike the Conceptual *fully* Concatenated key, the conceptual concatenated key does not include the concatenated key of the segment itself.

conceptual fully concatenated key. The conceptual fully concatenated key is an IMS DPROP concept useful for the propagation of entity segments that do not have a unique IMS fully concatenated key; but that are nevertheless uniquely identifiable.

The conceptual fully concatenated key of a segment consists of these parts:

- the concatenated key of the segment
- the concatenated keys of the segment's physical parent and physical ancestors

The conceptual fully concatenated key is therefore the combination of these parts:

- the IMS fully concatenated key
- the ID fields (if any) of the segment that contribute to the concatenated key of the segment
- the ID fields (if any) of the physical parent or ancestors that contribute to the concatenated keys of the physical parent or ancestor

So, the conceptual fully concatenated key is equal to that hypothetical IMS fully concatenated key, that you would see if including the ID fields into the IMS key-field at each hierarchical level.

The concept of conceptual fully concatenated key allows the support of segments with a unique conceptual fully concatenated key, much in the same way as segments with a unique IMS fully concatenated key.

concatenated key. The concatenated key is an IMS DPROP concept useful for the propagation of entity segments that are neither unique under their parent nor have a unique IMS key, but that are nevertheless uniquely identifiable through ID fields.

The concatenated key is a combination of these fields that identify the segment uniquely under its parent:

- the non-unique IMS key field (if any)
- ID fields

For segments having a unique IMS key field, the conceptual key and the IMS key field are identical.

Consistency Check utility (CCU). An IMS DPROP utility that checks whether the data that has been propagated between IMS and DB2 databases is consistent. If not, it reports the inconsistencies and generates statements the DBA can use to fix the inconsistencies. The CCU is applicable when generalized mapping cases are being used.

containing IMS segment. An IMS segment that contains internal segments (embedded structures) propagated by mapping case 3 Propagation Requests. It is referred to interchangeably as a “containing IMS segment” or “containing segment.”

containing segment. See containing IMS segment.

CRU. CDCDS Registration utility.

D

Data Capture exit routine. See IMS data capture exit routine.

data capture function. An IMS function that captures the changes needed for data propagation.

DataRefresher. An IBM licensed program that lets you extract selected operational data on a periodic or one-time basis.

Data Extract Manager (DEM). A DataRefresher component that extracts the IMS data to which changes will subsequently be propagated. DEM also creates control statements for the DB2 Load utility to load the extracted IMS data into DB2 tables.

data propagation. The application of changes to one set of data to the copy of that data in another database system. See also synchronous propagation and IMS DPROP asynchronous propagation.

DataRefresher DEM. DataRefresher data extract manager.

DataRefresher Map Capture exit routine (MCE). See Map Capture exit routine.

DataRefresher UIM. See User Input Manager.

DBRM. Database Request Module.

DB2 commit count. The number of IMS commit records that the IMS DPROP asynchronous propagation receiver is to apply to DB2 before it issues a DB2 commit.

DB2 Changed Data Capture exit routine. The routine to which the DB2 Changed Data Capture function passes the DB2 changes it has captured for propagation. This routine can be the IMS DPROP HUP routine, that propagates data, or your own exit routine.

DB2 Changed Data Capture function. A DB2 function that captures the DB2 changes needed for data propagation.

DB2 Changed Data Capture subexit routine. An optional IMS DPROP exit routine invoked whenever the HUP is called by DB2 changed data capture. The DB2 Changed Data Capture subexit routine can typically be used to perform generalized functions such as auditing all of the captured DB2 changes.

DB2-to-IMS propagation. Propagation of changed DB2 tables to IMS segments. It can be either:

- One-way DB2-to-IMS propagation
- DB2-to-IMS propagation, as part of two-way propagation

DBD. Database definition. The collection of macroparameter statements that describes an IMS database. These statements describe the hierarchical structure, IMS organization, device type, segment length, sequence fields, and alternate search fields. The statements are assembled to produce database description blocks.

DBDLIB. Database definition library.

DBPCB. Database program communication block.

DEDB. Data entry database.

DEM. Data Extract Manager.

directory. See IMS DPROP directory.

DLU. DL/1 Load Utilities. IMS DPROP utilities that are used to create (or re-create) the IMS databases from the content of the propagated DB2 tables. You can use DLU if you have implemented DB2 to IMS or two-way propagation.

DPROP-NR. The abbreviation for IBM IMS DataPropagator MVS/ESA through Version 2.2. At Version 3.1 the product name changed to IMS DataPropagator, abbreviated as IMS DPROP.

E

EKYMQCAP. The Capture component of MQ-DPROP. EKYMQCAP is an IMS data Capture exit routine. It runs as an extension to the updating IMS application programs, but it is transparent to them. EKYMQCAP obtains the changed data from the IMS Data Capture function and sends this data via MQSeries messages to the Apply Program.

EKYRESLB Dynamic Allocation exit routine. An IMS DPROP exit routine that can be used to allocate dynamically the IMS DPROP load module library to the EKYRESLB DD-name.

entity segment. The data being mapped from IMS to DB2 comes from one single hierarchic path down to a particular segment. This segment is called the entity segment. See also mapping case 1.

ER. Extract request.

Event Marker. A component of MQ-DPROP that runs on the same system as the IMS source databases. It is used to identify an event that occurs on the Source System. The customer must execute the Event Marker on the Source System at the time that the event occurs.

The Event Marker transmits an MQSeries message that identifies the event to the Apply Program. This MQSeries message is transmitted in FIFO sequence and in the same Propagation Data Streams as the changed IMS data.

When an occurrence of the Apply Program processes this message, the content of the target DB2 tables of this occurrence of the Apply Program reflect the content of the IMS source databases at the time that the Event Marker was executed on the Source System.

The Event Marker is used for an automated stop of the Apply Program when the content of the target DB2 tables reflects a particular Source System point in time.

exit routines. IMS DPROP contains seven exit routines. See the individual glossary entries for:

- DB2 Changed Data Capture exit routine
- DB2 Changed Data Capture subexit routine
- IMS Data Capture exit routine
- Field exit routine
- Map Capture exit routine
- Propagation exit routine
- Segment exit routine
- User exit routine

extension segment. The data being mapped from IMS to DB2 comes from a single hierarchic path down to an entity segment and from any segments immediately subordinate to the entity segment. The segments subordinate to the entity segment can have zero or one occurrence beneath a single occurrence of the entity segment. This type of subordinate segment is called an extension segment (as it extends the data in the entity segment). See also mapping case 2.

extract request (ER). A DataRefresher request to extract IMS data. Extract requests become IMS DPROP propagation requests once they are validated by the IMS DPROP MCE.

F

Field exit routine. An IMS DPROP exit routine you can write to complement the logic of IMS DPROP's generalized mapping cases. Field exit routines are typically used to convert an individual IMS data field between a customer format IMS DPROP does not support and a format you have defined in your propagation request.

FIFO. First-In-First-Out

fully concatenated key. See IMS fully concatenated key and conceptual fully concatenated key.

G

generalized mapping cases. The mapping cases provided by IMS DPROP. See mapping case 1, mapping case 2 and mapping case 3.

group definition file. The file that the Group Unload utility (GUU) uses to store the IMS sources that it extracts from the IMS DPROP directory tables. *See also, SCF Compare job and SCF Apply job*.

Group Unload utility (GUU). The IMS DPROP asynchronous propagation utility that extracts details of all IMS sources for the specified propagation group from the IMS DPROP directory tables at the receiver site and writes them to the Group Definitions File. *See also, SCF Compare job and SCF Apply job*.

GUU. Group Unload utility.

H

hierarchical update program (HUP). The IMS DPROP component that does the actual DB2-to-IMS propagation. HUP is the IMS DPROP-provided DB2 Changed Data Capture exit routine. The DB2 Changed Data Capture function calls HUP and provides to HUP the changed IMS rows.

Hierarchical to Relational propagation. This is one-way hierarchical to relational propagation: the one-way propagation of changed IMS segments to DB2 tables. The terms *hierarchical to relational propagation* and *one-way IMS-to-DB2 propagation* are interchangeable.

HUP. Hierarchical Update program.

HSSR. High speed sequential retrieval.

I

ID fields. *Identification (ID) fields* are non-key fields that:

- uniquely identify a segment under its parent
- do not change their value

Typical examples of IMS segments with ID fields, are segments where the data base administrator has not defined the ID fields as part of the IMS Key field. For example because the IMS applications need to retrieve the segment in another sequence than the ascending sequence of the ID fields.

identification fields. See ID fields.

IMS concatenated key. For an IMS segment, the concatenated key consists of:

- The key of the segment's immediate parent, and
- The keys of the segment's ancestors

Unlike the IMS **fully** concatenated key of the segment, the concatenated key does not include the key of the segment itself.

A logical child segment has two concatenated keys: a physical concatenated key and a logical concatenated key. The physical concatenated key consists of the key of the segment's physical parent and the keys of the physical ancestors of the physical parent. The logical concatenated key consists of the key of the segment's logical parent and the keys of the physical ancestors of the logical parent.

IMS Data Capture exit routine. The routine to which the IMS Data Capture function passes the IMS changes it has captured for propagation. For synchronous propagation, this routine can be the IMS DPROP RUP routine, that propagates data, or your own exit routine. For IMS DPROP asynchronous propagation, the data capture exit routine is a program you write that gets the changed data from IMS. Other programs that you write will later invoke IMS DPROP with the changed IMS data.

IMS data capture function. An IMS function that captures the changes needed for data propagation.

| **IMS DPROP**. The abbreviated name for the IBM IMS DataPropagator product. Previously, this product was called IMS DataPropagator, abbreviated as DPROP-NR.

| **IMS DPROP directory**. A set of DB2 tables containing the mapping and control information necessary to perform propagation.

IMS fully concatenated key. For an IMS segment, the fully concatenated key consists of:

- The key of the segment,
- The key of the segment's immediate parent, and
- The keys of the segment's ancestors.

Unlike the IMS concatenated key of the segment, the fully concatenated key includes the key of the segment itself.

IMS INQY data. The first 9904 (update) record in each IMS unit of work (UOW) contains IMS INQY data (transaction name, PSB name, and user ID). This information is written to the PRDS for the propagation group as the first record of the UOW.

IMS log files. The files that IMS uses to store details of all changes to IMS data. See also, batch log data sets, online data sets (OLDSs), system log data sets (SLDSs), and Changed Data Capture data sets (CDCDSs).

IMS logical concatenated key. One of the two IMS concatenated keys of a logical child segment (the other is an IMS physical concatenated key). The logical concatenated key consists of:

- The key of the segment's logical parent, and
- The keys of the physical ancestors of the logical parent.

IMS physical concatenated key. One of the two IMS concatenated keys of a logical child segment (the other is an IMS logical concatenated key). The physical concatenated key consists of:

- The key of the segment's physical parent, and
- The keys of the physical ancestors of the physical parent.

IMS-to-DB2 propagation. This is the propagation of changed IMS segments to DB2 tables. Distinguish between:

- One-way IMS-to-DB2 propagation
- IMS-to-DB2 propagation, as part of two-way propagation

internal segments. Internal Segments is the IMS DPROP and DataRefresher term for structures embedded in IMS Segments, that are propagated through mapping case-3 propagation requests. Each embedded structure (i.e. each internal segment), is propagated to a different table; each occurrence of the embedded structure to one row of the table.

invalid unit of work. An IMS UOW that is missing a first record (containing the INQY data). If the IMS DPROP asynchronous propagation Selector detects an invalid unit, it responds according to what you specified on the INVUOW keyword of the SELECT control statements. If you specified:

IGNORE The Selector continues processing

STOP The Selector issues an error message and terminates

ISC. Inter-system communications.

ISPF. Interactive system production facility or Interactive structured programming facility.

IXF. Integrated exchange format.

L

| **LOG-ASYNC**. The IMS log-based, asynchronous propagation functions of IMS DPROP.

| Once the IMS log records are archived (IMS Online Logs) or de-allocated (IMS Batch Logs) by IMS and then stored in time-stamp sequence, LOG-DPROP reads the IMS logs to find the changed data and then stores the changed data in PRDS datasets. The Receiver component of IMS DPROP reads the PRDSs, transforms the data into the relational format, and applies the changes to the target DB2 tables.

| See asynchronous propagation.

logical concatenated key. See IMS logical concatenated key

M

Map Capture exit (MCE) routine. The map capture exit routine provided by DPROP. MCE is used when you provide mapping information through DataRefresher. MCE is called by DataRefresher during mapping and data extract to perform various validation and checking operations. The IMS DPROP MCE should be distinguished from the DataRefresher Map Capture exit, the DataRefresher routine that calls MCE.

mapping case. A definition of how IMS segments are to be mapped to DB2 tables. IMS DPROP distinguishes between mapping case 1, mapping case 2, and user mapping cases.

mapping case 1. One of the generalized mapping cases provided by IMS DPROP. Mapping case 1 maps one single segment type, with the keys of all parents up to the root, to a row in a single DB2 table.

mapping case 2. One of the generalized mapping cases provided by IMS DPROP. Mapping case 2 maps one single segment type, with the keys of all parents up to the root, plus data from one or more immediately subordinate segment types (with a maximum of one occurrence of each segment type per parent), to a row in a single DB2 table.

mapping case 3. One of the generalized mapping cases provided by IMS DPROP. Mapping case 3 supports the propagation of segments containing embedded structures. A typical example of an embedded structure is a repeating group of fields.

- each embedded structure can be propagated to/from a different table. Mapping case 3 propagates each occurrence of an embedded structure, with the key of the IMS segment, and the keys of the physical parent and ancestor, to/from a row of one DB2 table.
- the remaining data of the IMS segment (that is the fields that are not located in a embedded structure) can be propagated to/from another table.

Mapping Verification and Generation (MVG). An IMS DPROP component that validates the mapping information for each propagation request and stores it in the IMS DPROP directory. For a propagation request belonging to a generalized mapping case, MVG generates an SQL update module. MVG is invoked internally by MCE and MVGU.

Mapping Verification and Generation utility (MVGU). An IMS DPROP utility invoked by the DBA. MVGU creates propagation requests when DataRefresher is not used to provide mapping information (i.e., when you put the mapping information directly into the MVG input tables). MVGU also deletes or rebuilds propagation requests in the IMS DPROP directory.

master table. The IMS DPROP directory master table, that is created when IMS DPROP is initialized. It consists of one row, containing system and error information.

MCE. Map Capture exit routine.

MIT. Master Index Table.

MQ-ASYNC. The MQSeries-based, asynchronous propagation functions of IMS DPROP.

An IMS Data Capture Exit routine provided by IMS DPROP obtains the IMS Database changes in real time from IMS and sends the changes via MQSeries messages to an IMS DPROP Apply program. The Apply program reads the MQSeries messages, transforms the data into relational format, and then applies the new data to the target DB2 tables.

MQ-ASYNC supports both near-real time propagation and automated point-in-time propagation.

MQSeries. A family of IBM licensed programs that provide message queuing services.

MQSeries for OS/390. The members of the MQSeries that run on OS/390 systems.

MSDB. Main storage database.

MSC. Multisystem communication.

MVG. Mapping Verification and Generation.

MVG input tables. A group of DB2 tables into which the DBA stores propagation request definitions when DataRefresher is not used to provide mapping information. Once the propagation requests are stored, the DBA invokes MVGU. MVGU invokes MVG, that validates the propagation request and copies the mapping definitions from the MVG input tables to the IMS DPROP directory.

MVGU. Mapping Verification and Generation utility.

N

Near RealTime. A delay of only a couple of seconds.

O

OLDS. Online Data Set.

One-way DB2-to-IMS propagation. This is the propagation of changed DB2 tables to IMS segments. Distinguish between:

- One-way DB2-to-IMS propagation
- DB2-to-IMS propagation, as part of two-way propagation

One-way IMS-to-DB2 propagation. This is the propagation of changed IMS segments to DB2 tables. Distinguish between:

- One-way IMS-to-DB2 propagation
- IMS-to-DB2 propagation, as part of two-way propagation

P

PCB. Program communication block.

persistent MQSeries message. An MQSeries message that survives a restart of the MQSeries Queue Manager.

physical concatenated key. See IMS physical concatenated key.

Point In Time Propagation. An Asynchronous propagation is said to operate in 'Point In Time' mode, when the data content of the target databases matches the content of the source databases at a previous, clearly identified Point In Time. For example, a Point In Time Propagation can be used to reflect in the content of the target databases the logical end of a business day, or the logical end of business month, or the end of specific Batch jobstream that updated the source databases.

PR. Propagation request.

PR ID. Propagation request identifier.

PRCT. Propagation Request Control Table

PRDS. Propagation Request Data Set

PRDS register file. A data set created by the IMS DPROP asynchronous propagation Selector that contains details of the associated PRDS.

PRDS register table. An IMS DPROP directory table that is created at the Receiver site when IMS DPROP is installed. The table is initially empty and you must populate it, using the PRU REGISTER control statements.

PRDS Registration utility (PRU). An IMS DPROP asynchronous propagation utility that registers PRDSs in the PRDS Register Table.

propagation. See data propagation.

Propagation Data Stream. A stream of changed IMS data that flows in MQSeries messages from the Capture Component of IMS DPROP to the Apply Component of IMS DPROP. Propagation data streams are defined with PRSTREAM control statements in the //EKYTRANS file of EKYMQCAP.

propagation delay. The time elapsed between the update of the IMS source database by the application programs and the update of the target DB2 table by IMS DPROP.

Propagation exit routine. An IMS DPROP exit routine you can write to propagate data when the generalized mapping cases don't meet your needs. A Propagation exit routine must provide all the logic for data mapping, field conversion, and propagation.

propagation group. A subset of the propagation requests in the IMS DPROP directory propagation request table (IMS DPROP asynchronous only).

You can define as many propagation groups as you like, but any propagation request can be associated with one and only one propagation group.

propagation log records. IMS log records that the IMS DPROP asynchronous propagation Selector writes to PRDSs:

- 9904 (update) records
- Commit or abort records
- SETS/ROLS records

propagation request control table (PRCT). An IMS DPROP directory table that is created at the Receiver site when IMS DPROP is installed. It contains details of all propagation requests defined to IMS DPROP and, in combination with the RCT, enables the Receiver to ascertain:

- Which propagation requests are assigned to which Receivers
- The activity status of all defined Receivers
- The activity status of all propagation requests that are assigned to defined Receivers

Propagation Request data set (PRDS). A sequential file into which the IMS DPROP asynchronous propagation Selector writes all propagation log records for a propagation group.

propagation request (PR). A request to propagate data between IMS and DB2. You define propagation requests for each segment type that is to be propagated.

PR set. A group of logically related propagation requests, identified by having the same PRSET ID. PR sets are typically used when you propagate the same IMS data to multiple sets of DB2 tables.

PRU. PRDS Registration utility.

PSB. Program specification block.

R

RCT. Receiver control table.

Receiver. An IMS DPROP asynchronous propagation component that retrieves the propagation log records from a PRDS and passes them to the RUP, that uses them to update the DB2 target tables.

Applies to LOG-DPROP.

RECEIVER control statement. A control statement that is input directly into the IMS DPROP asynchronous propagation Receiver JCL to specify:

- The name of the Receiver that is to process a PRDS
- The names of the DB2 subsystem to be accessed and the DB2 plan
- The number of committed UOWs to process before a DB2 commit is issued

Applies to LOG-DPROP.

Receiver control table (RCT). An IMS DPROP directory table, that is created at the Receiver site when IMS DPROP is installed. The table is initially empty and you must populate it, using the SCU CREATEREC control statement. It contains details of all Receivers and, in combination with the PRCT, enables the Receiver to ascertain:

- Which propagation requests are assigned to which Receivers
- The activity status of all defined Receivers
- The activity status of all propagation requests that are assigned to defined Receivers

Applies to LOG-DPROP.

Relational to Hierarchical propagation. This is one-way relational to hierarchical propagation: the one-way propagation of changed DB2 tables to IMS segments. The terms *relational to hierarchical propagation* and *one-way DB2-to-IMS propagation* are interchangeable.

relational update program (RUP). The IMS DPROP component that does the actual IMS to DB2 propagation. RUP is the IMS DPROP-provided IMS Data Capture exit routine.

For synchronous propagation, the IMS Data Capture function calls RUP with the changed IMS segments.

For user asynchronous propagation, your routine gets the changes from IMS and later calls RUP.

For IMS DPROP asynchronous propagation, the Receiver gets the changes from the Selector-Receiver Interface and later calls RUP. In either case, RUP propagates the changes to DB2.

RIR. RIR is an IMS DPROP abbreviation for DB2 Referential Integrity Relationship. Database administrators can define RIRs between tables in order to request that DB2 catches and prevents update anomalies in the relational databases.

Implementation of RIRs between propagated tables is:

- Optional for one-way IMS to DB2 propagation

- Strongly recommended for DB2 to IMS and two-way propagation

RTT. Resource translation table.

RUP. Relational Update program.

RUP control block table. A single IMS DPROP directory table that contains one RUP propagation control block (PRCB) for each propagated segment type. Each RUP PRCB contains details of the relevant database and segment.

S

SCF. Selector Control File.

SCF Apply job. Uses the SCF control statements to create new propagation groups and to list and modify existing propagation groups in the SCF.

SCF Compare job. Used to compare the contents of the Group Definitions File with the propagation groups in the SCF and to generate SCF control statements to bring the SCF into line with the Group Definitions File.

SCF control statements. Can be generated automatically by the IMS DPROP asynchronous propagation GUU or input directly into the IMS DPROP asynchronous propagation SCF Apply utility JCL. The control statements modify the contents of the SCF records.

SCU. Status Change utility.

segment exit routine. An IMS DPROP exit routine you can write to complement the logic of the generalized mapping cases. Segment exit routines are typically used to convert a changed data segment from the form it has in your IMS database to a form you have defined in your propagation request.

SELECT control statements. Control statements that are input directly into the IMS DPROP asynchronous propagation Selector JCL to define the execution options for the Selector.

Applies to LOG-DPROP.

Selector. An IMS DPROP asynchronous propagation component that collects propagation log records from the IMS log files and writes them to PRDSs for later processing by the IMS DPROP asynchronous propagation Receiver component.

Applies to LOG-DPROP.

Selector control file. Created at Selector installation or generation time and contains the following control information that is essential to the operation of the Selector:

- Database records and propagation group records
- DBRC information
- Timestamp information

Applies to LOG-DPROP.

SLDS. System Log Data Set.

SNAP. system network analysis program

Source System. An OS/390 system where IMS source databases of the IMS DPROP propagation reside.

SQL update module. A module generated by MVG for each propagation request belonging to a generalized mapping case. An SQL update module contains all the SQL statements required to propagate to DB2 the changed IMS data for that propagation request.

SSM. Subsystem member. An IMS JCL parameter that identifies the PDS member that describes connection between IMS and the DB2 subsystems.

Status Change utility (SCU). An IMS DPROP utility that:

1. Changes the status of propagation requests in the synchronous environment. Propagation requests can be active, inactive, or suspended. The SCU also performs a variety of other service functions.
2. Maintains the Timestamp Marker Facility and populates the RCT and the PRCT in IMS DPROP asynchronous propagation.

synchronous propagation. The propagation of data within the same unit-of-work as the update call.

T

Target System. An OS/390 system where DB2 target tables of the IMS DPROP propagation reside.

Timestamp Marker Facility. Supports the statements that create, assign, and delete timestamp markers in the SCF. It is run as part of the SCU.

TSMF. Timestamp Marker Facility.

TSMF Callable Interface. A facility that allows a user application to create a stop timestamp for one or more propagation groups.

Two-way propagation. The combination of IMS-to-DB2 propagation and DB2-to-IMS propagation for the same data.

TW propagation. See two-way propagation.

U

UIM. User Input Manager.

ULR. Uncommitted Log Record.

uncommitted log records (ULR). When the IMS DPROP asynchronous propagation Selector terminates, it writes all uncommitted log records (propagation log records that have not yet been either committed or aborted by IMS) to the uncommitted log record data set. On a subsequent Selector execution, these records will be either written to the appropriate PRDS (if they have been committed by IMS) or deleted from the uncommitted log record data set (if they have been aborted by IMS).

UOW. Unit of work.

| **USER-ASYNC**. The User asynchronous propagation functions of IMS DPROP.

user exit. See exit routines.

User Input Manager (UIM). A DataRefresher component to which you describe your IMS databases and the mapping between IMS databases and DB2 tables. The mapping is defined by submitting extract requests. You can specify on an extract requests that the UIM is to invoke the DataRefresher Map Capture exit routine provided by IMS DPROP and pass it the DataRefresher mapping definitions of the extract request.

user mapping case. A mapping case you can develop if the generalized mapping cases don't meet your needs.

V

| **Virtual Lookaside Facility (VLF)**. An MVS/ESA component that is a specific implementation of data spaces. IMS DPROP exploits VLF for a high-performance retrieval of mapping information and other control information.

VLF. Virtual Lookaside Facility.

Index

Numerics

A

B

C

D

E

F

N

P

Q

R

S

T

U

W

X

How to send your comments

IMS DataPropagator for z/OS
Customization Guide
Version 3 Release 1

Publication No. SC27-1214-00

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other Data Management Tools documentation. You can use the following methods to send your comments.

- Send your comments by e-mail to comments@vnet.ibm.com and include the name of the product, the version number of the product the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title, page number, or a help topic title).
- Complete the readers' comment form at the back of the book and return it by mail, by fax (800-426-7773 for the United States and Canada), or by giving it to an IBM representative.

Readers' Comments

IMS DataPropagator for z/OS
Customization Guide
Version 3 Release 1

Publication No. SC27-1214-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

May we contact you to discuss your comments? ☐ Yes ☐ No

Name

Address

Company or Organization

Phone No.

Readers' Comments
SC27-1214-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL   PERMIT NO. 40   ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department HHX/H3
PO Box 49023
San Jose, CA 95161-9023

Fold and Tape | **Please do not staple** | Fold and Tape

SC27-1214-00

Cut or Fold
Along Line

Program Number: 5655-E52

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC27-1214-00