DataRefresher
Version 1

Command Reference

SH19-6999-00

SH19-6999-00

DataRefresher
Version 1

Command Reference

Note!

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

First Edition (October 1994)

This edition applies to Version 1 of DataRefresher, Program Number 5696-703, and to all releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for readers' comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Software Solutions,
Information Development (IISL),
2 Burlington Road,
Dublin 4,
Ireland.
Fax: (Ireland) +353 - 1 -6614246
IBMMAIL: IEIBM3FL at IBMMAIL
INTERNET: IEIBM3FL@IBMMAIL.COM

Contents

Notices

References in this publication to IBM* products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Corporation, IBM Director of Licensing, 208 Harbor Drive, Stamford, Connecticut, United States, 06904.

Authorized use of IBM online books

For online versions of this book, we authorize you to:

Copy, modify, and print the documentation contained on the media, for use within your enterprise, provided you reproduce the copyright notice, all warning statements, and other required statements on each copy or partial copy.

Transfer the original unaltered copy of the documentation when you transfer the related IBM product (which may be either machines you own, or programs, if the program's license terms permit a transfer). You must, at the same time, destroy all other copies of the documentation.

You are responsible for payment of any taxes, including personal property taxes, resulting from this authorization.

THERE ARE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Your failure to comply with the terms above terminates this authorization. Upon termination, you must destroy your machine readable documentation.

Programming Interface information

This book is intended to help database administrators, DataRefresher* users, and application programmers use DataRefresher.

This book documents General-Use Programming and Associated Guidance Information.

---

* IBM and OS/2 are trademarks of the International Business Machines Corporation.

General-Use programming interfaces allow the customer to write programs that obtain the services of DataRefresher.

General-Use Programming Interface Information is identified where it occurs by the following graphic:

General-Use Programming Interface

General-Use Programming Interface...

End of General-Use Programming Interface

Trademarks and Service Marks

The following terms, denoted by an asterisk (*), used in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

AD/Cycle
AS/400
CICS
DataHub
DataPropagator
DB2
DB2/2
DB2/6000
DFSORT
DXT
DXT/D1
IBM
Information Warehouse
Language Environment
OS/2
PS/2
QMF
RACF
SAA
SQL/DS
Systems Application Architecture
System/370
S/370
VM/XA
VTAM
RS/6000
Virtual Machine/Enterprise Systems Architecture

The following terms, denoted by two asterisks (**), in this publication, are trademarks of other companies:

Bridge/Fastload (Bridge Technology Inc.)
VMS is a trademark of the Digital Equipment Corporation

About this book

This book provides reference information for using DataRefresher* Version 1 in an MVS host environment. It has been written with the intention of helping experienced database administrators, DataRefresher users, and application programmers obtain reference material needed to perform DataRefresher tasks. This book:

- Introduces DataRefresher and the IBM* Information Warehouse*
- Explains how to read syntax diagrams
- Defines the structure, content, and general rules for DataRefresher commands.
- Provides a table of all the DataRefresher commands
- Provides a syntax of all the DataRefresher commands, and a detailed description of all keywords, values and statements of each command
- Lists DataRefresher limits and return codes
- Describes DataRefresher models
- Describes EBCDIC and SOURCE source format
- Explains Integration Exchange Format (IXF)

If you intend to use DataRefresher from a workstation refer to the *DataRefresher OS/2 User's Guide* for more information.

What you should know

To gain the most benefit from this book, it would be helpful if you have familiarity with the following:

- Multiple Virtual Storage (MVS)
- MVS job control language (JCL)
- IBM DATABASE 2* (DB2*)
- Information Management System/Virtual Storage (IMS/VS) or Information Management System/Enterprise System Architecture (IMS/ESA)
- OS/VS DB/DC Data Dictionary
- IBM Virtual Machine/System Product (VM/SP)
- Time Sharing Option (TSO)
- Conversational Monitor System (CMS)
- SQL/Data System (SQL/DS*)
- Interactive System Productivity Facility (ISPF)
- A general knowledge of the structure and function of DataRefresher

* IBM, Information Warehouse, and SQL/DS are trademarks of the International Business Machines Corporation.

DataRefresher library overview

The following describes the contents and organization of information in the DataRefresher Version 1 library.

DataRefresher Version 1 (5696-703)	Order number
An Introduction This book provides an overview of DataRefresher. It describes the uses, benefits, and requirements of DataRefresher to help you evaluate the product.	GH19-6993-00
Licensed Program Specifications This document briefly describes the technical information for DataRefresher and is the warranty for the product.	GH19-9994-00
Administration Guide This book describes how to plan for the installation and the use of DataRefresher in your organization. It describes how to set up and administer DataRefresher.	SH19-6995-00
MVS and VM User's Guide This book describes how to use DataRefresher in an MVS or VM environment. In particular, it describes how to use the DataRefresher Administrative Dialogs and End User Dialogs to create and run extract requests.	SH19-6996-00
OS/2 User's Guide This book describes how to use DataRefresher on a workstation. It describes how to register your host data sources and create an extract which can be run on the host MVS system.	SH19-6997-00
Exit Routines This book describes how to write user exit routines to be used by DataRefresher when an extract is processed.	SH19-6998-00
Command Reference This book provides detailed reference information for all of the DataRefresher commands and procedures.	SH19-6999-00
Messages and Codes This book lists the DataRefresher messages with explanations and suggested responses.	SH19-5000-00
Diagnosis Guide This book contains the information required to diagnose problems with DataRefresher. It also contains information that can help you communicate with the IBM Support Center to isolate and solve problems with DataRefresher.	LY19-6386-00

Sources of related information

The following books are referenced in this publication:

IBM Database 2 publications

IBM DATABASE 2 Version 2: SQL Reference. (SC26-4380)

OS/VS2 publications

IBM Operators Library: OS/VS2 MVS System Commands. (GC38-0229)

OS/VS publications

IBM OS/VS DB/DC Data Dictionary Applications Guide. (SH20-9190)

IBM OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference. (SH20-9189)

System/370 publications

IBM S/370 Principles of Operation. (GA22-7000)

IBM 370/XA Principles of Operation. (SA22-7085)

Chapter 1. Using DataRefresher

You can use DataRefresher to extract data from a source database or file and format it for a target database or file on a different system.

Valid Data Sources	Valid Data Targets
IMS databases	DB2 tables
DB2[1] tables	SQL/DS tables
SQL/DS tables (VM only)	Physical sequential data sets
VSAM data sets	CMS files under VM
Physical sequential data sets	AS/400* files
	DB2/2* tables [1]
	DB2/6000* tables[1]

DataRefresher provides you with facilities for registering exit routines which can be used by an extract to extend the range of sources and targets. For example, the Generic Data Interface (GDI) exit routine makes it possible for you to extract data from an IXF file, while the Generic Output Interface (GOI) exit routine provides you with the flexibility to make changes to the extracted data so that it can be received by a target system not directly supported by DataRefresher.

For more information about exit routines, see *DataRefresher Exit Routines.*

DataRefresher and other products

DataRefresher is one of the products in the IBM Data Replication family, part of the IBM Information Warehouse Framework, which addresses database refresh, update, and administration requirements. DataRefresher can also be used with several other IBM and non-IBM products.

DataRefresher and the IBM Data Replication family

DataRefresher can be used with the following Data Replication products from IBM:

DataHub

DataHub* acts as a single control point for Data Replication. DataRefresher and DataPropagator Relational can be run from DataHub, enabling the DataHub user to administer the DataRefresher sources supported by the OS/2* user interface. The use of DataRefresher from within DataHub extends the range of DataHub, enabling it to become the single data replication control point for both relational and non-relational data sources.

---

1 Via standard utilities or vendor supplied products, such as Bridge Technology Fastload**

* DB2, AS/400, DB2/2, DB2/6000, DataHub, DataPropagator, PS/2, System/370, and QMF are trademarks of the International Business Machines Corporation.

** Bridge/Fastload is a trademark of Bridge Technology Inc.

DataPropagator NonRelational

DataPropagator NonRelational* is an IBM program product that propagates data from IMS to DB2 databases. Data propagation involves applying the changes made to a data set on one database system to a copy of that database on another database system, maintaining the consistency between the two copies of the same data. DataPropagator NonRelational also supports the refreshing of data from DataRefresher sources into its staging tables.

DataRefresher's data definitions and mappings can be used by DataPropagator NonRelational, allowing the provision of a coordinated refresh and update data between IMS and DB2. Additionally, DataRefresher's GUI enables you to define a DataPropagator NonRelational propagation request.

DataPropagator Relational

DataPropagator Relational provides automated copying facilities between the IMS DB2 Relational DataBase Family. DataPropagator Relational supports the direct application of changes from IMS into its staging tables by DataPropagator NonRelational. This enables DataPropagator Relational to propagate the changes as if they had come from a DB2 database. DataPropagator Relational also supports the refreshing of data from DataRefresher sources into its staging tables.

DataRefresher and other IBM products

DataRefresher has been designed to move large amounts of data from a source database, data set, or file, to a target database, data set or file. DataRefresher also complements the following IBM products:

Data Extract (DXT)

DataRefresher builds on the proven capabilities of its predecessor DXT, and many of its components are compatible with DXT Version 2 Releases 4 and 5. You can migrate your DXT Version 2 Release 4 and 5 File Description Table Libraries and Extract Request Libraries to DataRefresher.

Information on how to migrate from DXT to DataRefresher is provided in the *DataRefresher Administration Guide*.

Query Management Facility (QMF)

QMF* is a report writing product that runs on both MVS and VM. After you use DataRefresher to extract and load data into a DB2 or SQL/DS database, you can use QMF to create graphs and reports from the data.

You can also use QMF to import Integration Exchange Format (IXF) files of extracted data from DataRefresher and load these files into relational databases.

Personal/Application System (AS)

AS is an integrated decision support tool providing financial modeling, business graphics, statistical analysis, and end-user application development for relational data.

You can import IXF formatted tables into AS after extracting them using DataRefresher. You can also invoke the DataRefresher End User Dialogs, or send a dialog data extract to DataRefresher, from within AS.

Enhanced Connectivity Facilities (ECF)

ECF is a tool for connecting an IBM PC, PS/2*, or PS/55 workstation to a System/370* host processor.

ECF provides an interface which you can use with the DataRefresher Dialogs to run an existing DataRefresher extract request from your workstation. When you use DataRefresher to extract data to an output IXF file, ECF can download the data to your workstation.

DataRefresher and non-IBM products

Many organizations are moving their informational data from large host systems to powerful workstations. DataRefresher can be used together with other vendor's products to move data from your host systems to a series of smaller database systems on a PS/2 or RS/6000* machine.

Fastload, for example, can be used with DataRefresher to load data into DB2/2 and DB2/6000 tables. This means that DataRefresher can join the data from multiple data sources and pass this data to Fastload, which loads the data into DB2/2 or DB2/6000.

Chapter 2. Task overviews for some DataRefresher extracts

This chapter outlines the tasks that DataRefresher performs to extract data from a specific source and direct data to a specific target. If you want to perform the extract without using the OS/2 interface or the dialogs, use the tables as a guideline and modify the steps to suit your needs.

The sources are:

- IMS database (MVS)
- VSAM dataset
- DB2 + IMS
- DB2 + VSAM
- IMS + VSAM
- DB2 database (MVS)
- SQL/DS database
- DB2 + DB2 (same system)

The targets are:

- DB2 table (MVS)
- SQL/DS table (VM or VSE)
- Physical sequential dataset (MVS)
- CMS file (VM)
- Dataset or file in IXF (MVS or VM)

Where appropriate the name of the DataRefresher model best suited to accomplish a task is provided in parentheses.

For more information about extracting data using DataRefresher see the *DataRefresher MVS and VM User's Guide*, and the *DataRefresher OS/2 User's Guide*.

IMS to:

DB2 or SQL/DS	Physical sequential dataset or CMS file	Dataset or file in IXF
1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 4. Write extract request (DVREDEXT) and JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 5. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 6. Run DEM to get request from EXTLIB, extract data, and load data into target table	1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 4. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 5. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 6. Run DEM to get request from EXTLIB and extract data	1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 4. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 5. Specify IXF in SUBMIT command 6. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 7. Run DEM to get request from EXTLIB and extract data

VSAM to:

DB2 or SQL/DS	Physical sequential dataset or CMS file	Dataset or file in IXF
1. Create DXTFILE description of VSAM data (DVREDREP) 2. Create DXTVIEW of file (DVREDRVF) 3. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 4. Write extract request (DVREDEXT) and JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 5. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 6. Run DEM to get request from EXTLIB, extract data, and load data into target table	1. Create DXTFILE description of VSAM data (DVREDREP) 2. Create DXTVIEW of file (DVREDRVF) 3. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 4. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 5. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 6. Run DEM to get request from EXTLIB and extract data	1. Create DXTFILE description of VSAM data (DVREDREP) 2. Create DXTVIEW of file (DVREDRVF) 3. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 4. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 5. Specify IXF in SUBMIT command 6. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 7. Run DEM to get request from EXTLIB and extract data

DB2 + IMS to:

DB2 or SQL/DS	Physical sequential dataset or CMS file	Dataset or file in IXF
1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Create DXTFILE of GDI exit request for DB2 (DVRXOGRX) 4. Create DXTVIEW of the exit file 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 7. Specify join in WHERE clause of SELECT statement 8. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 9. Run DEM to get request from EXTLIB, extract data, and load data into target table	1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Create DXTFILE of GDI exit request for DB2 (DVRXOGRX) 4. Create DXTVIEW of the exit file 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 7. Specify join in WHERE clause of SELECT statement 8. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 9. Run DEM to get request from EXTLIB and extract data	1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Create DXTFILE of GDI exit request for DB2 (DVRXOGRX) 4. Create DXTVIEW of the exit file 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 7. Specify join in WHERE clause of SELECT statement 8. Specify IXF in SUBMIT command 9. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 10. Run DEM to get request from EXTLIB and extract data

DB2 + VSAM to:

DB2 or SQL/DS	Physical sequential dataset or CMS file	Dataset or file in IXF
1. Create DXTFILE description of VSAM data (DVREDREP) 2. Create DXTVIEW of file (DVREDRVF) 3. Create DXTFILE of GDI exit request for DB2 (DVRXOGRX) 4. Create DXTVIEW of the exit file 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 7. Specify join in WHERE clause of SELECT statement 8. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 9. Run DEM to get request from EXTLIB, extract data, and load data into target table	1. Create DXTFILE description of VSAM data (DVREDREP) 2. Create DXTVIEW of file (DVREDRVF) 3. Create DXTFILE of GDI exit request for DB2 (DVRXOGRX) 4. Create DXTVIEW of the exit file 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 7. Specify join in WHERE clause of SELECT statement 8. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 9. Run DEM to get request from EXTLIB and extract data	1. Create DXTFILE description of VSAM data (DVREDREP) 2. Create DXTVIEW of file (DVREDRVF) 3. Create DXTFILE of GDI exit request for DB2 (DVRXOGRX) 4. Create DXTVIEW of the exit file 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 7. Specify join in WHERE clause of SELECT statement 8. Specify IXF in SUBMIT command 9. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 10. Run DEM to get request from EXTLIB and extract data

IMS + VSAM to:

DB2 or SQL/DS	Physical sequential dataset or CMS file	Dataset or file in IXF
1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Create DXTFILE description of VSAM data (DVREDREP) 4. Create DXTVIEW of file (DVREDRVF) 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 7. Specify join in WHERE clause of SELECT statement 8. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 9. Run DEM to get request from EXTLIB, extract data, and load data into target table	1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Create DXTFILE description of VSAM data (DVREDREP) 4. Create DXTVIEW of file (DVREDRVF) 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 7. Specify join in WHERE clause of SELECT statement 8. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 9. Run DEM to get request from EXTLIB and extract data	1. Create DXTPSB description of IMS data (DVREDREP) 2. Create DXTVIEW of PCB (DVREDRVP) 3. Create DXTFILE description of VSAM data (DVREDREP) 4. Create DXTVIEW of file (DVREDRVF) 5. Run UIM to store data descriptions in FDTLIB (DVREDDXT) 6. Write extract request (DVREDEXT) and JCS to reformat data using DRU (DVREDRUV) 7. Specify join in WHERE clause of SELECT statement 8. Specify IXF in SUBMIT command 9. Run UIM to store extract request and JCS in EXTLIB (DVREDEXT) 10. Run DEM to get request from EXTLIB and extract data

DB2 to:

DB2 or SQL/DS	Physical sequential dataset	CMS file	Dataset or file in IXF
1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Write JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 3. Run REM to extract data and load into target table (DVREDRXM)	1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Run REM to extract data (DVREDRXM).	1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Write JCS to reformat data using the DRU (DVREDRUV) 3. Run REM to extract data (DVREDRXM)	1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Write JCS to reformat data using the DRU (DVREDRUV) 3. Run REM to extract data (DVREDRXM) 4. Specify IXF in SUBMIT command

SQL/DS to:

DB2 or SQL/DS	Physical sequential dataset	CMS file	Dataset or file in IXF
1. Write JCL to start REM and link to SQL/DS (DVREDREV) 2. Write JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 3. Run REM to extract data and load data into target table (DVREDRXV)	1. Write JCL to start REM and link to SQL/DS (DVREDREV) 2. Write JCS to reformat data using DRU (DVREDRUM) 3. Run REM to extract data (DVREDRXV)	1. Write JCL to start REM and link to SQL/DS (DVREDREV) 2. Run REM to extract data (DVREDRXV)	1. Write JCL to start REM and link to SQL/DS (DVREDREV) 2. Run REM to extract data (DVREDRXV). 3. Specify IXF in SUBMIT command

DB2 + DB2 (same system) to:

DB2 or SQL/DS	Physical sequential dataset	CMS file	Dataset or file in IXF
1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Write JCS to load extracted data into a DB2 table (DVREDDJC) or an SQL/DS table (DVREDSJC) 3. Run REM to extract data and load into target table 4. Specify join in WHERE clause (DVREDRXM)	1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Run REM to extract data 3. Specify join in WHERE clause (DVREDRXM)	1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Write JCS to reformat data using the DRU (DVREDRUV) 3. Run REM to extract data 4. Specify join in WHERE clause (DVREDRXM)	1. Write JCL to start REM and link to DB2 (DVREDREM) 2. Write JCS to reformat data using the DRU (DVREDRUV) 3. Run REM to extract data 4. Specify join in WHERE clause (DVREDRXM) 5. Specify IXF in SUBMIT command

Chapter 3. Reading DataRefresher syntax diagrams

A syntax diagram shows you how to specify a command so that DataRefresher can correctly interpret what you type.

Syntax diagrams show the keywords, values, and punctuation for each DataRefresher command. Read DataRefresher syntax diagrams from left to right and top to bottom following the horizontal line (the main path). If the line ends with an arrowhead, the command syntax is continued and the next line starts with an arrowhead. A double arrow indicates the end of the command syntax.

When you type a command from the syntax, be sure to include all punctuation, such as quotation marks, commas, equal signs, and blanks as shown in the syntax diagram.

Syntax notation of keywords, values and statements

Each keyword is made up of one or more keywords, some of which have values associated with them. Keywords are sometimes grouped together to make up a statement. Keywords appear in upper case letters.

The value associated with a keyword can be one of three types:

default value
shown one line above the main syntax path. These values are used in the command if you do not specify another value.

non-variable value
shown with upper case letters. These are compulsory in the command. Specify non-variable values exactly as printed in the syntax diagrams.

variable value
shown in lower case italic letters. Specify variable values by replacing the variable with a permitted value.

The following syntax illustrates keywords, default, non-variable, and variable values.

To specify that KEYWORD1 has the non-variable value, you would enter:

```
KEYWORD1=NON-VARIABLE
```

If you were to enter:

```
KEYWORD1
```

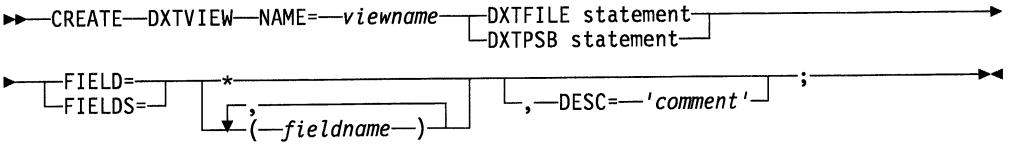
the default value associated with that keyword would also be assumed.

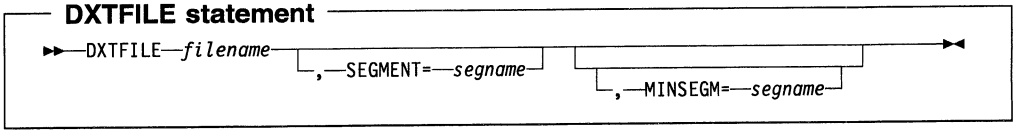
KEYWORD2 has a variable value. For example, you could enter:

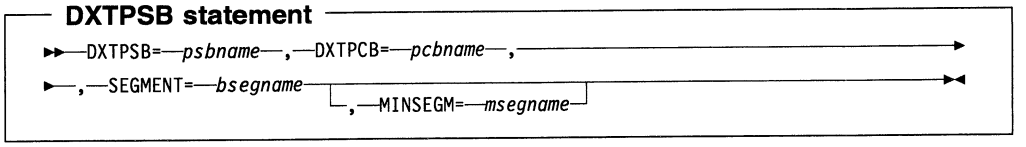
```
KEYWORD2 variable
```

Keywords and values that make up a statement are grouped together in the syntax diagrams. If a statement is too large to fit into the syntax diagram, it is represented by the name of the statement and framed.

For example, the CREATE DXTVIEW syntax diagram contains two framed statements:

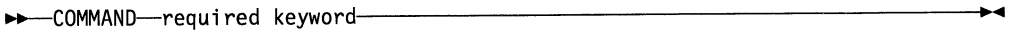






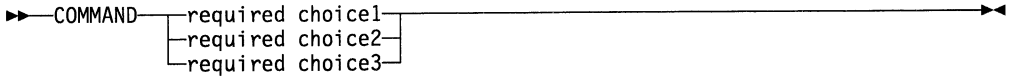
Required and optional keywords and values

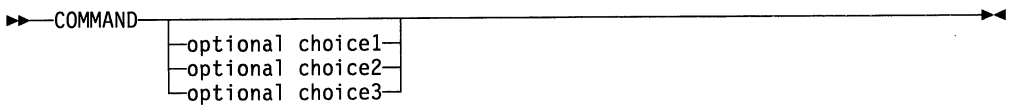
Required keywords and values appear on the main path of the syntax diagram. When entering required keywords and values, you must use the order shown in the syntax diagrams.



Optional keywords and values appear below the main path of the syntax diagram. You can enter optional keywords and values in any order.

When there is a list of keywords or values to choose from, a keyword could be required or optional. If one of the choices lies on the main path, a keyword or value in the list is required. If all choices lie above or below the main path, a keyword or value is optional.

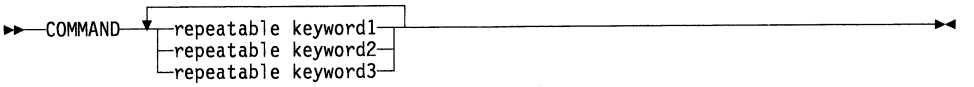




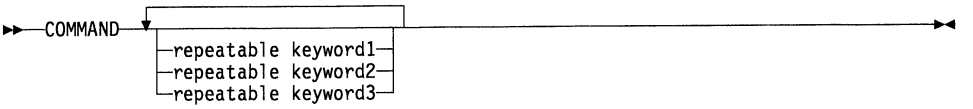
Repeat symbol

An arrow returning to the left, above the syntax path, indicates that items can be repeated following these conventions:

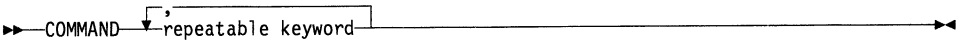
- If there is a vertical list of keywords or values and one item is on the main path, you can specify as many keywords or options as you want, but you must specify **at least one**. The same keyword or option cannot be repeated.



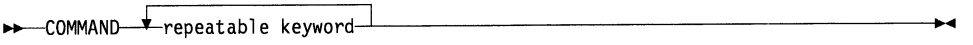
- If there is a vertical list of keywords or options and none lie on the main path, you can specify as many as you want but none are required. The same keyword or option cannot be repeated.



- A comma or semicolon included in the repeat symbol indicates you must separate multiple keywords or values with the comma or semicolon.

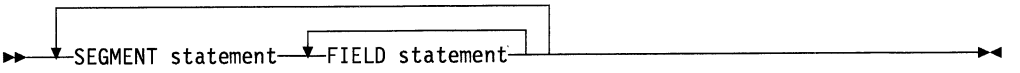


- If just one keyword or value or one statement lies on the main path within a repeat symbol, it can be repeated as many times as needed.



Example of the repeat symbol

In the CREATE DXTFILE command, the SEGMENT statement and FIELD statement can be repeated:



These repeat symbols indicate that you can have one or more FIELD statements within a SEGMENT statement, and one or more SEGMENT statements in a CREATE DXTFILE command.

Chapter 4. Using DataRefresher commands

You can use DataRefresher commands to extract data from either relational or non-relational data sources. There are also other DataRefresher commands that you can use to extract data from both of these source types.

Extracting data from non-relational data sources

When you are extracting data from a non-relational source, you use the General Data Extract Feature which consists of two major components:

- **User Input Manager (UIM)**
- **DataRefresher Manager (DEM)**

To extract from a non-relational data source:

1. Create data descriptions that describe the format of the source data and the intended format of the extracted data.
2. Create a job stream that invokes UIM to validate your data descriptions and save them in the FDTLIB.
3. Build and send the extract request to the EXTLIB. You must use a subset of the Structured Query Language (SQL) SELECT statement in the extract request. UIM is used to validate the extract request and save it in the EXTLIB.
4. Invoke the DEM to run the extract.

In addition to UIM and DEM, DataRefresher provides:

Dictionary Access Program (DAP)
A program for generating data descriptions of source files and non-relational databases from which data can be extracted. DAP retrieves these descriptions from the IBM OS/VS DB/DC Data Dictionary.

Structures Access Program (SAP)
A program which uses user-specified data structures to generate data descriptions and extract request statements. The SAP can also be used to generate statements when creating DB2 tables to hold the extracted data.

Extracting data from relational data sources

When you are extracting data from a relational source you use the Relational DataRefresher Feature which consists of one major component:

Relational Extract Manager (REM)
This is the extract program for relational databases. It is a DB2 application on an MVS system and an SQL/DS application on a VM system.

Because the source data is relational (from DB2 or an SQL/DS database), the REM uses the data descriptions stored in the database catalog. There is no need to create new data descriptions, REM can be used to extract the data directly.

The REM runs the extract request by:

1. Combining the extracted data with its control information to tell the target system what to do with the extracted data.
2. Writing the extracted data, and optionally the control information, to a data set or file.

You may wish to extract the data and move it to a data set or IXF file, so that you can use it with another program or product, for example QMF.

The following outlines how to extract data from a relational database:

1. Write the JCL or VM commands to start REM and link to the source database.
2. Build an extract request using `SUBMIT` and `EXTRACT`.
3. Write JCS to direct the extracted data into the target.
4. Run the job to extract the data and move the extracted data to the target.

When setting up DataRefresher to be used with REM:

- Implement a data-protection scheme for DB2 and SQL/DS data. If you do not have such a scheme, DataRefresher users can read data that they may not be authorized to read. The scheme can be based on the DB2 and SQL/DS authorization facilities or the view defined for DataRefresher in DB2 and SQL/DS.
- Use the JCL procedures, EXECs, and extract request models supplied with DataRefresher.

Other ways to use DataRefresher commands

Online commands
The online DataRefresher commands are TSO REXX EXECs that let you build and execute DataRefresher requests online. This is an alternative way to run the UIM, DEM and REM without using DataRefresher dialogs or JCL to submit the request.

Administrative Dialogs
The DataRefresher administrator can use the Administrative Dialogs commands to perform several tasks such as creating and maintaining data descriptions, extract requests, and JCL/JCS.

End User Dialogs
The less experienced DataRefresher user can use the End User Dialogs commands to perform data extract.

General rules for writing DataRefresher commands

Each command described in this guide includes a syntax diagram, keyword and value descriptions, and usage examples.

The keywords are classified as either REQUIRED or OPTIONAL.

The commands are divided into chapters according to function. Each chapter has a brief introduction explaining the function of the commands presented in the

chapter. Where applicable, there is also reference to any other book in the DataRefresher library that explains the function in more detail.

What portion of a line you can use

When writing a DataRefresher command:

- You can use columns 1 through 72 of a line for a command.
- You can use columns 73 through 80 only for sequence numbers.

Note: DataRefresher enforces these rules for the REM, DEM, UIM, and DAP.

Writing the command

You write a DataRefresher command by including its name, keywords, values, punctuation included in the syntax diagram, and a semicolon to indicate the end of the command. (Semicolons are required on all commands except DataRefresher Dialog commands, DEM operator commands, and DataRefresher Online commands.) Keywords and their values can be grouped into a *statement*. Be sure to:

- Include spaces between the name, keywords, and values
- Start the name anywhere from column 1 through column 72
- Write the keywords in the order that they appear in the syntax diagram

Continuing the command

You do not need any indicator to continue a command from one line to the next, apart from single arrows at the end of one line and at the beginning of the next. For more information, see Chapter 3, "Reading DataRefresher syntax diagrams" on page 11.

Each line consists of 72 characters, the first line holds characters 1 through 72, the second line holds characters 73 through 144, and so on.

These lines are concatenated to form the command, so you must include spaces between keywords and values at the end of one line and the beginning of the next. For example, if there is a character in column 72 of one row, column 1 of the next row must contain a blank if you want separation. Otherwise, DataRefresher will concatenate those keywords or values.

Reserved words

There are no reserved words in DataRefresher. Variables that you provide can be any word. For example, SUBMIT, which is the name of a DataRefresher command, can be used as the name of an extract request.

Inserting blanks

You can insert blank lines anywhere in a DataRefresher command. You can insert one or more blanks spaces before or after:

- Punctuation in the syntax diagram
- The equal sign between the keyword and value
- Any keyword or value
- The semicolon delimiter

Writing comments

You can write comments (to improve readability) without changing the command:

- Begin a comment with the characters /* and end it with the characters */
- You can place a comment anywhere that you can put a blank

DataRefresher naming conventions

The values you supply with DataRefresher commands must conform to certain naming conventions. Rather than explaining which rules apply to each individual name each time it is discussed, this section summarizes DataRefresher naming conventions.

Table 1. DataRefresher naming conventions

Type of name	Maximum length	First character can be	Remaining characters can be
Alphameric	8 Characters	Alphabetic (A-Z)	Alphabetic (A-Z) or Numeric (0-9)
Alphameric/ special characters	8 Characters	Alphabetic (A-Z), @, #, or $	Alphabetic (A-Z), Numeric (0-9), @, #, or $
SQL	18 characters	Alphabetic (A-Z), @, #, $, or _	Alphabetic (A-Z), Numeric (0-9), @, #, $, or _
SQL Quoted	18 characters written between double quotation marks (")	Any character except double quotation marks (")	Any character except double quotation marks (")
DBCS*	Requires a shift-out character (X'OF') and shift-in character (X'OF')	Must be shift-out character.	Any DBCS characters. Must end with shift-in character
DataRefresher	32 characters	Alphabetic (A-Z), @, #, $, or _	Alphabetic (A-Z), Numeric (0-9), @, #, $, or _
DataRefresher Quoted	32 characters written between double quotation marks (")	Any character except double quotation marks (")	Any character except quotation marks (")

***Note** When writing DBCS names, each "shift-out" character must have a matching "shift-in" character without an interrupting "shift-out" character between them. Any double-byte character can be placed between the matching "shift-out" and "shift-in" characters, (the character string must be an even number of bytes). No further validation is done on the characters between the matching "shift-out" and "shift-in" characters.

Notes:

1. All names you create are stored in either the FDTLIB or the EXTLIB where all trailing blanks are removed.
2. If a DataRefresher Quoted name conforms to the DataRefresher naming conventions, it will lose its quotation marks when it is punched from the FDTLIB or the EXTLIB.

Table of all DataRefresher commands

The following table lists each DataRefresher command in alphabetical order, providing a short description of each command and the page number where the command is explained in detail.

Command	Component	Description	Page
CANCEL	UIM	Cancels an extract request submitted to the DEM for execution.	24
CANCEL	Administrative Dialogs	Ends dialog session for current request without saving panel modifications, and returns to the panel where the dialog was initiated.	226
CANCEL	End User Dialogs	Creates a job to cancel an extract request that has been submitted to the DEM for execution by the End User Dialogs.	230
CHANGE	DEM Operator	Operator command that changes DEM operating characteristics.	156
CHECK	End User Dialogs	Checks to insure that the necessary information for a successful extract has been entered.	231
CONDHALT	DEM Operator	Operator command that temporarily stops a DEM and displays names of extract requests that are executing.	159
CREATE DATATYPE	UIM	Creates a user-defined data type description.	26
CREATE DXTFILE	UIM	Creates a data description for a VSAM or physical sequential data set, or a data source accessed by a generic data interface (GDI) exit.	29
CREATE DXTPSB	UIM	Creates a data description for an IMS PSB.	46
CREATE DXTVIEW	UIM	Creates a DataRefresher view.	61
DCANCEL	Online	Invokes the CANCEL (UIM) command to cancel an extract request.	188
DCREATE	Online	Invokes one of the CREATE commands to create a data description.	191
DDELETE	Online	Invokes the DELETE (UIM) command to delete data descriptions from the FDTLIB.	195
DELETE	UIM	Deletes data descriptions from FDTLIB.	67
DISPLAY	DEM Operator	Operator command that displays current values of a DEM's operating characteristics.	160
DISPLAY	End User Dialogs	Displays a saved extract request.	232
DLIST	Online	Invokes the LIST (UIM) command to list the extract requests that have been submitted to the DEM for execution.	198
DPRINT	Online	Invokes the PRINT (UIM) command to print data descriptions.	202
DPUNCH	Online	Invokes the PUNCH (UIM) command to punch data descriptions.	206
DRUN	Online	Runs the DEM online to process an extract request.	210

Command	Component	Description	Page
DRUNR	Online	Runs the REM online to process a relational extract request.	214
DSEND	Online	Builds an extract request and sends it for execution.	217
DSTATUS	Online	Invokes the STATUS (UIM) command to check the status of an extract request submitted to the DEM for execution.	222
ERASE	End User Dialogs	Erases a saved extract request from the Dialogs library.	233
EXECUTE	DAP	OS/VS DB/DC Data Dictionary command used to invoke DataRefresher's Dictionary Access Program (DAP).	140
GETDEF	UIM	Obtains DataRefresher views from the FDTLIB and makes them available to authorized users in the DataRefresher End User Dialogs environment.	69
HALTBATCH	DEM Operator	Operator command that stops the execution of extract requests.	162
INITDEM	DEM	Initialization command that starts the DEM.	146
LIST	UIM	Requests a list of extract requests that have been submitted to the DEM for execution.	71
PRINT	UIM	Prints data descriptions.	73
PUNCH	UIM	Punches data descriptions.	76
RESET	End User Dialogs	Initializes entry fields to default values.	234
RESUME	DEM Operator	Operator command that restarts the DEM.	163
SAVE	Administrative Dialogs	Interrupts dialog and saves information entered up to that point.	227
SAVE	End User Dialogs	Saves the current extract request in the DataRefresher Dialogs library.	235
SEND	End User Dialogs	Sends an extract request to the UIM or REM for processing.	237
STATUS	UIM	Checks the status of an extract request in the EXTLIB.	80
STATUS	End User Dialogs	Creates a job to check the status of an extract request that has been submitted to the DEM for execution by the End User Dialogs.	240
SUBMIT / EXTRACT	UIM	Creates an extract request to extract data from DataRefresher supported nonrelational sources or sources accessed through GDI exits.	82
SUBMIT / EXTRACT	REM	Creates an extract request to extract data from IBM supported relational databases.	166
TERMINATE	DEM Operator	Operator command that stops the DEM.	164
USE DXTFILE	DEM	Initialization command that specifies the physical sequential or VSAM data set, or generic data interface (GDI) data source that the DEM can access	150

Command	Component	Description	Page
USE DXTPSB	DEM	Initialization command that specifies the DataRefresher PSB the DEM can access.	152
USE EXTID	DEM	Initialization command that specifies a specific extract request that the DEM can access.	154

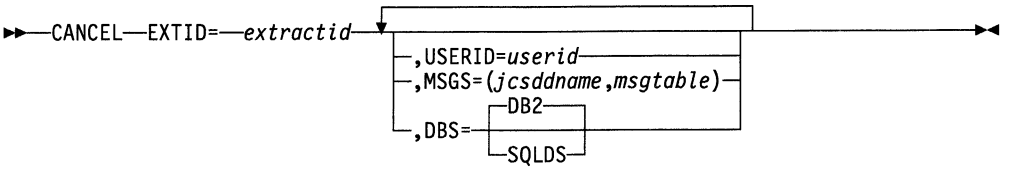
Chapter 5. UIM commands

This chapter contains an alphabetic listing of User Input Manager (UIM) commands. These commands are used to extract data from a non-relational data source.

For more information about using the UIM see the *DataRefresher MVS and VM User's Guide*.

CANCEL

Use the CANCEL command to cancel an extract request that has been submitted to the extract manager for execution, and remove it from the EXTLIB.



CANCEL (REQUIRED)
cancels an extract request.

EXTID = *extractid* (REQUIRED)
specifies the name of the extract request you want to cancel.

Replace *extractid* with the ID of the relevant extract request that was specified on the SUBMIT command.

USERID = *userid* (OPTIONAL)
specifies the user ID of the extract request you want to cancel.

Replace *userid* with the value of the USERID keyword specified on the SUBMIT command. If the USERID keyword was not used on the SUBMIT command, do not include the USERID keyword on the CANCEL command.

MSGS = (*jcsddname,msgtable*) (OPTIONAL)
indicates that you want the messages issued by DataRefresher about your CANCEL command to be sent to a relational message table.

If you do not include the MSGS keyword, the UIM sends messages resulting from your CANCEL command only to the DXTPRINT data set.

Replace *jcsddname* with the name of the data set containing the JCS that loads the messages. This JCS must contain both *CD and *EO statements so that DataRefresher can replace them with the generated load control deck and the messages.

Replace *msgtable* with the name of the relevant message table. The *msgtable* can be written as either *authid.tablename* or *tablename* where:

authid
specifies the authorization ID. If *authid* is not included, the relevant load utility assumes it is the same as the user ID written in the JOB statement of the JCS data set that you use to load your CANCEL messages.

authid must be an alphameric/special characters name.

tablename
specifies the name of the table. It can be either an SQL name or an SQL quoted name.

DBS=DB2 | SQLDS (OPTIONAL)
specifies the database system into which messages from the CANCEL command are loaded. The values you can specify are:

DB2
indicates that you want your messages loaded into a DB2 table.

SQLDS
indicates that you want your messages loaded into an SQL/DS table.

Note: If the MSGS keyword was not included on the CANCEL command, DataRefresher will ignore the DBS keyword.

Examples of CANCEL

Example 1

This example cancels an extract request, assuming that:

- EXTREQ1 is the extract ID, and SMITH is the user ID of the extract request
- DB2JCS is the name of the JCS data set that can route the messages issued by the UIM to the desired DB2 message table
- You want messages issued by the UIM sent to the message table named DXTMSGS
- You want messages from the CANCEL command loaded into the DB2 system

```
CANCEL EXTID=EXTREQ1,USERID=SMITH,
       MSGS=(DB2JCS,DXTMSGS),DBS=DB2;
```

Example 2

This example cancels an extract request, assuming that:

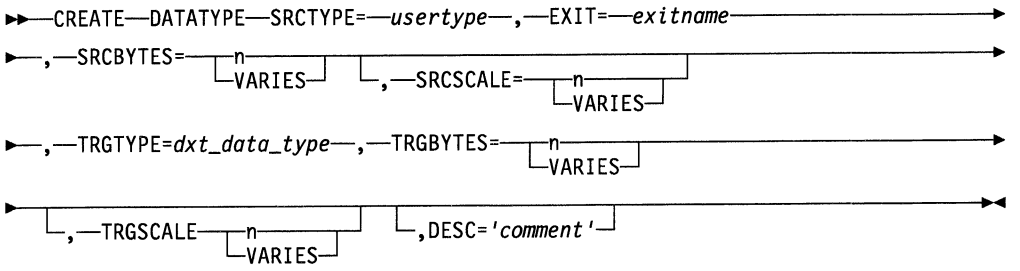
- EXTREQ1 is the extract ID and the extract request does not have a user ID
- SQLJCS is the name of the JCS data set that can load the messages issued by the UIM into the desired message table
- You want messages issued by the UIM sent to the message table named DXTMSGS
- You want messages from the CANCEL command loaded into the SQL/DS system

```
CANCEL EXTID=EXTREQ1,MSGS=(SQLJCS,DXTMSGS),
       DBS=SQLDS;
```

Note: Although Examples 1 and 2 have the same EXTID, they do not have the same USERID and therefore do not cancel the same extract request.

CREATE DATATYPE

Use the CREATE DATATYPE command to create a new user-defined data type description.



CREATE DATATYPE (REQUIRED)
specifies that a user data type is being created.

SRCTYPE=*usertype* (REQUIRED)
specifies a 2 alphameric character value used to uniquely identify this user data type.

Replace *usertype* with a unique value among both standard DataRefresher data types and other user data types.

EXIT=*exitname* (REQUIRED)
identifies the name of the exit routine load module used to convert the source format associated with the data type into the target format supported by DataRefresher.

Replace *exitname* with the name of the user data type exit you want to invoke for this user data type.

SRCBYTES=*n* | VARIES (REQUIRED)
specifies the length (in bytes) of the source field if *n* is specified; or indicates that the number of bytes may vary (VARIES).

The UIM will make a DEF call to the exit when a field is defined using this data type, and the exit will validate the BYTES value specified for the field.

n replace *n* with the length, in bytes, of the fixed length user data type. This value must be a number from 1 through 32767.

VARIES
indicates that the fields of this user data type may have different BYTES values coded in the FIELD statement of the CREATE DXTFILE or CREATE DXTPSB command.

SRCSCALE=*n* | VARIES (OPTIONAL)
specifies the scale of the source field if *n* is specified; or indicates that the scale may vary (VARIES).

The UIM will make a DEF call to the exit when a field is defined using this data type, and the exit will validate the SCALE value specified for the field.

n replace *n* with the scale of the user data type. This value can be an integer from 0 through to 255.

VARIES
indicates that the fields of this user data type may have different SCALE values coded in the FIELD statement of the CREATE DXTFILE or CREATE DXTPSB command.

TRGTYPE=*dxt_data_type* (REQUIRED)
specifies which DataRefresher data type the user data type will be converted to.

Replace *dxt_data_type* with a 1 or 2 character data type:

A for fields containing date data.

B for fields containing single byte binary data.

C for fields containing an EBCDIC character string.

D for fields containing a long (8-byte) floating point number.

E for fields containing a short (4-byte) floating point number.

F for fields containing a 32-bit, signed binary integer.

G for fields containing solely DBCS data.

H for fields containing a 16-bit, signed binary integer.

P for fields containing a packed decimal number.

S for fields containing timestamp data.

T for fields containing time data.

VC for fields containing variable-length character data.

VG for fields containing variable-length graphic data.

Z for fields containing a zoned decimal number.

TRGBYTES=*n* | VARIES (OPTIONAL)
specifies the length (in bytes) of the target field (in converted format) if *n* is specified; or indicates that the number of bytes may vary (VARIES).

A DEF call is made by the UIM to the exit when the field is defined in the CREATE command.

n indicates the length of the target field. Replace *n* with the length of the data that this exit will return.

VARIES
specifies that the maximum target field length will be established for each field using this data type by the DEF call.

TRGSCALE=*n* | VARIES (OPTIONAL)
specifies the scale of the target field (in converted format) if *n* is specified; or indicates that the scale may vary (VARIES). TRGSCALE is only applicable when TRGTYPE is P or Z.

A DEF call is made by the UIM to the exit when the field is defined in the CREATE command.

Fields of type P and Z represent numbers in packed or zoned decimal format, a representation that can be interpreted in many different ways because it does not contain an explicit decimal point.

The values you can specify are:

n if you want the decimal point to be placed to the left of the last *n* digits in the decimal representation. Replace *n* with a positive integer that specifies the scale of the data this exit will return.

The value of *n* can range from 0 through the number of decimal digits that the described field contains. For an *n*-byte field of type Z, the maximum scale is *n*. For an *n*-byte field of type P, the maximum scale is *2n-1*. The default for *n* is 0.

VARIES
indicates that the target scale will be established for each field using this data type by the DEF call.

DESC='*comment*' (OPTIONAL)
saves a comment about the user data type you are creating. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

Example of CREATE DATATYPE

This example creates a user-defined data type that you have named XX. The exit that processes the XX data type is called XXEDIT.

```
CREATE DATATYPE EXIT      = XXEDIT,
                SRCTYPE   = XX,
                SRCBYTES  = 5,
                SRCSCALE  = 4,
                TRGTYPE   = P,
                TRGBYTES  = 10,
                TRGSCALE  = 8,
                DESC      = 'DATA IS IN XX FORMAT';
```

Notes:

1. The value given to the SRCTYPE keyword in the CREATE DATATYPE command should be the same as the value you give to the TYPE keyword in the CREATE DXTFILE or CREATE DXTPSB command.
2. The value you give to the SRCBYTES keyword in the CREATE DATATYPE command should be the same as the value on the BYTES keyword (if specified) in the CREATE DXTFILE or CREATE DXTPSB command.
3. The value given to the SRCSCALE keyword in the CREATE DATATYPE command should be the same as the value on the SCALE keyword (if specified) in the CREATE DXTFILE or CREATE DXTPSB command.

For more information about user data types and user data type exits, see *DataRefresher Exit Routines*.

CREATE DXTFILE

Use the CREATE DXTFILE command to create a data description for a VSAM or physical sequential data set, or a data source accessed by a GDI exit. You create DXTFILE descriptions to describe:

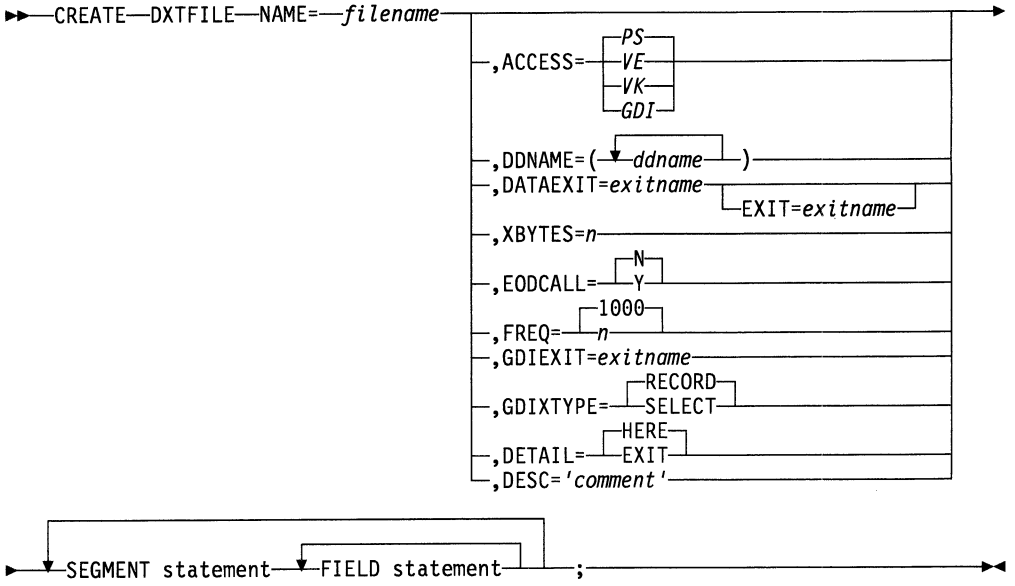
- Simple files (with one record type)
- Structured files (with multiple record types or internal segments)

To create a DXTFILE description of a simple file, use CREATE DXTFILE with FIELD statements only.

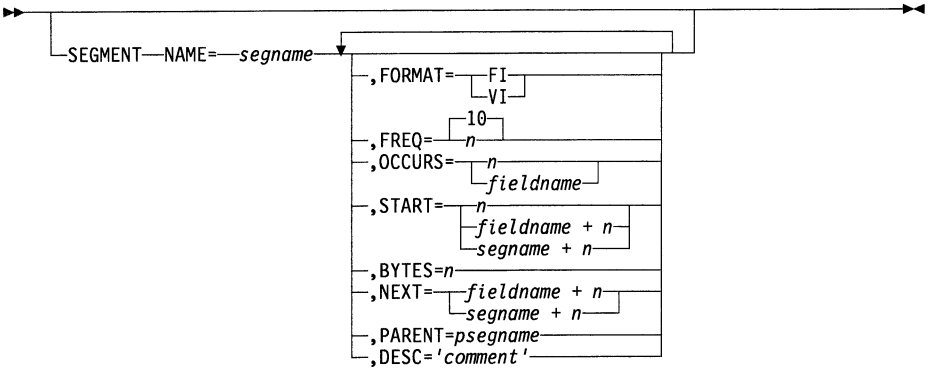
To create a DXTFILE description of a structured file, use CREATE DXTFILE with both SEGMENT and FIELD statements.

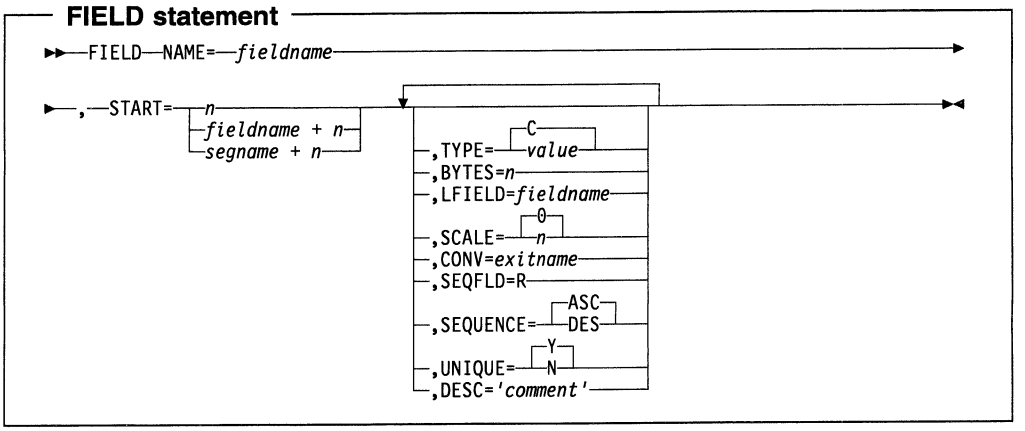
You can also specify GDI exits in your DXTFILE descriptions to describe data sources accessed by these exits.

The maximum length of a record you can define is 512K.



SEGMENT statement





CREATE DXTFILE (REQUIRED)
specifies that a DXTFILE description is being created.

NAME=*filename* (REQUIRED)
specifies the name to be assigned to this DXTFILE description. Make the name unique among all DXTFILE description names in the FDTLIB. This name can be a DataRefresher name, DataRefresher quoted name, or DBCS name. See "DataRefresher naming conventions" on page 18 for more information. DataRefresher supports up to 32 characters (excluding double quotes) for file names. If your organization uses RACF* to protect its DataRefresher data items, give this *filename* to the RACF administrator.

ACCESS=PS | VE | VK | GDI (OPTIONAL)
indicates the file organization.

The default for ACCESS is PS.

PS physical sequential data set

VE VSAM ESDS

VK VSAM KSDS or VSAM ESDS with an alternate index identified as the sequence field by the FIELD statement

GDI data source that will be accessed through a GDI exit

DDNAME=(*ddname*) (OPTIONAL)
specifies the names of the DD statements for all data sets used during UIM and/or DEM processing.

The names you specify can be alphamerics with special character as described in "DataRefresher naming conventions" on page 18. If you specify more than one DD name, enclose all DD names in parentheses.

VSAM and physical sequential data sources: For VSAM and physical sequential data sources, DDNAME specifies the name of the DD statement for the data set you want the DEM to use when extracting data associated with this DXTFILE description. Only one DD name can be specified with VSAM and physical sequential files, and it must have a corresponding DD

---

* RACF is a trademark of the International Business Machines Corporation.

** VMS is a trademark of the Digital Equipment Corporation.

statement in the JCL for starting the DEM. Be sure to communicate this DD name to the person who writes the DEM JCL.

Note: You can also use the USE DXTFILE (DEM) command to:

- Specify the DD name if not specified with CREATE DXTFILE
- Override the DD name specified with CREATE DXTFILE

GDI exit data sources (local sources only): For GDI exit data sources, DDNAME specifies the names of DD statements for the data sets that the GDI exit needs when it is started by either the UIM or DEM.

You may not want to use DDNAME for GDI exit data sources when:

- There are no data sets that need to be allocated through UIM or DEM JCL for the GDI user exit.
- The names of the required DD statements are coded in the GDI user exit; therefore, they do not need to be passed from DataRefresher to the GDI user exit. (Appropriate DD statements would still be required in the UIM or DEM JCL for this case.)
- The GDI user exit dynamically allocates the data sets that are accessed.

You can specify up to 8 DD names for GDI data sources. Specify a DD name when the GDI user exit needs to access definition information from one or more data sets. This situation applies to:

- GDI select exits
- GDI record exits when you specify DETAIL=EXIT

The DD names must have corresponding DD statements in the JCL for starting the UIM. Be sure to communicate these DD names to the person who writes the UIM JCL.

You can also use DDNAMES to specify the names of the DD statements for the data sets that you want the DEM (through the GDI user exit) to use when extracting data associated with this DXTFILE description. The DD names you specify must have corresponding DD statements in the JCL for starting the DEM. Be sure to communicate these DD names to the person who writes the DEM JCL.

Note: You can also use the USE DXTFILE (DEM) command to:

- Specify the DD names if not specified with CREATE DXTFILE
- Override the DD names specified with CREATE DXTFILE

DATAEXIT=*exitname* / **EXIT=***exitname* **(OPTIONAL)**
is used when each record of the file must be interpreted or changed by a data exit before DataRefresher can extract data from it.

Replace *exitname* with the name of the data exit you want invoked. It must match the name of the load module containing the data exit you want to use.

You must specify XBYTES with either DATAEXIT or EXIT.

Do not use DATAEXIT or EXIT for GDI select exits or segments.

XBYTES=*n* (REQUIRED if DATAEXIT or EXIT is specified)
specifies the length (in bytes) of the output record that will be returned by a data exit to DataRefresher for a data extract.

Replace *n* with the length, in bytes, of the record *after* processing by the data exit. The value of *n* can be any integer from 1 through 32760. For variable-length constructs, use this keyword to specify the maximum length, in bytes, of the record *after* processing by the data exit.

Do not use XBYTES for segments.

EODCALL=N | Y (OPTIONAL only when DATAEXIT | EXIT is specified)
requests that DataRefresher make an end-of-data call to the data exit.

End-of-data calls, if requested, are made in addition to the normal calls for segment or record occurrences, and are useful if the exit is doing summarization.

For records from a DXTFILE, end-of-data means at the end of the file or database, subject to any search strategy.

FREQ=*n* (OPTIONAL)
optimizes access strategies by using an estimate of the expected frequency of records in a DXTFILE. The value of *n* can be any number from .01 through 16777215.00 (without commas). Examples of permitted values are:

```
56
25.00
3.6
99.12
```

All FREQ values output by the DAP contain two decimal positions.

The default value for FREQ is 1000.

GDIEXIT=*exitname* (REQUIRED if ACCESS=GDI)
specifies the name of your GDI exit. GDIEXIT is required when you specify ACCESS=GDI. Depending on how the GDI exit is started, this can either be the:

- Load module name
- Entry point name of the GDI exit.

GDIXTYPE=RECORD | SELECT (OPTIONAL)
specifies whether your GDI exit is a select exit or record exit.

You must also specify ACCESS=GDI.

DETAIL=HERE | EXIT (OPTIONAL)
This keyword is only used for GDI record exits. It specifies whether the field and segment descriptions are defined in the FIELD and SEGMENT statements of the CREATE DXTFILE statement (HERE), or through the describe call to the GDI exit (EXIT).

You must also specify ACCESS=GDI.

DESC='*comment*' (OPTIONAL)
saves a comment about the data description you are creating. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

SEGMENT (OPTIONAL)
is specified to:

- Define segments in a structured file with multiple record types.
- Define internal segments in a structured file, where the file may or may not have multiple record types.

Specify one SEGMENT for each record type or internal segment (repeating group of data within its parent segment) you want to describe.

A segment that contains internal segments or nested internal segments is called a parent segment. A parent segment can have up to 16 nested internal segments.

You cannot specify a data exit for a segment. If you do, DataRefresher will return an error message.

Segments can be fixed or variable in length and can contain variable-length fields. These variable-length fields can be either variable-length character (VC) or variable-length graphic (VG) data types.

SEGMENT statements for structured files with multiple record types but no internal segments can be written in any order. (The FIELD statements for a particular SEGMENT must directly follow it.)

Internal segments in a structured file impose a hierarchy like that of an IMS/VS DL/I database. Therefore, you must write the physical and internal segments in hierarchical order (that is, write SEGMENT values from top to bottom and left to right). For example, you must include SEGMENT values for the parent containing segment and SEGMENT values for each of the internal segments in the hierarchy.

Because internal segments impose a hierarchy in a file and physical segments do not, the DXTVIEWs based upon these files are different. When there are internal segments, the DXTVIEW defines a path; when there are no internal segments, the DXTVIEW defines the segment as a record type.

For more information on describing segments in a DXTFILE description, see also the *DataRefresher MVS and VM User's Guide.*

NAME=*segname* (REQUIRED with SEGMENT)
specifies the unique name that identifies the segment or internal segment.

The name must be a DataRefresher name or DataRefresher quoted name as described in "DataRefresher naming conventions" on page 18, and must be different from the names assigned by SEGMENT values to the other segments in the DXTFILE description. DataRefresher supports up to 32 characters (excluding double quotes) for segment and internal segment names.

FORMAT=FI | VI (REQUIRED only for internal segments)
indicates whether an internal segment is a fixed-length internal (FI) or variable-length internal (VI) segment.

FREQ=*n* (OPTIONAL)
specifies the estimated average frequency of occurrences of the internal segment within the parent segment. For internal segments, if OCCURS is specified, FREQ defaults to that value; otherwise it defaults to 10.

OCCURS=*n* / *fieldname* (REQUIRED for internal segments only)
indicates how many occurrences of the internal segment there are in the parent segment. The values associated with OCCURS can be:

n specifies the fixed number of occurrences of the internal segment.

fieldname
specifies the name of the field that contains the (variable) value of the number of occurrences of an internal segment. The field must exist in the parent of the internal segment and must physically precede the internal segment.

START=*n* / *fieldname* + *n* / *segname* + *n* (REQUIRED for internal segments only)
specifies the starting position of an internal segment in the parent segment.

n is used for a fixed starting position. Replace *n* with the fixed starting position of the internal segment in the parent segment. The value of *n* must be positive.

fieldname + *n*
is used for a variable starting position. Replace *fieldname* with the name of a field that exists in the parent segment of the internal segment being described. Replace *n* with the number of bytes from the end of the named field to the starting position of this internal segment. The value of *n* must be positive. If the named field and this internal segment are adjacent, write *fieldname* + 1.

segname + *n*
is also used for a variable starting position. Replace *segname* with the name of a segment that has the same parent segment as the internal segment being described. Replace *n* with the number of bytes from the end of the named segment to the starting position of this internal segment. The value of *n* must be positive. If the named segment and the next occurrence of the internal segment are adjacent, write *segname* + 1.

BYTES=*n* (OPTIONAL)
provides the length of a fixed length internal segment. Replace *n* with the length in bytes of the fixed-length internal segment.

When defining fixed-length internal segments, you must specify either BYTES or NEXT; do not specify both.

NEXT=*fieldname* + *n* / *segname* + *n* (REQUIRED for variable-length internal segments)
specifies how to step from one occurrence of an internal segment to the next by specifying the starting location of the next occurrence of the internal segment.

NEXT can also be used for a fixed-length internal segment instead of the BYTES keyword. If you specify BYTES, you cannot specify NEXT.

fieldname + *n*
specifies the name of a field (*fieldname*) within the internal segment being defined and the number of bytes (*n*) from the end of the named field to the next occurrence of the same internal segment. The value of

n must be positive. If the named field and the next occurrence of the internal segment are adjacent, write *fieldname* + 1.

segname + n
specifies the name of any segment (*segname*) being defined with this DXTFILE and the number of bytes (*n*) from the end of it to the next occurrence of this internal segment. The value of *n* must be positive. If the named segment and the next occurrence of the internal segment are adjacent, write *segname* + 1.

PARENT=*psegname* (REQUIRED for internal segments only)
names the parent segment of the internal segment. Replace *psegname* with the name of the parent segment.

DESC=*comment* (OPTIONAL)
saves a comment about the segment you are describing. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

FIELD (REQUIRED if GDIEXIT is not specified)
creates DXTFILE descriptions of simple and structured files. You can describe the fields in any order.

Include one FIELD keyword for each field you want to include in your DXTFILE description. The fields you describe may overlap, that is, all or part of one field may be contained in another field. Not all of the space in the record must be described (the record can have gaps).

When you create a DataRefresher file description of a structured file, you must follow each SEGMENT with one or more FIELDS , unless you have internal segments which are nested. Except in the case of internal segments, the field descriptions in one segment are independent of those in every other segment. Therefore, you can designate a sequence field for each segment, and you can give two fields in different segments the same name.

You can use FIELD keywords to describe fixed and variable-length fields. Variable-length fields can be of either variable-length character (VC) or variable-length graphic (VG) data types. You can tell DataRefresher the length of the variable-length field by specifying a length field with LFIELD. (See the LFIELD keyword for more information on length fields.)

You cannot define a variable-length field as a sort field or sequence field.

If you use a data exit routine to preprocess records in your source data, use FIELD to describe the fields in the record as they will appear *after* processing by the exit. There is an exception. If the field is the key field of a VSAM KSDS, the FIELD for the key must describe the field as it looks in storage so that DataRefresher can use it in its search strategy. If your exit attempts to change the value of a key field, the DEM suspends processing of the current batch of extract requests and queues them again for later processing.

Note: You cannot specify a FIELD statement for a GDI select exit, or if you specify DETAIL=EXIT for a GDI record exit. For a GDI select exit, the description of the extract source must be provided by the exit. For a GDI record exit, when you specify DETAIL=EXIT, the fields and segments are defined through a describe call to the GDI exit.

NAME=*fieldname* (REQUIRED)
assigns the name to the field being described. The name can be a DataRefresher name, a DataRefresher quoted name, or a DBCS name, as described in "DataRefresher naming conventions" on page 18. Field names can be up to 32 characters, excluding double quotes but including shift-out and shift-in characters for DBCS names. If the name you choose is a DBCS name, the name must contain *only* DBCS characters. No single-byte characters are permitted. The name must be bracketed by a shift-out (X'0E') and shift-in (X'0F') character, as in this example:

```
'<F_L_D_>'
```

(X'0E' is represented by < and X'0F' by >.)

START=*n* / *fieldname* + *n* / *segname* + *n* (REQUIRED)
describes the starting position of a field in a record or segment. The start position is relative to the beginning of the segment.

n is used for a fixed starting position. Replace *n* with

- The start position of a field in a record for a simple file
- The field within the segment or internal segment for a structured file

The value of *n* must be positive.

fieldname + *n*
is used for a variable starting position. Replace *fieldname* with the name of a field that exists on the same level as the field being described. For a simple file this can be any field; for a structured file, the field must exist in the same segment as the one being described. Replace *n* with the number of bytes from the end of the named field to the starting position of this field. The value of *n* must be positive. If the named field and this field are adjacent, write *fieldname* + 1.

segname + *n*
If the segment to which this field belongs contains internal segments, and the field physically follows an internal segment, then you can replace *segname* with the name of that internal segment.

Replace *n* with the number of bytes from the end of the named segment to the start of this field. The value of *n* must be positive.

If the named segment and this field are adjacent, write *segname*+1.

If you are describing a segment with a variable-length format, be sure to account for the 2-byte length field at the start of the segment. For example, a value of START=1 means it begins at the start of the length field, not at the byte following the length field.

If you are describing a physical sequential data set with variable-length records (RECFM = V or VB), remember that the first four bytes of each record contain a system-generated length indicator.

TYPE=*value* (OPTIONAL)
indicates the data type of the field being described.

KSDS key fields are of type character (**C**), graphic (**G**), fullword (**F**), halfword (**H**), packed decimal (**P**), or zoned decimal (**Z**).

If you are writing a DXTFILE description that describes a user-defined data type, the TYPE of the field must match the SRCTYPE that was specified on the CREATE DATATYPE command.

The values of TYPE include:

A date data.

B single byte binary data.

C EBCDIC character string.

D long (8-byte) floating point number.

E short (4-byte) floating point number.

F 32-bit, signed binary integer.

G solely DBCS data.

H 16-bit, signed binary integer.

P packed decimal number.

S timestamp data.

T time data.

VC variable-length character data.

VG variable-length graphic data.

xx a user-defined data type (*xx* gets replaced with the name of the use data type).

Z zoned decimal number.

BYTES=*n* (REQUIRED for field lengths not determined by data type)
specifies how many bytes a field occupies or the maximum number of bytes a field can be if it is a variable-length (VC or VG) data type. BYTES must be specified for fields of type A, C, P, S, T, Z, VC, and VG. Specify *n* in bytes. If you specify that the DEM start a data exit to preprocess source records (using the DATAEXIT or EXIT keyword), the value of BYTES must indicate the length of the field *after* processing by the exit.

If you are writing a DXTFILE description that describes a user-defined data type, the value you code here depends on the value you coded on the SRCBYTES keyword of the CREATE DATATYPE command. If SRCBYTES has a fixed number on the CREATE DATATYPE command, you do not need to include the BYTES keyword on the CREATE DXTFILE command. If, however, VARIES was specified for SRCBYTES on the CREATE DATATYPE command, then you must specify the length of this field with the BYTES keyword here.

When you specify BYTES for fields whose lengths are determined by data type, specify the correct length as shown here:

Data Type	Exact Length
B	1
D	8
E	4
F	4
H	2
xx	value of SRCBYTES keyword on CREATE DATATYPE command

When you specify BYTES for fields whose lengths are not determined by data type, specify a length in the ranges shown here:

Data Type	Minimum Length	Maximum Length
S	16	26
A	1	10
C	1	254
G	2	254[1]
P	1	16
T	1	8
VC	1	32767
VG	2	32766[2]
Z	1	16
[1]127 DBCS characters, BYTES must be even		
[2]16383 DBCS characters, BYTES must be even		

Note: VG and G data types represent double-byte character set (DBCS) data; therefore you must use two bytes to represent one DBCS character when you calculate the length or starting position of an internal segment. For more information, see Appendix D, "Coding the Description" on page 255.

LFIELD=*fieldname* (REQUIRED for a variable-length field in a variable-length internal segment; OPTIONAL otherwise)
names the field that contains the actual length of a variable-length field (field type of VC or VG). This length field must physically precede the variable-length field.

The value of the field named with LFIELD must be an even number if the field is defined as a VG data type. The length field must be of numeric type data, and must have a zero scale if it is packed or zoned decimal; it cannot be floating point. A length field can have a zero value, which means that in this particular record, the variable-length field is not present.

If you do not specify a length field for a variable-length field in a fixed-length segment, DataRefresher assumes that the length of the field is

'whatever is left' in the record. That is, its length is equal to the record length plus one, minus the starting position of the field.

LFIELD cannot be specified for user data types.

SCALE=*n* | 0 (OPTIONAL)
is used only for fields of type P or Z, or for a user-defined data type, to indicate the position of the decimal point in the numbers that the field represents. Fields of type P and Z represent numbers in packed or zoned decimal format, a representation that can be interpreted in many different ways because it does not contain an explicit decimal point.

If you are writing a DXTFILE description that describes a user-defined data type, the value you code here depends on the value you coded on the SRCSCALE keyword of the CREATE DATATYPE command:

- If SRCSCALE has a fixed number on the CREATE DATATYPE command, you do not need to include the SCALE keyword on the CREATE DXTFILE command. If you do include it here, the values for SRCSCALE and SCALE must be the same.
- If, however, VARIES was specified for SRCSCALE on the CREATE DATATYPE command, then you must specify the scale of this field with the SCALE keyword here.

The values you may specify are:

n for the decimal point to be placed to the left of the last *n* digits in the decimal representation. Replace *n* with a positive integer.

The value of *n* can range from 0 through the number of decimal digits that the field being described contains. For an *n*-byte field of type Z, the maximum scale is *n*. For an *n*-byte field of type P, the maximum scale is *2n-1*.

0 for no decimal point in the number. (If the value of the field is assumed to be an integer.)

CONV=*exitname* (OPTIONAL)
specifies the name of a date/time conversion exit that converts date or time data types into standard ISO format. The CONV parameter cannot be specified for any data type other than date or time. See *DataRefresher Exit Routines* for more information on writing date/time conversion exits.

SEQFLD=R (OPTIONAL)
identifies this field as a key field so that the DEM or GDI exit can process extract requests more efficiently. Include SEQFLD=R only if the field being described is the major sort/key field of the file. Therefore, one of the following:

- The primary key field or an alternate key field in a VSAM KSDS
- An alternate key field in a VSAM ESDS
- The major sort field in a physical sequential data set or VSAM ESDS
- The primary key field in an internal segment occurrence
- The sequence or key field in a GDI record file

Do not write SEQFLD=R for more than one field unless the fields in question are in separate segments of the DXTFILE description. If the DXTFILE

description is of a structured file, you can designate a sequence field for each segment, which normally should be the same field or refer to the same area. Nominating a sequence field means that the DEM or GDI record exit being used to process the extract request can do so more efficiently.

SEQUENCE=ASC | DES (OPTIONAL)
indicates whether the values in the field being defined appear in ascending or descending order. SEQUENCE can only be used in conjunction with the SEQFLD. You need not use this parameter for a VSAM KSDS, but if you do, the sequence must be ascending. For a VSAM ESDS or a physical sequential data set, the values you can specify are:

ASC
for ascending values.

DES
for descending values.

UNIQUE=Y | N (OPTIONAL)
indicates whether the field being defined has unique values for all of its records. UNIQUE can only be used if SEQFLD is also specified.

Y for fields that contain unique values.

N for fields that do not contain unique values.

DESC='*comment*' (OPTIONAL)
saves a comment about the field you are describing. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

Note: DataRefresher has a maximum number of fields and segments that can be defined in each data description. This maximum number is 1530. So, for example, if you have a file with 30 segments, it can have a maximum of 1500 fields.

Examples of CREATE DXTFILE

Example 1

This example accesses data from a file in IXF with a GDI record exit. You specify DETAIL=EXIT which indicates that the details of the fields and segments are defined through a describe call to the GDI record exit; therefore you do not write SEGMENT and FIELD statements. In this example, GDIXTYPE defaults to RECORD. The GDI record exit:

- Reads the IXF records
- Translates the IXF header records into a DataRefresher data definition during the *describe* call
- Returns a row of data for each fetch call

```
CREATE DXTFILE
             NAME = QMFIXF2A,
             DESC = 'QMF QUERY 2A RESULT, EXPORTED IN IXF FORMAT',
             FREQ = 250,
             DDNAME = Q2A,
             ACCESS = GDI,
             GDIEXIT = GDIXFM,
             DETAIL = EXIT;
```

Example 2

This example creates a DXTFILE description of a structured file: a VSAM data set with multiple record types and no internal segments.

Assume the following:

- The DXTFILE description = INVNTRY and its two segments = PRODSEG and WHSESEG.
- The six fields in the PRODSEG segment are named: INUM, ISFX, INAME, IGRP, IPRICE, and the key field IKEY (all of which are contained in the PRODSEG records).
- Describe five fields in the WHSESEG segment: INUM, LOCN, QONHAND, QONORD, and the key field IKEY (all of which are contained in the WHSESEG records). IKEY, the sequence field for records of either type, appears in both the PRODSEG and the WHSESEG segments.
- Notify DataRefresher that you wrote a data exit to interpret a record of this source data before the data is extracted. The exit, WHSEXIT, interprets the numeric value in the 2-byte LOCN field of the WHSESEG segment, changing it to a 10-byte character field containing the name of the city where the warehouse is located.

```
CREATE
DXTFILE NAME=INVNTRY, ACCESS=VK, DDNAME=IFILE,
        EXIT=WHSEXIT,XBYTES=29, FREQ=500,
        DESC='DATA DESCRIPTION OF A STRUCTURED FILE'
/* THE FILE OPERANDS APPLY TO BOTH SEGMENTS */
  SEGMENT NAME=PRODSEG,
  DESC='SEGMENT FOR THE PROD RECORDS'
    FIELD NAME=INUM,   START=1,  BYTES=3,
          DESC='CONTAINS PRODUCT''S NUMBER'
    FIELD NAME=ISFX,   START=4,  BYTES=2,
          DESC='SUFFIX OF RECORD KEY'
    FIELD NAME=INAME,  START=6,  BYTES=11,
          DESC='DESCRIPTION OF ITEM'
    FIELD NAME=IGRP,   START=17, BYTES=10,
          DESC='NAME OF PRODUCT GROUP'
    FIELD NAME=IPRICE, START=27, BYTES=3,
          TYPE=P, SCALE=2,
          DESC='UNIT SELLING PRICE'
    FIELD NAME=IKEY,   START=1,  BYTES=5,
          SEQFLD=R,    SEQUENCE=ASC, UNIQUE=Y,
          DESC='CONTAINS INUM AND ISFX'
  SEGMENT NAME=WHSESEG,
          DESC='SEGMENT FOR THE WHSE RECORDS'
    FIELD NAME=INUM,     START=1,  BYTES=3,
          DESC='CONTAINS PRODUCT''S NUMBER'
    FIELD NAME=LOCN,     START=4,  BYTES=10,
          DESC='SUFFIX OF RECORD KEY'
    FIELD NAME=QONHAND,  START=16, BYTES=2, TYPE=H,
          DESC='ITEMS STORED IN WAREHOUSE'
    FIELD NAME=QONORD,   START=18, BYTES=2, TYPE=H,
          DESC='ADDITIONAL ITEMS ORDERED'
    FIELD NAME=IKEY,     START=1,  BYTES=5,
          SEQFLD=R, SEQUENCE=ASC, UNIQUE=Y,
          DESC='CONTAINS INUM AND LOCN';
```

Note: There are two unused bytes between the field LOCN and the field QONHAND.

Example 3

This example creates a DXTFILE description for a structured file: a VSAM data set with an internal segment and a variable-length field. Assume you want to:

- Create and name a DXTFILE description of a structured file, DXTACCNT, and name its two segments DEPT and ACCOUNT.
- Describe four fields in the DEPT segment: DEPT_NO, DEPT_NBR_ACCTS, DEPT_NAME_LEN, and DEPT_NAME. The variable-length field DEPT_NAME follows the one-byte binary length field DEPT_NAME_LEN. The DEPT_NBR_ACCTS field specifies the number of ACCOUNT segment occurrences in a particular record.
- Describe three fields in the ACCOUNT segment: ACC#, ACC_OP, and ACC_CL. The ACCOUNT segment is a fixed-length internal segment whose number of occurrences varies from record to record. Since the ACCOUNT segment follows the DEPT_NAME field (a variable-length field in the DEPT segment), the start position of the ACCOUNT segment is relative to the end of the DEPT_NAME field. The start position of each field in the ACCOUNT segment is relative to the beginning of the ACCOUNT segment. Each occurrence of the ACCOUNT segment will result in one output row.

```
CREATE
DXTFILE NAME=DXTACCNT, ACCESS=VE, DDNAME=AFILE,
        DESC='FILE WITH VAR LENGTH FIELD AND INTERNAL SEGMENT'
  /* DESCRIBES DEPARTMENT SALES SEGMENT */
  SEGMENT NAME=DEPT, DESC='DEPARTMENT SALES INFO'
    FIELD NAME=DEPTNO,         START=1,  TYPE=C,  BYTES=2
    FIELD NAME=DEPT_NBR_ACCTS, START=3,  TYPE=H
    FIELD NAME=DEPT_NAME_LEN,  START=5,  TYPE=B
    FIELD NAME=DEPT_NAME,      START=6,  TYPE=VC, BYTES=20,
          LFIELD=DEPT_NAME_LEN
  /* DESCRIBES CUSTOMER ACCOUNT SEGMENT */
  SEGMENT NAME=ACCOUNT, FORMAT=FI, BYTES=8, OCCURS=DEPT_NBR_ACCTS,
          START=DEPT_NAME+1, DESC='CUSTOMER ACCOUNT INFO',
          PARENT=DEPT
    FIELD NAME=ACC#,    START=1,  TYPE=C,  BYTES=4
    FIELD NAME=ACC_OP,  START=5,  TYPE=H
    FIELD NAME=ACC_CL,  START=7,  TYPE=H
    ;
```

Example 4

This example creates a DXTFILE description of a simple file: a VSAM data set with only one record type.

Assume the following:

- The DXTFILE description = PRODUCTS.
- The six fields in the DXTFILE description are: INUM, ISFX, INAME, IGRP, IPRICE, and IKEY.
- The sixth field (IKEY) has a field of type C and of length 5. IKEY begins at the start of each record, is the sequence key for the records, and has unique, ascending values. (IKEY is a concatenation of the INUM and the ISFX fields.)

```
CREATE
DXTFILE
  NAME=PRODUCTS,   /* NAME OF DESCRIPTION         */
  ACCESS=VK,       /* VSAM KSDS                  */
  DDNAME=PRODUCTS, /* DEFAULT DD-STATEMENT       */
                   /* NAME FOR SOURCE FILE       */
  FREQ=500,        /* EXPECTED FREQUENCY OF      */
                   /* RECORDS                    */
  DESC='DATA DESCRIPTION OF A SIMPLE FILE'
FIELD
  NAME=INUM,       /* NAME OF FIELD              */
  START=1,         /* STARTING POSITION          */
  BYTES=3,         /* FIELD LENGTH               */
                   /* TYPE DEFAULTS TO 'C'       */
  DESC='CONTAINS THE PRODUCT''S NUMBER'
FIELD NAME=ISFX,   START=4,  BYTES=2,
      DESC='SUFFIX OF RECORD KEY'
FIELD NAME=INAME,  START=6,  BYTES=11,
      DESC='DESCRIPTION OF ITEM'
FIELD NAME=IGRP,   START=17, BYTES=10,
      DESC='NAME OF PRODUCT GROUP'
FIELD NAME=IPRICE, START=27,
      TYPE=P,      /* PACKED DECIMAL FIELD       */
      BYTES=3,     /* FIVE DECIMAL DIGIT CAPACITY */
      SCALE=2,     /* REPRESENTING NUMBERS       */
                   /* OF THE FORM XXX.XX         */
      DESC='UNIT SELLING PRICE'
FIELD
  NAME=IKEY,
  START=1,  BYTES=5, /* INCLUDES INUM AND ISFX    */
  SEQFLD=R, SEQUENCE=ASC, UNIQUE=Y,
  DESC='CONTAINS FIELDS INUM AND ISFX';
```

Example 5

This example accesses data in a DB2 table with a GDI select exit. The GDI select exit:

- Uses dynamic SQL to interface with DB2
- Can handle any SQL statement
- Can use the DB2 describe interface to generate information for DataRefresher's *describe* call

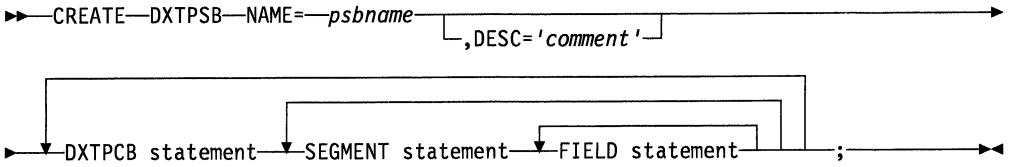
```
CREATE DXTFILE
       NAME = EMPTABLE,
       DESC = 'DB2 EMPLOYEE TABLE',
       ACCESS = GDI,
       GDIEXIT = GDIDB2S,
       GDIXTYPE = SELECT;
```

Notes:

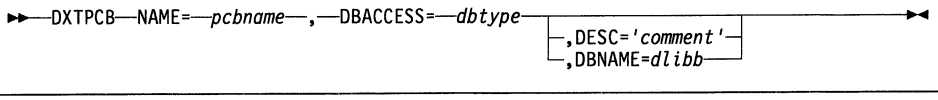
1. DETAIL is not allowed for GDI select files.
2. Field definitions are not allowed for a GDI select exit; the description of the extract source must be provided by the exit.
3. A DXTVIEW named EMPTABLE is automatically generated by DataRefresher. (DataRefresher automatically generates DXTVIEWs for GDI select exits.)

CREATE DXTPSB

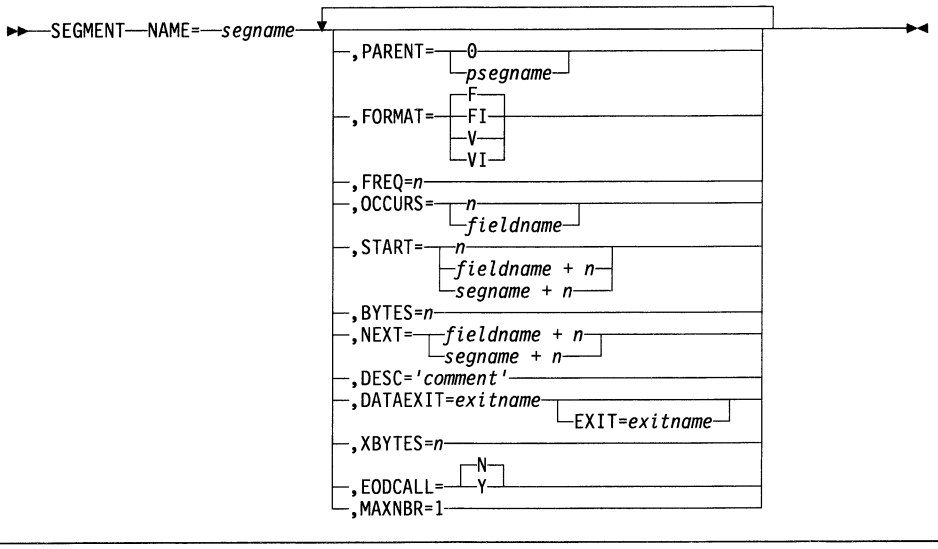
Use the CREATE DXTPSB command to create DXTPSB descriptions of IMS databases. Data descriptions are required only for DEM extract requests.

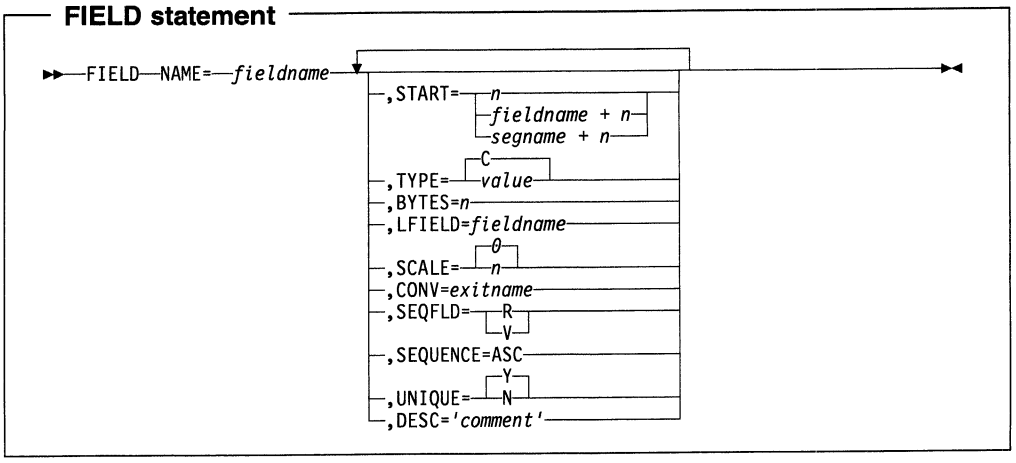


DXTPCB statement



SEGMENT statement





CREATE DXTPSB (REQUIRED)
specifies that a DXTPSB description is being created.

NAME=*psbname* (REQUIRED)
assigns the name to the DXTPSB description. Replace *psbname* with the name. The name must be an alphameric/special characters name, as described in "DataRefresher naming conventions" on page 18, and must be unique among all DXTPSB names in the FDTLIB.

The name of your DXTPSB can match the name of its corresponding IMS/VS PSB, but it does not need to. Therefore, you can include any number of different descriptions of the same PSB in an FDTLIB. If your site uses RACF to protect its DataRefresher data items, give this *psbname* to the RACF administrator.

Be sure to give the PSB name to the person who writes the JCL for running the DEM. That way, the DEM can process extract requests that reference the DXTPSB. If your site uses RACF to protect its DataRefresher data items, you must also give the PSB name to the RACF administrator.

DESC='*comment*' (OPTIONAL)
saves a comment about the data description you are creating. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

DXTPCB (REQUIRED)
describes database PCBs. You can write more than one PCB for the PSB you are describing. Write them in the order in which their PCBs are defined in the IMS PSB.

NAME=*pcbname* (REQUIRED)
assigns a name to the PCB being described. Replace *pcbname* with the name. The name must be an alphameric/special characters name, as described in "DataRefresher naming conventions" on page 18, and must be unique among the names assigned to the PCBs included in this DXTPSB description.

DBACCESS=*dbtype* (REQUIRED)
describes the type of database. The value of DBACCESS depends on the organization of the physical database that contains the root segment defined for the PCB.

If it is a physical database, write the IMS/VS access method that IMS/VS DL/I uses to retrieve data from it.

If it is a logical database, write the IMS/VS access method for the physical database that contains the logical root segment. The values you can specify are:

HSAM
Hierarchical Sequential Access Method

SHSAM
Simple Hierarchical Sequential Access Method

HISAM
Hierarchical Indexed Sequential Access Method

SHISAM
Simple Hierarchical Indexed Sequential Access Method

HIDAM
Hierarchical Indexed Direct Access Method

Note: Also used for HDAM with Secondary Index.

HDAM
Hierarchical Direct Access Method

Note: To use a Secondary Index with an HDAM database, use HIDAM.

MSDB
Main Storage databases (Fast Path)

DEDB
Data Entry databases (Fast Path)

HSSR
High Speed Sequential Retrieval

DESC='*comment***' (OPTIONAL)**
saves a comment about the data description you are creating. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

DBNAME=*dlidb* **(OPTIONAL)**
specifies the DL/I database name as defined in your DBD.

DBNAME can be an alphameric name from 1 to 8 characters. The default for DBNAME is the DXTPCB name.

SEGMENT (REQUIRED)
is used to tell DataRefresher of the hierarchical order of IMS segments (also called database segments) and internal segments (repeating groups within the database segments).

Use one SEGMENT for each database segment and internal segment you want to describe in the DXTPCB. For an MSDB PCB, you can write one database segment and multiple internal segments. For a DEDB PCB, you can write up to 127 database segments; for all other PCBs, you can write up to 255 segments.

Internal segments and database segments can be fixed or variable in length and can contain variable-length fields. These variable-length fields can be either variable-length character (VC) or variable-length graphic (VG) data types.

A segment that contains an internal segment or nested internal segments or that has a child database segment is a parent segment. A parent segment can have up to 16 nested internal segments.

Write the database segments and internal segments in hierarchical order (from top to bottom and left to right). You must include a SEGMENT keyword for the root segment, each database segment, and each internal segment you describe in the hierarchy.

For internal segments with variable start positions (specified by NEXT and START), you must consider any referenced parent segments or fields containing VG and G data types. VG and G data types represent double-byte character set (DBCS) data; therefore you must use two bytes to represent one DBCS character when you calculate the starting position of the internal segment. See Appendix F, "Output data records and fields" on page 259 for more information about VG and G type data.

NAME=*segname* (REQUIRED)
is used to name both database segments and internal segments.

For database segments, replace *segname* with the IMS/VS name of the segment type, that is, the name that was used for the segment in the DBD generation.

For internal segments, replace *segname* with any name by which DataRefresher will identify the internal segment.

The segment name must be unique to the DXTPCB. It must be different from all other segment names in the DXTPCB. The name must be a DataRefresher or DataRefresher quoted name, as described in "DataRefresher naming conventions" on page 18. DataRefresher permits up to 8 characters (excluding double quotes) for DataRefresher segment names for PCBs and up to 32 characters for internal segments. (Database segment names must match IMS segment names which are a maximum of 8 characters.)

PARENT=*0 / psegname* (REQUIRED)
indicates the parent of the segment. For database segments, the parent must be a database segment; for internal segments the parent segment is either a database segment or another internal segment. The values you can specify are:

0 if the segment being described is a root segment

psegname
if the segment is not a root segment. Replace *psegname* with the name of the parent segment.

FORMAT= F I V I FI I VI (REQUIRED for internal segments; OPTIONAL for database segments)
indicates the format of the segment.

A segment may be described in its database DBD as a variable-length segment and yet be a fixed-length segment in storage when retrieved through certain PCBs. This happens only when one or more SENFLD

statements for the segment are present in the control commands that define the PCB. If this applies to your SEGMENT definition, code FORMAT=F or let it default to F.

The values you can specify are:

F database segment of fixed length

FI fixed-length internal segment

V database segment of variable-length

VI variable-length internal segment

BYTES=*n* (REQUIRED for database segments, OPTIONAL for fixed-length internal segments, not used for variable-length internal segments)
specifies the length in bytes of the segment.

Database segments
For a fixed-length segment, replace *n* with the length of the segment, in bytes.

For a variable-length segment, replace *n* with the maximum length of the segment, in bytes. (Include the data portion as well as the 2-byte length field for the segment.)

For segments associated with a data exit routine, BYTES indicates the length of a segment *before* processing by the exit.

The value of *n* can come from two different sources:

- The BYTES keyword of the related DBDGEN SEGMENT command. This is the source if no SENFLD statements were used to define the PCB to IMS/VS.
- The SENFLD statements for the segment if one or more are used in the segment's definition. All such segments are of fixed-length to DataRefresher, whatever the database format. Their lengths should be just large enough to cover all the sensitive fields.

If a segment is the concatenation of multiple IMS/VS DL/I segments (in a logical database), indicate a value equal to the length of the output area that IMS/VS DL/I needs to contain the concatenated IMS/VS DL/I segments.

Fixed-length internal segment
Replace *n* with the number of bytes that a fixed-length internal segment occupies.

For fixed-length internal segments specify either BYTES or NEXT, not both.

START=*n* / *fieldname* + *n* / *segname* + *n* (REQUIRED for internal segments; not allowed for database segments)
specifies the starting position of an internal segment in a parent segment.

For fixed starting position, replace *n* with the starting position of the internal segment in the parent segment. The value *n* must be positive.

For variable starting position write one of the following:

fieldname + n
Replace *fieldname* with the name of a field that exists in the parent segment of the segment being described.

Replace *n* with the number of bytes from the end of the named field to the starting position of this internal segment. The value of *n* must be positive.

If the named field and this internal segment are adjacent, write *fieldname*+1.

segname + n
Replace *segname* with the name of a segment that has the same parent segment as the internal segment being described.

Replace *n* with the number of bytes from the end of the named segment to the starting position of this internal segment. The value of *n* must be positive.

If the named segment and this internal segment are adjacent, write *segname*+1.

NEXT=*fieldname + n | segname + n* **(REQUIRED for variable-length internal segments; OPTIONAL for fixed-length internal segments; not allowed for database segments)**
indicates how to step from one occurrence of the internal segment to the next. Use NEXT for fixed-length internal segments if you do not specify BYTES.

For fixed-length internal segments specify either BYTES or NEXT, not both. Write NEXT using either:

fieldname + n
Replace *fieldname* with the name of a field within the segment where the NEXT appears.

Replace *n* with the number of bytes from the end of the named field to the next occurrence of the internal segment. The value of *n* must be positive.

If the named field and the next occurrence of the internal segment are adjacent, write *fieldname*+1.

segname + n
Replace *segname* with the name of a segment internal to the segment where NEXT appears.

Replace *n* with the number of bytes from the end of the named segment to the next occurrence of the internal segment. The value of *n* must be positive.

If the named segment and the next occurrence of the internal segment are adjacent, write *segname*+1.

OCCURS=*n | fieldname* **(REQUIRED for internal segments; not allowed for database segments)**
indicates how many occurrences of an internal segment there are in a parent segment. OCCURS is not used for database segments. OCCURS

can be an integer or the name of a field in the parent whose value is the number of occurrences.

n for a fixed number of occurrences, replace *n* with that number.

fieldname
for a variable number of occurrences, replace *fieldname* with the name of the field that contains the value for the number of occurrences.

DATAEXIT=*exitname* | EXIT=*exitname* (OPTIONAL)
is used if the segment within the PCB must be interpreted or changed by a data exit before DataRefresher can extract from it. Replace *exitname* with the name of the data exit that you wish to invoke. This is the name of the load module containing the data exit that you wish to use.

If you specify DATAEXIT or EXIT you must also specify XBYTES. Do not use DATAEXIT or EXIT for internal segments.

XBYTES=*n* (OPTIONAL)
is used if the segment within the PCB must be interpreted or changed by a data exit before DataRefresher can extract it. Replace *n* with the length in bytes of the segment *after* it is processed by the data exit specified with EXIT. *n* can be any integer from 1 through 32760. For variable-length constructs, specify the maximum length in bytes of the segment after it is processed by the data exit.

If you specify XBYTES, you must also specify DATAEXIT | EXIT.

Do not specify XBYTES for internal segments.

EODCALL=N | Y (OPTIONAL and only used when DATAEXIT | EXIT is specified)
requests that DataRefresher make an end-of-data call to the data exit.

End-of-data calls, if requested, are made in addition to the normal calls for segment/record occurrences, and are useful if the exit is doing summarization.

For root level segments from an IMS database, end-of-data means at the end of the database, subject to any search strategy.

For lower level segments from an IMS database, end-of-data is relative to each occurrence of the parent. For each occurrence of the immediate parent segment an EOD call is made if requested.

For more information about end-of-data calls, see the *DataRefresher Exit Routines*.

FREQ=*n* (OPTIONAL)
optimizes access strategies by providing DataRefresher with an estimate of the average number of occurrences of a segment within its parent, or within the database if PARENT=0. The value of *n* is any number (without commas) from .01 through 16777215. Examples of permitted values are:

```
56
25.00
3.6
99.12
```

All FREQ values output by the DAP contain two decimal positions.

Defaults for FREQ

- If PARENT=0, the default value for FREQ is 1000.
- If PARENT does not equal 0 and it is a database segment, the default value for FREQ is 10.
- If PARENT does not equal 0 and it is an internal segment, the default value is the OCCURS value if specified, otherwise it is 10.

MAXNBR=1 (OPTIONAL)
is included if the DL/I segment can have no more than one occurrence within an occurrence of its parent DL/I segment.

MAXNBR=1 is not allowed with internal segments.

DESC='*comment*' (OPTIONAL)
saves a comment about the segment you are describing. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

FIELD (REQUIRED)
describes every field in a segment.

The FIELD keywords for a segment can appear in any order. Include at least one FIELD for each PCB you describe. You need not include a FIELD for every described segment and internal segment.

The fields are either:

- *ordinary* fields

  Ordinary fields are either a field that is not a key field or an index field, or they can be a key field or index field that you do not want to identify as such to DataRefresher. Ordinary fields do not use the SEQFLD keyword.

- *sequence* fields

  Sequence fields are either a key field or an index field. Sequence fields will always use the SEQFLD keyword.

FIELD can describe fixed and variable-length fields. Variable-length fields can be of either variable-length character (VC) or variable-length graphic (VG) data types. You tell DataRefresher the length of the variable-length field by specifying a length field with LFIELD. (See LFIELD on page 57 for more information on length fields.) Do not define a variable-length field as a sequence field.

If you use an exit routine to preprocess records in your source data, use FIELD to describe the segments' fields as they will appear *after* processing by the exit. One exception: if the field is the sequence field of the PSB, the FIELD keyword for the sequence field must describe the field as it looks in storage so DataRefresher can use it to implement its search strategy. If your exit attempts to change the length, data type, or value of the sequence field, the DEM suspends processing of the current batch of extract requests and queues them again for later processing.

NAME=*fieldname* (REQUIRED)
specifies the name of the field being described. For ordinary fields, replace *fieldname* with the name of the field being described.

For sequence fields, replace *fieldname* with the DBD name that appears with the keyword that defined the field to IMS. This is FIELD for a key field and XDFLD for an index field.

Field names must be DataRefresher, DataRefresher quoted, or DBCS names, as described in "DataRefresher naming conventions" on page 18.

If the field name is a DBCS name, the name must contain *only* DBCS characters. No single-byte characters are permitted. The name must be bracketed by a shift-out (X'0E') and shift-in (X'0F') character, as in this example:

```
'<F_L_D_>'
```

(X'0E' is represented by < and X'0F' by >.)

Sequence field names must be 8 characters or less. Ordinary field names can be up to 32 characters including the shift-out and shift-in characters and excluding double quotes.

TYPE=C | *value* **(OPTIONAL)**
indicates the type of data in the field.

Sequence fields are of type character (**C**), graphic (**G**), fullword (**F**), halfword (**H**), packed decimal (**P**), or zoned decimal (**Z**).

The values you specify are:

A *date* data

B single byte binary data

C EBCDIC character string

D long (8-byte) floating point number

E short (4-byte) floating point number

F 32-bit, signed binary integer

G solely DBCS data

H 16-bit, signed binary integer

P packed decimal number

S *timestamp* data

T *time* data

VC variable-length character data

VG variable-length graphic data

xx user-defined data type (*xx* gets replaced with the name of the use data type)

Z zoned decimal number

Fields of type D are normally designated as character or hexadecimal fields in the database DBD because IMS/VS has no special designation for floating point fields.

When a field is defined during DBD generation, an IMS/VS DL/I data type must be specified on its FIELD control statement. This can be C (alphameric), F (32 bit signed binary integer), H (16-bit signed binary integer), P (packed decimal), or X (hexadecimal data),

When specifying TYPE for a DBD field, do not assume that the IMS/VS DL/I field type determines the DataRefresher field type. Ordinarily, this is true for fields of types P, F, or H, but it is not always true for fields of types C or X. The C or X designation traditionally has been used for fields whose data types are foreign to IMS/VS DL/I, such as the floating point field in the example. Consult someone familiar with the database for information about how the fields actually represent the data.

START=*n* / *fieldname* + *n* / *segname* + *n*
(REQUIRED for ordinary fields, sequence index fields with the same source and target segments, and sequence key fields.)
specifies the starting position of a field in a segment or internal segment.

If a sequence field is an index field with different source and target segments, do not use START.

If an index field is described and it does not represent a true field, do not use START.

Define the starting position of an internal segment as a fixed position within the parent segment or as a variable position relative to the ending position of either another field or an internal segment.

For a fixed starting position replace *n* with the value that indicates where the field begins in its segment or internal segment after retrieval on a IMS/VS DL/I call. The value of *n* can come from the START keyword of either the:

- DBDGEN FIELD command that defines the field.
- The PSBGEN SENFLD command.

  SENFLD may not have been used in the definition of the containing PCB, but if one was used, take the value of the START keyword from the SENFLD.

For a variable starting position write either:

fieldname + *n*
Replace *fieldname* with the name of a field that exists in the same segment as the one being described. Therefore, the field must exist in the same segment as the one being described.

Replace *n* with the number of bytes from the end of the named field to the start of this field. The value of *n* must be positive.

If the named field and this field are adjacent, write *fieldname* + 1.

segname + *n*
Replace *segname* with the name of the internal segment which physically precedes the field on which START appears.

Replace *n* with the number of bytes from the end of the named segment to the start of this field. The value of *n* must be positive.

If the named segment and this field are adjacent, write *segname*+1.

If you are describing a segment with a variable-length format, be sure to account for the 2-byte length field at the start of the segment. For example, a value of START=1 means it begins at the start of the length field, not at the byte following the length field.

BYTES=*n* (REQUIRED for sequence fields and ordinary fields with lengths not determined by data type, namely ordinary fields of type A, C, P, S, T, VC, VG, and Z)

BYTES specifies how many bytes a field occupies or, for a variable-length field (VC or VG data types), the maximum number of bytes a field can be.

For ordinary fields, replace *n* with the field length in bytes.

For sequence fields, replace *n* with the field length in bytes. For index fields, this is the sum of the lengths of the component fields in the source segment.

If you are writing a DXTPSB description that describes a user-defined data type, the value you code here depends on the value you coded on the SRCBYTES keyword of the CREATE DATATYPE command. If SRCBYTES has a fixed number on the CREATE DATATYPE command, you do not need to include the BYTES keyword on the CREATE DXTPSB command. If, however, VARIES was specified for SRCBYTES on the CREATE DATATYPE command, then you must specify the length of this field with the BYTES keyword here.

If you specify BYTES for fields whose lengths are determined by data type, you must specify the correct length as shown here:

Data Type	Exact Length
B	1
D	8
E	4
F	4
H	2
xx	value of SRCBYTES specified on CREATE DATATYPE command

If you specify BYTES for fields whose lengths are not determined by data type, you must specify a length in the ranges shown here:

Data Type	Minimum Length	Maximum Length
S	16	26
A	1	10
C	1	254
G	2	254[1]
P	1	16
T	1	8
VC	1	32767
VG	2	32766[2]
Z	1	16

[1] 127 DBCS characters, BYTES must be even

[2] 16383 DBCS characters, BYTES must be even

Note: VG and G data types represent double-byte character set (DBCS) data; therefore, you must use two bytes to represent one DBCS character when you calculate the length or starting position. See Appendix F, "Output data records and fields" on page 259 for more information.

LFIELD=*fieldname* (REQUIRED for variable-length fields in variable-length internal segments, optional otherwise)
LFIELD is used only for ordinary fields to name the field that contains the length of a variable-length field (field type of VC or VG). The field named with LFIELD must physically precede the variable-length field and must exist in the same segment as the variable-length field.

The value of the field named with LFIELD must be even if the field is defined as a VG data type. The length field must be numeric data, and must have a zero scale if it is packed or zoned decimal; it cannot be floating point. A length field can have a zero value, which means that the variable-length field is not present.

If you do not specify an LFIELD for a variable-length field in a fixed-length segment, DataRefresher assumes that the length of the field is whatever is left in the record (its length is equal to the record length plus one, minus the starting position of the field).

SCALE=*n* / *0* (REQUIRED for ordinary fields of P and Z data types)
indicates the position of the decimal point in the decimal representation of the field type.

If you are writing a DXTPSB description that describes a user-defined data type, the value you code here depends on the value you coded on the SRCSCALE keyword of the CREATE DATATYPE command.

- If SRCSCALE has a fixed number on the CREATE DATATYPE command, you do not need to include the SCALE keyword on the CREATE DXTPSB command. If you do include the SCALE, it must be equal to the SRCSCALE keyword.
- If, however, VARIES was specified for SRCSCALE on the CREATE DATATYPE command, then you must specify the scale of this field with the SCALE keyword here.

The values you can specify are:

n if you want the decimal point to be placed to the left of the last *n* digits in the decimal representation. Replace *n* with a positive integer.

The value of *n* can range from 1 through the number of decimal digits that the field being described contains. For an *x*-byte field of type Z, the maximum scale, *n*, is *x*. For an *x*-byte field of type P, the maximum scale, *n*, is *2x-1*.

0 if there is no decimal point in the number (the value of the field is assumed to be an integer).

CONV=*exitname* (OPTIONAL for ordinary fields, not used for sequence fields)
specifies the name of a date/time conversion exit that converts date or time data into standard ISO format. CONV cannot be specified for any data type other than date(A) or time(T). See *DataRefresher Exit Routines* for more information on writing date/time conversion exits.

SEQFLD=R | V (REQUIRED for sequence fields, not used for ordinary fields)
specifies whether or not sequence fields have the same source and target segments.

The values you can specify are:

R if the field is a key field or index field whose source and target segments are the same.

V if the field is an index field whose source and target segments are different, or if the index field does not represent a true field.

SEQUENCE=ASC (OPTIONAL for sequence fields, not used for ordinary fields)
indicates that sequencing is by ascending value of the field. (This is always the case with IMS/VS key or index fields.)

UNIQUE=Y | N (OPTIONAL for sequence fields, not used for ordinary fields)
indicates whether the field is unique.

A key field in the root segment of the structure viewed by the PCB is unique if its values throughout the database are distinct.

A key field in a dependent segment is unique if all its values for any specific parent are distinct.

An index field is unique if its values for all of its source segments are distinct.

An index field that does not appear to be unique may in fact be unique because of "sparse indexing." Sparse indexing is indicated by the presence of NULLVAL in the defining XDFLD command.

The values you can specify are:

Y if the field has the "unique" attribute.

N if the field does not have the unique attribute.

DESC='*comment*' (OPTIONAL)
saves a comment about the field you are describing. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

Note: DataRefresher has a maximum number of fields and segments that can be defined in each DXTPCB. The maximum number is 1530. For example, if you have a PCB with 30 segments, it can have a maximum number of 1499 fields. The PCB itself counts as one field.

Examples of CREATE DXTPSB

Example 1

This example defines an IMS DL/I PSB with no variable-length fields or internal segments. Assume you want to:

- Name the DXTPCB RESTRDN
- Use information in the SENFLD statements that are present in the DBD generation statements.

```
CREATE
DXTPSB  NAME=DXTSELDN,
        DESC='CREATE A DXTPSB DESCRIPTION OF DXTSEL'
  DXTPCB  NAME=RESTRDN,  DBACCESS=HIDAM,
          DESC='DXTPCB NAMED RESTRDN'
  SEGMENT NAME=DEPT,  PARENT=0, BYTES=42,  FREQ=10,
          DESC='SEGMENT NAMED DEPT'
    FIELD   NAME=DNUM,  START=1,      BYTES=2, SEQFLD=R,
            DESC='DEPARTMENT NUMBER'
    FIELD   NAME=DAREA, START=3,      BYTES=14,
            DESC='DEPARTMENT''S SALES TERRITORY'
    FIELD   NAME=DMGR,  START=17,     BYTES=3,
            DESC='EMPLOYEE NUMBER OF DEPARTMENT''S MANAGER'
    FIELD   NAME=DDIV,  START=20,     BYTES=10,
            DESC='DEPARTMENT''S DIVISION'
    FIELD   NAME=DCITY, START=30,     BYTES=13,
            DESC='NAME OF DEPARTMENT''S CITY'
  SEGMENT NAME=STAFF, PARENT=DEPT,  BYTES=20, FREQ=10,
            DESC='SEGMENT NAMED STAFF'
    FIELD   NAME=ENUM,  START=13,     BYTES=3, SEQFLD=R,
            DESC='EMPLOYEE SERIAL NUMBER'
    FIELD   NAME=ENAME, START=1,      BYTES=9,
            DESC='NAME OF EMPLOYEE'
    FIELD   NAME=JOB,   START=16,     BYTES=5,
            DESC='EMPLOYEE''S JOB CLASSIFICATION'
    FIELD   NAME=YEARS, START=11,     TYPE=H,
            DESC='NUMBER OF YEARS EMPLOYEE HAS WORKED'
  SEGMENT NAME=HIST,  PARENT=STAFF, BYTES=9, FREQ=10,
          DESC='SEGMENT NAMED HIST'
    FIELD   NAME=DATE,  START=3,      TYPE=H,
            DESC='YEAR EMPLOYEE ASSUMED POSITION'
    FIELD   NAME=JOB,   START=5,      BYTES=5,
            DESC='EMPLOYEE''S POSITION'
    FIELD   NAME=YEARS, START=1,      TYPE=H,
            DESC='NO. OF YEARS EMPLOYEE HAS ASSUMED POSITION';
```

Notes:

1. The use of defaults greatly reduced the amount of coding necessary.
2. The fact that SENFLD statements are present in the DBD generation statements has affected:
   - The fields that can be described for the STAFF and HIST segments. In field-sensitive segments such as these, only those fields mentioned with SENFLD statements can be accessed by DataRefresher.
   - The BYTES keywords for the STAFF and HIST SEGMENT definitions. The in-storage images of these segments are shorter than the segments as they appear in the database. The in-storage images of segments of either type are just long enough to cover all the fields described by the SENFLD statements for the segments.
   - The START keywords for the STAFF and HIST segments' FIELD definitions. The values of these keywords are those contained in the corresponding SENFLD statements, rather than those in the associated FIELDS of the DBD

generation. That is, the SENFLD statements have shifted the locations of their fields in the in-storage segments.

Example 2

This example defines an IMS DL/I PSB with a variable-length field and an internal segment. Assume you want to:

- Name your DXTPSB SAMPPSB
- Name the DXTPCBs SAMPPCB1 and SAMPPCB2
- Name three segments in the first DXTPCB (A, MONTHLY, B) and one segment in the second DXTPCB (A)

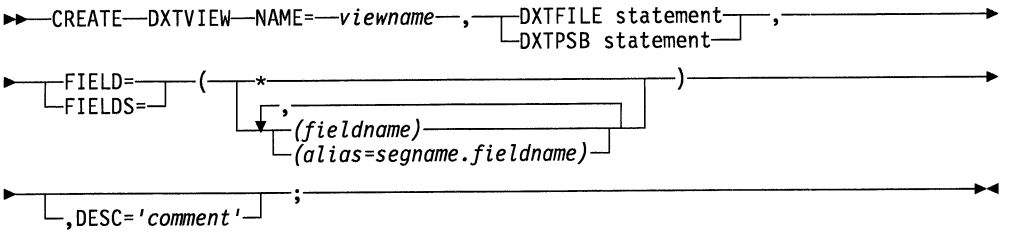
```
CREATE
 DXTPSB NAME=SAMPPSB
  /* FIRST PCB                */
  DXTPCB    NAME=SAMPPCB1,  DBACCESS=HIDAM
   SEGMENT  NAME=A,  PARENT=0,  FREQ=1150,  FORMAT=V,  BYTES=230
     FIELD  NAME=AKEY,   BYTES=2,  START=3,  SEQFLD=R
     FIELD  NAME=ANAME,  BYTES=6,  START=5
     FIELD  NAME=AVLN,   TYPE=H,   START=145
     FIELD  NAME=AV,     TYPE=VC,  START=147,  LFIELD=AVLN,
              BYTES=83
  /* INTERNAL SEGMENT DEFINITION, MONTHLY INFORMATION  */
   SEGMENT  NAME=MONTHLY,  FORMAT=FI,  BYTES=10,  OCCURS=12,
              START=20,
            PARENT=A
     FIELD  NAME=MONTHID,   BYTES=2,  START=1
     FIELD  NAME=MONTHVL1,  TYPE=F,   START=3
     FIELD  NAME=MONTHVL2,  TYPE=P,   START=7, BYTES=4,  SCALE=1
   SEGMENT  NAME=B, FREQ=2,  PARENT=A,  BYTES=55
     FIELD  NAME=BKEY,   BYTES=3,   START=1,  SEQFLD=R
     FIELD  NAME=BDATA,  BYTES=50,  START=4

  /* SECOND PCB               */
  DXTPCB    NAME=SAMPPCB2,  DBACCESS=HISAM
   SEGMENT  NAME=A,  PARENT=0, BYTES=10
     FIELD  NAME=AKEY,  BYTES=2,  START=1,  SEQFLD=R
   ;
```

Notes:

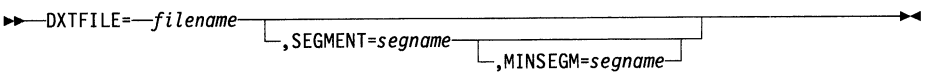
1. The variable-length field AV follows its two-byte binary length field AVLN.
2. The internal segment MONTHLY always has 12 occurrences. Its start position is fixed within the parent segment A.
3. The fields AVLN and AV in the A segment follow the internal segment MONTHLY in the record. The internal segment MONTHLY is contained in SAMPPCB1. MONTHLY starts at position 20, repeats 12 times, and is followed by field AVLN (in SAMPPCB1) which starts at position 145.

CREATE DXTVIEW

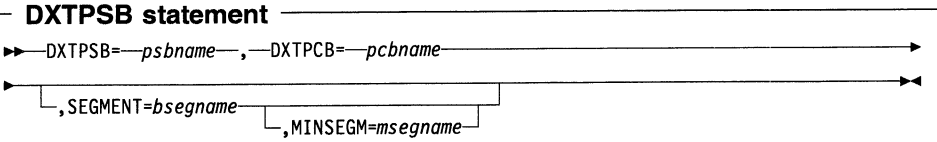
Use the CREATE DXTVIEW (UIM) command to create a DXTVIEW, a type of data description, of a file or IMS/VS DL/I database. Data descriptions are required only for DEM extract requests.



DXTFILE statement



DXTPSB statement



Generic Data Interface (GDI) Select Exit: If you are using a GDI select exit, you cannot create a DXTVIEW; DataRefresher automatically generates a *dummy* view (no fields are defined) for a GDI select exit. The dummy view DataRefresher generates has the same name as the DXTFILE description. If the DXTFILE for the GDI select file is deleted, DataRefresher automatically deletes the associated DXTVIEW.

You can get the effect of multiple views of a GDI select file by defining multiple CREATE DXTFILE commands for the same GDI select exit; DataRefresher automatically generates an associated DXTVIEW based on each CREATE DXTFILE command.

CREATE DXTVIEW (REQUIRED)
specifies that a DXTVIEW is being created.

NAME=*viewname* (REQUIRED)
assigns a name to the DXTVIEW. Replace *viewname* with the name you assign. The name can be a DataRefresher name, a DataRefresher quoted name, or a DBCS name, as described in "DataRefresher naming conventions" on page 18; it must be unique among all the names for DXTVIEWs in your FDTLIB. DataRefresher supports up to 32 characters for DXTVIEW names.

DXTFILE=*filename* (REQUIRED for a DXTVIEW of a file)
creates a DXTVIEW of a file. Replace *filename* with the name of the DXTFILE description upon which your new DXTVIEW is based.

SEGMENT=*segname* (REQUIRED for structured files)
is used when *filename* is a structured file. (A structured file is a file with multiple record types or internal segments.)

- For structured files with multiple record types but no internal segments, replace *segname* with the name of the segment whose fields you want to use in your DXTVIEW.
- For a structured file with internal segments, replace *segname* with the name of the bottom segment of the path. (Internal segments in a file impose a hierarchy like that of an IMS/VS DL/I database; all segments in the path from the top level segment to the bottom segment specified by *segname* are automatically included in the DXTVIEW.)

MINSEGM=*segname* (OPTIONAL)
identifies the minimum or lowest segment that must be present in your DXTVIEW path. MINSEGM is used only for structured files with internal segments. It can be specified for any segment in the path over which the DXTVIEW is defined, including internal segments.

You can only specify MINSEGM if you have already specified SEGMENT. Replace the *segname* value of the MINSEGM keyword with the name of the segment representing the minimum requirement for data extraction qualification.

When an extract request accesses data at the MINSEGM level or deeper, the hierarchical path at least down to the MINSEGM segment must be present in order to qualify as a source DXTVIEW occurrence. When an extract request does not need data as deep into the hierarchy as the MINSEGM level, (for example, a root-only extract from a hierarchy where MINSEGM is a subordinate segment), then a qualifying occurrence must be present at each level accessed to generate an output row.

After the existence of the MINSEGM segment is established, then data that exists beneath that segment occurrence can be extracted. However, it is not necessary to have a complete path to the bottom to extract data immediately subordinate to the MINSEGM segment.

If MINSEGM is not specified, the default is the segment specified by SEGMENT.

End-of-data notes for segments: For any DXTVIEW that is defined over a hierarchical path containing a segment for which a data exit with an end-of-data call is defined, that segment must be the lowest segment of the path (named specifically with the SEGMENT keyword).

If the lowest segment of a DXTVIEW has a data exit with an end-of-data call, the utility of the MINSEGM keyword is limited. MINSEGM can still be specified for any segment of the hierarchical path, but the functioning will be exactly the same for the two following MINSEGM specifications:

- MINSEGM= segment with a data exit that has an end-of-data call
- MINSEGM= that segment's parent segment

When you use an exit with an end-of-data call, you are effectively taking over the end-of-data processing, and creating a logical unit of the two segments. DataRefresher will treat the two segments as a unit for

purposes of MINSEGM processing and extraction from incomplete hierarchical paths. For more information about end-of-data calls, see *DataRefresher Exit Routines*.

DXTPSB=*psbname* (REQUIRED for a DXTVIEW of a database PCB)
creates a DXTVIEW of a data base PCB. Replace *psbname* with the name of the DXTPSB description that contains the DXTPCB description that your DXTVIEW is based on.

DXTPCB=*pcbname* (REQUIRED with DXTPSB)
specifies the name of the DXTPCB. Replace *pcbname* with the name of the DXTPCB description that your DXTVIEW is based on.

SEGMENT=*bsegname* (REQUIRED with DXTPSB)
identifies the bottom segment for your DXTVIEW. Replace *bsegname* with the name of the bottom segment, which must be the name of a database segment or internal segment in the DXTPSB description. It need not, however, be the bottom segment of the hierarchy. If the *bsegname* is a segment from the IMS/VS hierarchy, then no internal segments are available through the DXTVIEW, even though they may physically reside in *bsegname* or further up the hierarchical path above *bsegname*.

MINSEGM=*msegname* (OPTIONAL)
identifies the minimum required segment that must be present in your DXTVIEW path. It can be specified for any segment from the path over which the DXTVIEW is defined, including any internal segment.

Replace *msegname* with the name of the segment representing the minimum requirement for data extraction qualification.

For more information on the MINSEGM keyword, see the explanation of MINSEGM under the DXTFILE keyword on page 62.

If MINSEGM is not specified, the default is the segment specified with SEGMENT.

FIELD= | FIELDS= (REQUIRED)
specifies the fields you want in the DXTVIEW.

This depends on what your DXTVIEW is based on:

- For a DXTFILE description of a simple file, you can specify any fields in the file.
- For a database, you can name only fields that are described in the DXTPCB description and that lie in the path of the DXTVIEW determined by the bottom segment designated by SEGMENT.
- For a DXTFILE description of a structured file with multiple record types and no internal segments, your DXTVIEW is limited to one segment. Therefore, you can name only fields in that segment in your DXTVIEW.
- For a DXTFILE description of a structured file with internal segments, SEGMENT defines a path. You can name only fields that lie in the path of the DXTVIEW determined by the bottom segment you designated by SEGMENT.

The values you can specify for the FIELD | FIELDS are * (asterisk), *fieldname*, *alias=segname.fieldname*, or a combination of both *fieldname*

and *alias=segname.fieldname*. If you specify more than one *fieldname* or *alias=segname.fieldname*, then you must enclose the list in parentheses and separate the fields with commas.

*** (asterisk)**
makes visible every field in your DXTFILE description, every field in your designated DXTFILE description segment, or every field in the path of your DXTPCB description.

When specifying fields for a DXTVIEW of a database, you cannot use FIELD=* when the path for the view contains two or more eligible fields that have the same names. Use aliases to distinguish fields with the same name.

You can use FIELD=* when fields in the DXTPCB description have duplicate names, as long as the fields in the path for the DXTVIEW being defined do not contain these duplicate names.

(*fieldname*)
is used to identify the fields that you want to view. Replace *fieldname* with a list of field names, separated by commas if no other fields with the same name exist. For a database description and for a structured DXTFILE description that may have duplicate field names, use alias=*segname.fieldname*.

Field names must be DataRefresher, DataRefresher quoted, or DBCS names, as described in "DataRefresher naming conventions" on page 18.

alias=*segname.fieldname*
assigns an *alias* to any field in your list. An alias is an alternate name for a field and must be a DataRefresher, DataRefresher quoted, or DBCS name, as described in "DataRefresher naming conventions" on page 18. That alias must then be used instead of the field name when the field is selected through your DXTVIEW.

DataRefresher never requires you to use an alias for a field in a non-IMS file. However, a DXTPCB description or DXTFILE description of a structured file with internal segments can describe two or more fields with the same name, as long as these fields are in different segments of the database. If this is the case and you want these fields in your DXTVIEW, you must assign aliases to all but one of them. For example, if there are three fields with the same name in your DXTVIEW, you must assign aliases to two of them so that they can be distinguished from one another.

Note: The underlying DXTFILE description or DXTPCB description for a DXTVIEW can be updated. If it is, the DXTVIEW then refers to the fields in the updated file or DXTPCB description. Be sure when an update occurs that the dependent DXTVIEWs are still correct, or modify the DXTVIEWs so that they are compatible with the updated data description.

DESC='*comment***' (OPTIONAL)**
saves a comment about the data description you are creating. For more information about coding this keyword, see Appendix D, "Coding the Description" on page 255.

Examples of CREATE DXTVIEW

Example 1

This example creates a DXTVIEW of a simple VSAM data set. The DXTFILE description upon which your DXTVIEW is based is named PRODUCTS.

```
CREATE
DXTVIEW NAME=PRODUCTS,
        DESC='DXTVIEW NAMED PRODUCTS',
        DXTFILE=PRODUCTS,
        FIELD=*;
```

Notes:

1. The DXTVIEW and DXTFILE description can have the same name because they are different types of data descriptions.
2. This is a DXTVIEW of a simple file (one record type) so every field in the file is eligible for extraction; in this case the * for the FIELDS keyword specifies that all fields are to be extracted.
3. There are no segments because it is a simple file.

Example 2

This example creates two DXTVIEWs of a structured file, a VSAM data set with multiple record types and no internal segments. The DXTFILE description upon which your view is based is named INVNTRY. Assume you want to:

- Name the first DXTVIEW PRODVW and make all fields in the PRODSEG segment visible in your DXTVIEW

```
CREATE
DXTVIEW  NAME=PRODVW,
         DESC='DXTVIEW NAMED PRODVW',
         DXTFILE=INVNTRY,
         SEGMENT=PRODSEG,
         FIELD=*;
```

- Name the second DXTVIEW WHSEVW and make only the fields IKEY, INUM, LOCN, and QONHAND in the WHSESEG visible
- Designate QTY as an alias for the field QONHAND in the second DXTVIEW.

```
CREATE
DXTVIEW  NAME=WHSEVW,
         DESC='DXTVIEW NAMED WHSEVW',
         DXTFILE=INVNTRY,
         SEGMENT=WHSESEG,
         FIELD=(IKEY,INUM,LOCN,QTY=WHSESEG.QONHAND);
```

Example 3

This example creates two DXTVIEWs of a DXTPSB with internal segments. The DXTPSB description upon which your DataRefresher view is based is named SAMPPSB. You create two views over PCB SAMPPCB1.

```
CREATE
DXTVIEW  NAME=VMONTH,
         DESC='VIEW OVER A AND MONTHLY',
         FIELDS=*,
         DXTPSB=SAMPPSB,
         DXTPCB=SAMPPCB1,
         SEGMENT=MONTHLY;

CREATE
DXTVIEW  NAME=VIEWAB,
         DESC='VIEW OVER A AND B',
         FIELDS=*,
         DXTPSB=SAMPPSB,
         DXTPCB=SAMPPCB1,
         SEGMENT=B,
         MINSEGM=A;
```

Notes:

1. In the first DXTVIEW, VMONTH, the bottom segment (SEGMENT) is MONTHLY. Therefore, MONTHLY (and all segments from the root segment to MONTHLY) must be present to enable data extraction. MINSEG is not included because it will default to the segment named with the SEGMENT keyword.
2. For DXTVIEW VIEWAB, segment B is defined as the bottom segment in the path; segment A is specified with MINSEGM, thus defining a partial path in your DXTVIEW. Segment A is the minimum segment required to enable data extraction from your path. Segment B does not have to be present.

Example 4

This example creates a DXTVIEW of a DataRefresher file with internal segments. The DXTFILE description upon which this view is based is called DXTACCNT. Assume you want:

- To name the DXTVIEW VIEWDXTACCNT
- To specify ACCOUNT as the name of the bottom segment for this DXTVIEW.
- To include all the fields in the path, so you specify * (asterisk) on the FIELDS keyword.

MINSEGM will default to the ACCOUNT segment which you named as the bottom segment in the path.

```
CREATE
DXTVIEW NAME=VIEWDXTACCOUNT,
        DESC='CREATE A DXTVIEW OF DXTACCNT',
        DXTFILE=DXTACCNT,
        FIELDS=*,
        SEGMENT=ACCOUNT;
```

DELETE

Use the DELETE command to delete unwanted data descriptions (to which you have access) from the FDTLIB.

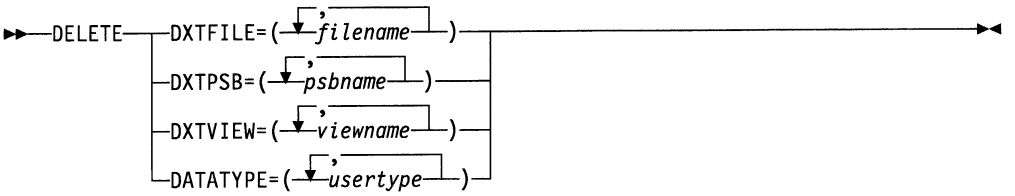
When you issue a DELETE command for a DXTFILE or DXTPSB, the UIM automatically writes a message to the DXTPRINT data set listing any DXTVIEWs that are based on the data description you are deleting. This way, your DXTFILEs and DXTPSBs can be easily cross-referenced with their corresponding DXTVIEWs.

The DELETE command is also an intermediate step in updating DataRefresher:

- File descriptions
- PSB descriptions
- View descriptions
- User data type descriptions

When updating one of these descriptions, first delete the existing version of the data description using the DELETE command and then create the updated version of the data description. For more information about updating a data description, see the PUNCH command.

If you delete the DXTFILE data description of a generic data interface (GDI) select exit, DataRefresher automatically deletes the associated DXTVIEW.



DELETE (REQUIRED)
specifies one or more data descriptions you want to delete.

DXTFILE=(*filename*) (OPTIONAL)
identifies the DXTFILE description(s) to delete.

Replace *filename* with the names of the DXTFILE descriptions you want to delete, separating the names with commas. You can delete up to 16 DataRefresher file descriptions in one DELETE command.

If you are deleting only one DXTFILE description, omit the parentheses.

DXTPSB=(*psbname*) (OPTIONAL)
identifies the DXTPSB description(s) to delete.

Replace *psbname* with the names of the DXTPSB descriptions you want to delete, separating the names with commas. You can delete up to 16 DXTPSB descriptions in one DELETE command.

If you are deleting only one DXTPSB description, you can omit the parentheses.

DXTVIEW=(*viewname*) (OPTIONAL)
identifies the DXTVIEW description(s) to delete.

Replace *viewname* with the names of the DXTVIEW descriptions you want

to delete, separating the names with commas. You can delete up to 16 DXTVIEW descriptions in one DELETE command.

If you are deleting only one DXTVIEW description, you can omit the parentheses.

DATATYPE=(*usertype*) (OPTIONAL)
specifies the user data type(s) to delete.

Replace *usertype* with the names of the user data types, separated by commas. These names are the 2 character identifiers used on the SRCTYPE keyword of the CREATE DATATYPE command.

Up to 16 user data types can be deleted with one DELETE command.

If you are deleting only one user data type, you can omit the parentheses.

Examples of DELETE

Example 1

This example deletes the DXTPSB descriptions named PSB1 and PSB2:

```
DELETE DXTPSB=(PSB1,PSB2);
```

Example 2

This example deletes only one DXTFILE description named FILE1:

```
DELETE DXTFILE=FILE1;
```

GETDEF

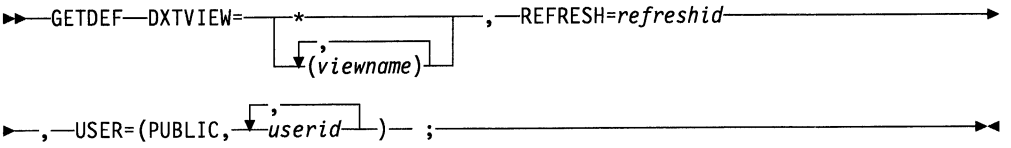
Use the GETDEF command to get DXTVIEW descriptions from local or remote FDTLIBS for the DataRefresher administrator. These descriptions are used to set up the DataRefresher End User Dialogs.

When you extract data from DEM data sources (an IMS data base, a VSAM or physical sequential data set, or data sources accessed by GDI exits) and you are using the DataRefresher End User Dialogs, the DEM requires view descriptions of the source data and validation of a user's access to the source data.

The GETDEF (UIM) command is used by the End User Dialogs administrator to obtain this information from the DXTVIEW descriptions stored in the FDTLIB. The requested DXTVIEW entry, or series of entries, are then used to produce a formatted list of field names and their data types, lengths, and associated remarks in an output data set.

A program called the master index table (MIT) update program then uses this information to update the master index table (MIT). The updated MIT can eventually use this information to authorize or restrict access to tables and views when users write extract requests.

You can also use GETDEF to produce a list of the field names in a DXTVIEW that you are authorized to use when extracting data. For GDI select files, GETDEF calls the exit for details.



GETDEF (REQUIRED)
gets DXTVIEW descriptions from local or remote FDTLIBs.

Following its successful completion, the output from the GETDEF command will be returned in the form of a data set (MVS) or CMS file (VM). To route this table/view description data to the proper destination, a special JCS data set must be defined and included in the UIM job stream for this purpose. The UIM will expect the DD name associated with this data set to be **EUEIJCS**. The data set must contain an *EO statement to identify the insertion point for table/view descriptive data.

DXTVIEW=* | *viewname* (REQUIRED)
identifies the DXTVIEW(s) to be accessed.

The values you can specify are:

*** (asterisk)**
indicates that all DXTVIEW entries that the user is authorized to use are to be selected. The * is primarily used for the initial construction of the MIT.

(*viewname*)
indicates that only view data for the named DXTVIEW(s) is requested. A maximum of 16 view names can be requested. All views you request must be enclosed in parentheses; multiple views must be separated by

commas. (DataRefresher automatically generates DXTVIEWs for GDI select exits; these views have the same name as the DXTFILE.)

If a given DXTVIEW name is unauthorized or not in the FDTLIB, an appropriate message will be generated.

REFRESH=*refreshid* (REQUIRED)
specifies a 5-digit number supplied by the Administrative Dialogs End User administration task. This number correlates data returned from the GETDEF command with the MIT to be updated. The dialogs flag the rows in the MIT so that the proper rows get updated when the descriptive data is returned.

USER=(PUBLIC,*userid*) (REQUIRED)
identifies the name of one or more users for whom access to the named DXTVIEWs is to be verified. All names must be enclosed in parentheses; multiple names must be separated by commas. PUBLIC is always specified to indicate the first user.

A DXTVIEW is not RACF registered if:

- RACF is not installed or active
- RACF is active but the DVRDEFN class has not been defined
- RACF is active and the DVRDEFN class has been defined, but this particular DXTVIEW has not been defined to RACF.

If a DXTVIEW being requested either explicitly (by name) or implicitly (by *) is not RACF registered then output for that DXTVIEW is generated for the first user only (PUBLIC). If a DXTVIEW is RACF registered, then output for that DXTVIEW is generated for all named users who are authorized by RACF to access the DXTVIEW.

Examples of GETDEF

Example 1

This example gets the definition of DXTVIEW VIEW1, and returns the definition for each named user if that user has access to the RACF-protected resource:

```
GETDEF DXTVIEW=(VIEW1),
       REFRESH=00001,
       USER=(PUBLIC,JONES,JENKINS);
```

Example 2

This example gets the definitions for three different views, VIEW1, VIEW2, and VIEW3. If the three views have not been RACF registered, then all three views will be returned. If the three views have been RACF registered, then only the views that have been defined to RACF as public will be returned.

```
GETDEF DXTVIEW=(VIEW1,VIEW2,VIEW3),
       REFRESH=00001,
       USER=(PUBLIC);
```

LIST

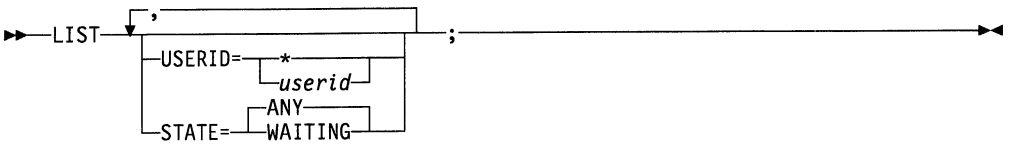
Use the LIST command to write the following output to the DXTPRINT data set:

- ID of the extract request
- User ID of the extract request originator
- Name of the PSB from which the extract request retrieves data (if extracting data from an IMS database)
- Name of the file or PCB from which the extract request retrieves data
- Date that the extract request was submitted
- Time that the extract request was submitted
- Number of output rows specified with the OUT keyword of the extract request
- Priority number as specified with the PRI keyword of the extract request
- Disposition of the extract request

This information is useful in establishing DEM initialization parameters.

The LIST command can provide information on either all of the requests in EXTLIB, regardless of their source or status, or on a defined subset of extract requests. These subsets can include only extract requests:

- Of a named user
- Currently awaiting execution.



LIST (REQUIRED)
writes information about DEM extract requests to the DXTPRINT data set.

USERID=* | *userid* (OPTIONAL)
provides the ID of the user whose requests are to be listed. This user ID must match the user ID provided when the request was submitted. The values you can specify are:

*** (asterisk)**
means that all requests that satisfy STATE specifications are listed, regardless of their USERID values.

userid
lists extract requests for a specific user. Replace *userid* with the user ID written on the USERID keyword of the SUBMIT command for the extract request(s) you want to list.

If the USERID keyword is omitted, then all requests that do not have a USERID keyword specified with the extract request, and satisfy the STATE specifications, are listed.

STATE=ANY | WAITING (OPTIONAL)
helps to identify the extract requests to be listed. The values you can specify are:

ANY
means that all extract requests in EXTLIB which satisfies the USERID specifications are listed, regardless of their current status.

WAITING
means that only extract requests currently awaiting execution and which satisfies the USERID specifications are listed.

Examples of LIST

Example 1

This example lists all requests currently waiting for execution. You can use this to schedule the DEM.

```
LIST USERID=*, STATE=WAITING ;
```

Example 2

This example lists requests identified with particular USERID. You can use this to check your requests.

```
LIST USERID=SMITH ;
```

Example 3

This example lists only those requests identified with a particular user and currently waiting for execution. You can use this to see which of your requests in the EXTLIB have been validated and are waiting to be run.

```
LIST USERID=USER1, STATE=WAITING ;
```

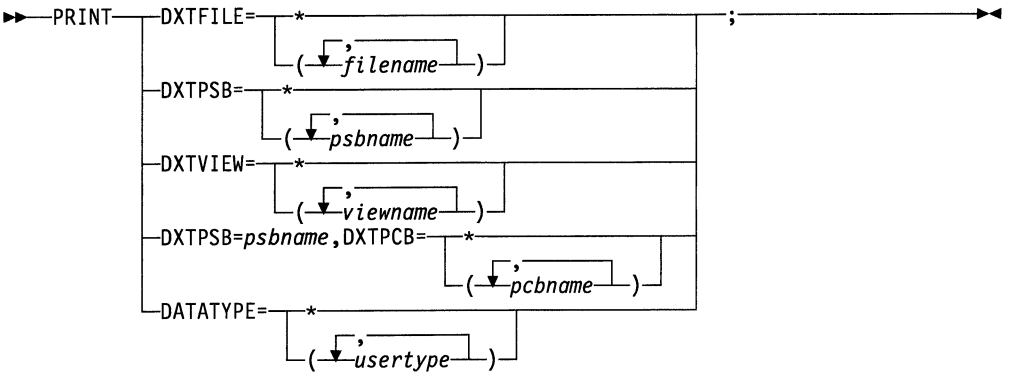
PRINT

Use the PRINT command to print a data description. You can use the printed results (which appear as a DXTPRINT data set) to examine the contents of the local or remote FDTLIB. A printed data description will not contain any comments that may have appeared with the data description when it was created.

When you issue a PRINT (UIM) command for a DXTFILE or DXTPSB, the UIM automatically issues a message indicating whether any DataRefresher views exist that are based on the data description you are printing. If the DXTFILE or DXTPSB has more than one associated DXTVIEWs that you are authorized to see, the message lists their names. This way, your DXTFILEs and DXTPSBS can be easily cross-referenced with their corresponding DXTVIEWs. The UIM writes these messages to the DXTPRINT data set at the same time it writes the PRINT command.

When the UIM executes a PRINT command for a GDI select file, it does not generate field level information.

Up to 16 data descriptions can be printed with one PRINT command (enclosed in parentheses and separated by commas).



PRINT (REQUIRED)
prints a data description.

DXTFILE=* | (*filename*) (OPTIONAL)
identifies the DXTFILE description(s) to print. The values you can specify are:

*** (asterisk)**
specifies every DXTFILE description you are authorized to use.

(*filename*)
Replace *filename* with the names of the DXTFILE descriptions you want to print, separating the names with commas.

You can specify up to 16 DXTFILE descriptions in one PRINT (UIM) command.

If you are printing only one DXTFILE description, you can omit the parentheses.

DXTPSB=* | (*psbname*) (OPTIONAL)
identifies the DXTPSB description(s) to print. The values you can specify are:

*** (asterisk)**
specifies every DXTPSB description you are authorized to use.

(*psbname*)
Replace *psbname* with the names of the DXTPSB descriptions you want to print, separating the names with commas.

You can specify up to 16 DXTPSB descriptions in one PRINT (UIM) command.

If you are printing only one DXTPSB description, you can omit the parentheses.

DXTVIEW=* | (*viewname*) (OPTIONAL)
identifies the DXTVIEW description(s) to print. The values you can specify are:

*** (asterisk)**
specifies every DXTVIEW description you are authorized to use.

(*viewname*)
Replace *viewname* with the names of the DXTVIEW descriptions you want to print, separating the names with commas.

You can specify up to 16 DXTVIEW descriptions in one PRINT (UIM) command.

If you are printing only one DXTVIEW description, you can omit the parentheses.

DXTPSB=*psbname*, DXTPCB=* | (*pcbname*) (OPTIONAL)
identifies one or more DXTPCB description(s) within a specific DXTPSB description you want to print.

Replace *psbname* with the name of the DXTPSB description that contains the DXTPCB description(s) you want to print. Replace *pcbname* with either:

*** (asterisk)**
specifies every DXTPCB description you are authorized to use (in the DXTPSB description that you have specified).

(*pcbname*)
Replace *pcbname* with the names of the DXTPCB descriptions (in the DXTPSB description you specified) you want to print, separating the names with commas.

You can specify up to 16 DXTPCB descriptions in one PRINT (UIM) command.

If you are printing only one DXTPCB description, you can omit the parentheses.

DATATYPE=* | *usertype* **(OPTIONAL)**
prints a previously defined user data type.

*** (asterisk)**
specifies all data types defined to DataRefresher.

usertype
specifies the 2 character identifier that was included on the SRCTYPE keyword of the CREATE DATATYPE command.

Examples of PRINT

Example 1

This example prints the DXTFILE descriptions named FILE1 and FILE2:

```
PRINT DXTFILE=(FILE1,FILE2);
```

Example 2

This example prints the DXTPCB descriptions PCBA, PCBB, and PCBC in the DXTPSB description named PSB1:

```
PRINT DXTPSB=PSB1,DXTPCB=(PCBA,PCBB,PCBC);
```

Example 3

This example prints every DXTVIEW description in the FDTLIB that you are authorized to use:

```
PRINT DXTVIEW=*;
```

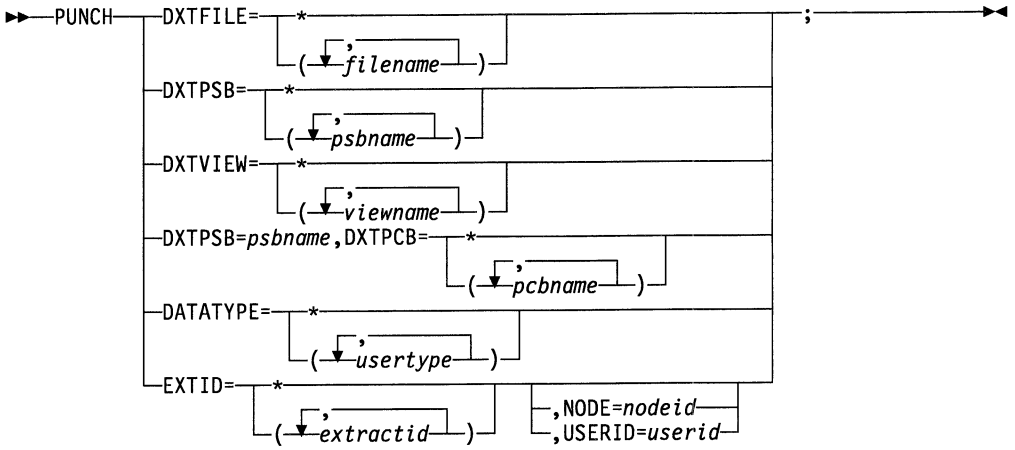
PUNCH

Use the PUNCH command to put a data description in the DXTPUNCH data set.

You can use the punched results, which appear as DXTPRINT and DXTPUNCH data sets, to examine the contents of the FDTLIB and EXTLIB and to update a data description.

A punched data description will not contain any comments that may have appeared with the data description when it was created.

When the UIM executes a PUNCH command for a GDI select file, it does not generate field level information.



PUNCH (REQUIRED)
punches a data description.

DXTFILE=* | (*filename*) (OPTIONAL)
identifies the DXTFILE description(s) to punch. The values you can specify are:

*** (asterisk)**
specifies every DXTFILE description you are authorized to use.

(*filename*)
Replace *filename* with the names of the DXTFILE descriptions you want to punch, separating the names with commas.

You can specify up to 16 DXTFILE descriptions in one PUNCH (UIM) command.

If you are punching only one DXTFILE description, you can omit the parentheses.

DXTPSB=* | (*psbname*) (OPTIONAL)
identifies the DXTPSB description(s) to punch. The values you can specify are:

*** (asterisk)**
specifies every DXTPSB description you are authorized to use.

(*psbname*)
Replace *psbname* with the names of the DXTPSB descriptions you want to punch, separating the names with commas.

You can specify up to 16 DXTPSB descriptions in one PUNCH (UIM) command.

If you are punching only one DXTPSB description, you can omit the parentheses.

DXTVIEW=* | (*viewname*) (OPTIONAL)
identifies the DXTVIEW description(s) to punch. The values you can specify are:

*** (asterisk)**
specifies every DXTVIEW description you are authorized to use.

(*viewname*)
Replace *viewname* with the names of the DXTVIEW descriptions you want to punch, separating the names with commas.

You can specify up to 16 DXTVIEW descriptions in one PUNCH (UIM) command.

If you are punching only one DXTVIEW description, you can omit the parentheses.

DXTPSB=*psbname*, DXTPCB=* | (*pcbname*) (OPTIONAL)
identifies one or more DXTPCB description(s) within a specific DXTPSB description.

Replace *psbname* with the name of the DXTPSB description that contains the DXTPCB description(s). Replace *pcbname* with either:

*** (asterisk)**
specifies every DXTPCB description you are authorized to use (in the DXTPSB description that you have specified).

(*pcbname*)
Replace *pcbname* with the names of the DXTPCB descriptions (in the DXTPSB description you specified) you want to punch, separating the names with commas.

You can specify up to 16 DXTPCB descriptions for a given DXTPSB description.

If you are punching only one DXTPCB description, you can omit the parentheses.

DATATYPE=* | *usertype* (OPTIONAL)
punches a previously defined user data type.

*** (asterisk)**
specifies all data types defined to DataRefresher.

usertype
specifies the 2 character identifier specified on the SRCTYPE keyword of the CREATE DATATYPE command.

Up to 16 user data types can be printed with one PRINT command (enclosed in parentheses and separated by commas).

EXTID=* | *extractid* **(OPTIONAL)**
identifies the ID of the extract request(s) to PUNCH. The values you can specify are:

*** (asterisk)**
specifies every extract request you are authorized to use.

extractid
specifies a particular extract request.

Replace *extractid* with the IDs of the of the extract requests you want to punch, separating the IDs with commas. You can specify up to 16 IDs in one PUNCH (UIM) command.

If you are punching only one extract request, you can omit the parentheses.

NODE= *nodeid* **(OPTIONAL)**
identifies the node ID of the extract request you are punching.

Replace *nodeid* with the value of the NODE keyword specified on the SUBMIT command for the extract request you are punching. If the NODE keyword was not included on the SUBMIT command, do not write it in your PUNCH command.

USERID= *userid* **(OPTIONAL)**
identifies the userid of the extract request you are punching.

Replace *userid* with the value of the USERID keyword specified on the SUBMIT command for the extract request you are checking. If the USERID keyword was not included on the SUBMIT command, do not write it in your PUNCH command.

Examples of PUNCH

Example 1

This example punches the DXTFILE descriptions named and FILE1 and FILE2:

```
PUNCH DXTFILE=(FILE1,FILE2);
```

Example 2

This example punches the DXTPCB descriptions PCBA, PCBB, and PCBC in the DXTPSB description named PSB1:

```
PUNCH DXTPSB=PSB1,DXTPCB=(PCBA,PCBB,PCBC);
```

Example 3

This example punches every DXTVIEW description in your FDTLIB that you are authorized to use:

```
PUNCH DXTVIEW=*;
```

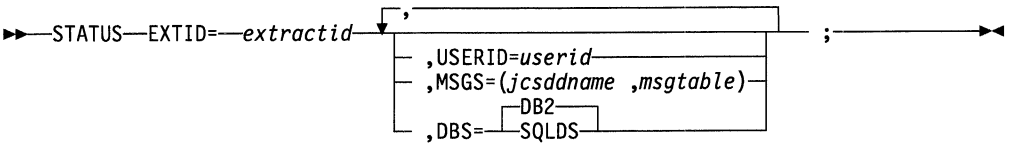
Example 4

This example punches all extract requests with a userid of SMITH:

```
PUNCH EXTID=*,USERID=SMITH;
```

STATUS

Use the STATUS (UIM) command to check the status of an extract request.



STATUS (REQUIRED)
checks the status of a DEM extract request.

EXTID=*extractid* (REQUIRED)
identifies the ID of the extract request.

Replace *extractid* with the ID of the relevant extract request. The value of *extractid* must be an SQL name.

USERID=*userid* (OPTIONAL)
identifies the user ID of the extract request whose status you are checking.

Replace *userid* with the value of the USERID keyword of the SUBMIT command for the extract request you are checking. If the USERID keyword was not included in the SUBMIT command, do not write it in your STATUS command. The value of *userid* must be an alphameric/special characters name.

MSGS=(*jcsddname,msgtable*) (OPTIONAL)
indicates that you want the messages (issued by DataRefresher about your STATUS command) to be sent to a relational message table. If you do not include the MSGS keyword, the UIM sends the messages only to the DXTPRINT data set.

Replace *jcsddname* with the name of the data set containing the JCS to load the messages. This JCS must contain both *CD and *EO statements so that DataRefresher can replace them with the generated load control deck and the messages.

Replace *msgtable* with the name of the relevant message table. The *msgtable* can be written as either *authid.tablename* or *tablename* where

authid
specifies the authorization ID. If *authid* is not included, the relevant load utility assumes it is the same as the user ID written in the JOB statement of the JCS data set that you use to load your STATUS messages.

authid is an alphameric/special characters name.

tablename
specifies the name of the table. It can be either an SQL name or an SQL quoted name.

DBS=DB2 | SQLDS (OPTIONAL)
specifies the database system where messages from the STATUS command are loaded. this determines the type of load control deck DataRefresher generates. The values you can specify are:

DB2
to load your messages into a DB2 table.

SQLDS
to load your messages into an SQL/DS table.

If the MSGS keyword is not included, DataRefresher ignores the DBS keyword.

Examples of STATUS

Example 1

This example:

- Checks the status of an extract request whose extract ID is EXTREQ1 and whose user ID is SMITH
- Sends messages issued by the UIM to the DB2 message table named DXTMSGS
- Nominates the DB2JCS data set to load the messages issued by the UIM into the message table
- Loads messages from the STATUS command into the DB2 system

```
STATUS EXTID=EXTREQ1,USERID=SMITH,
       MSGS=(DB2JCS,DXTMSGS),DBS=DB2;
```

Example 2

This example:

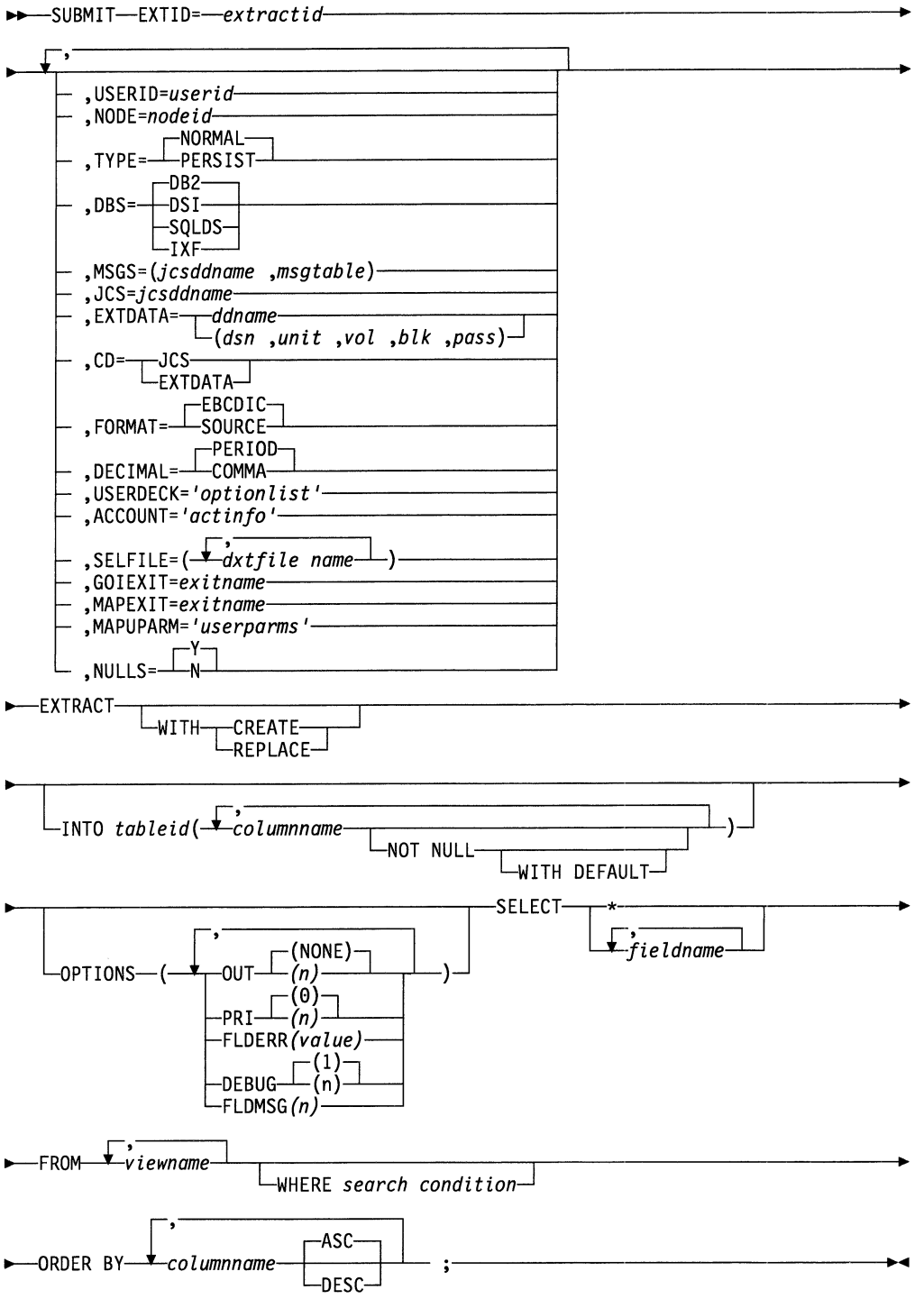
- Checks the status of an extract request whose extract ID is EXTREQ1, and which does *not* have a user ID
- Sends messages issued by the UIM about your STATUS command to the DB2 message table named DXTMSGS
- Nominates the DB2JCS data set to load the messages issued by the UIM into the DB2 message table
- Loads messages from the STATUS command into the DB2 system

```
STATUS EXTID=EXTREQ1,MSGS=(DB2JCS,DXTMSGS),
       DBS=DB2;
```

Note: Because the two examples of STATUS commands have different user IDS, they would not check the status of the same extract request.

SUBMIT/EXTRACT

Use the SUBMIT command and the EXTRACT statement to extract data from a DEM data source.



SUBMIT (REQUIRED)
submits a DEM extract to DataRefresher.

EXTID=*extractid* (REQUIRED)
identifies the extract ID of your extract request.

Replace *extractid* with any identifier you want to use for this extract request.

extractid can be an SQL, alphameric/special characters, or DBCS name.

USERID=*userid* (OPTIONAL)
identifies the user ID of your extract request.

Replace *userid* with an identifier of your choice.

While *userid* can be any alphameric/special characters name, you should replace it with the user ID assigned to you by your installation. This helps ensure that your extract request is uniquely identified.

NODE=*nodeid* (OPTIONAL)
identifies the node ID of your extract request.

Replace *nodeid* with an identifier of your choice.

While *nodeid* can be any alphameric name, you should replace it with the node ID assigned by your installation. This helps ensure that your extract request is uniquely identified.

TYPE=NORMAL | PERSIST (OPTIONAL)
indicates whether the extract request will be purged from the EXTLIB after it has been processed by the DEM.

NORMAL
directs the DEM to purge the extract request from the EXTLIB once it has been processed.

PERSIST
directs the DEM to keep the extract request in the EXTLIB after it has been processed. The DEM will process the extract request again when it is next invoked.

DBS=DB2 | DSI | SQLDS | IXF (OPTIONAL)
identifies the database system into which your extract output is loaded specifies what type of load control deck to generate.

DB2
loads your extract output into a DB2 table.

DSI
loads your extract output into a DataPropagator Relational CCD table, and prefixes the data with four additional fields required for the CCD table.

SQLDS
loads your extract output into an SQL/DS table.

IXF
writes your output in IXF.

Control information (IXF header records) is always included with the extracted data. When you specify DBS=IXF, you must also specify the CD keyword. The USERDECK keyword is not permitted if DBS=IXF.

MSGS = (*jcsddname,msgtable*) (OPTIONAL)
indicates that you want the messages issued by DataRefresher about your SUBMIT command to be sent to a relational message table.

If you do not include the MSGS keyword, the UIM sends messages resulting from your SUBMIT command only to the DXTPRINT data set.

Replace *jcsddname* with the name of the data set containing the JCS to load the messages. This JCS must contain both *CD and *EO statements so that DataRefresher can replace them with the generated load control deck and the messages.

Replace *msgtable* with the name of the relevant message table. The *msgtable* can be written as either *authid.tablename* or *tablename* where:

authid
specifies the authorization ID.

If *authid* is not included, the relevant load utility assumes it is the same as the user ID written in the JOB statement of the JCS data set that you use to load your SUBMIT messages.

authid is an alphameric/special characters name.

tablename
specifies the name of the table. It can be either an SQL name or an SQL quoted name.

Note: Do not use the MSGS keyword if DBS=IXF.

JCS=*jcsddname* (OPTIONAL)
indicates that you want DataRefresher to generate an output job. In addition to invoking a load utility, the JCS could invoke the SORT utility or any other program you may want to use.

Replace *jcsddname* with the DD name of the data set containing the JCS associated with your extract request. If you do not include the JCS keyword, you must include either the EXTDATA keyword or the GOIEXIT keyword.

EXTDATA=*ddname* | (*dsn,unit,vol,blk,pass*) (OPTIONAL)
identifies the data set to which DataRefresher writes the output. Use it if you want to direct some or all of your extract request output to a physical sequential data set. The physical sequential data set must not have a variable record format (RECFM V) except when the extracted output is in IXF format.

The minimum logical record length (LRECL) of a physical sequential data set is 20, even if the extracted data has a smaller LRECL.

If the EXTDATA keyword is not included, then you must write the JCS keyword and the JCS that you name must contain either an *EO statement or the GOIEXIT keyword.

DataRefresher lets you to specify *DSN as a substitution variable in the DD statement in your JCS for the output data set name. When the extract request is executed, the data set name specified in the EXTDATA keyword will replace the *DSN in the DD statement.

When data in IXF is written into a generated output job, the variable length records are broken up into 80-byte record lengths, with the standard DataRefresher use of the character # to indicate continuation.

The values you can specify for EXTDATA are:

ddname
identifies the name of a DD statement to which DataRefresher writes your extract output. The DD name that you write must have a corresponding DD statement in the JCL that you use to run the DEM. In addition, if you write this keyword and the JCS keyword on your extract request, the JCS that you refer to must contain the data set name that you write on this keyword. In any case, the UIM issues a message notifying you of the logical record length that your extract request needs for the data set you identified (with a DD statement) in the JCL you use to run the DEM.

Note: Do not use the names SYSOUT, DXTOUT, DXTOUT1, DXTOUT2...DXTOUT99 as DD names. They are used for internal processing by the DEM.

(*dsn,unit,vol,blk,pass***)**
indicates that you want to allocate (and catalog) a data set for your output while it is processing your extract request. You can override the DYNAMOUT keyword of the INITDEM command and override the default values for dynamically allocated data sets.

The DEM saves and catalogs all data sets it dynamically allocates. Before reusing a dynamically allocated data set, therefore, you must first delete and uncatalog it. If you try to reuse a dynamically allocated data set before uncataloging it, MVS sends a message requesting that you cancel or provide a device type. You must cancel the data set before you can proceed.

If you write this form of EXTDATA, you must include *dsn*, but *unit, vol, blk,* and *pass* are optional. Include commas for every value that you do not write, unless the comma (or commas) is not followed by another value.

dsn
assigns a name to the data set that DataRefresher dynamically allocates. You must include *dsn*.

The name that you write can be up to 8 characters in length. The first letter of the name must be alphabetic, a special character, or a hyphen; the remaining characters can be alphabetic, special character, hyphens, or numeric.

dsn can also be a series of up to five names, joined by periods, of no more than 8 characters each. If you concatenate names, the total length of the string can be no longer than 44 characters.

unit
overrides the default unit identification for your data set. If the unit that you write is a device, replace *unit* with a name of 1 to 8 characters in length; the characters can be alphameric/special characters or hyphens. If the unit is the name of a user group, replace *unit* with a name of 1 to 8 alphameric characters in length.

The value of *unit* and *vol* must be compatible. (If the value of *unit* is not compatible with the value of *vol*, MVS requests more information from the operator and waits for a response before continuing.)

The default value of *unit* is **SYSDA**.

vol overrides the default volume serial identification for your data set. You can specify only one volume serial; if you need multiple volumes, you must use the *ddname* form of EXTDATA. Replace *vol* with hyphens or with an alphameric/special characters name of 1 to 6 characters in length.

The value of *unit* and *vol* must be compatible. (If the value of *vol* is not compatible with the value of *unit*, MVS requests more information from the operator and waits for a response before continuing.)

blk overrides the default blocking factor for your extract request. The blocking factor determines the number of extract records that DEM includes in each block. Replace *blk* with an integer from 1 through 5000.

pass
assigns a password to the dynamically allocated data set. Replace *pass* with an alphameric/special characters name.

In MVS, if you protect your data set with a password, you should have an MVS password data set. (See *OS/VS2 MVS Utilities* for more information about the IEHPROGM utility which you can use to maintain your password data set.)

Notes:

1. If you want to reuse a data set and it has a password, you must know the password before you can delete and uncatalog the data set.
2. When you are dynamically allocating a data set and DBS=IXF, DataRefresher calculates the values for the DCB as follows:
   - RECFM is VB
   - LRECL is either 85, or the length of the longest IXF record type.
   - BLOCKSIZE (where DL denotes the length of the data record)

     If DL <= 4180, BLOCKSIZE is *n* * DL + 4, where *n* is the integer resulting from the division of 8360/DL (remainder ignored), and the 4 accommodates the block descriptor word (BDW).

     If DL > 4180, BLOCK SIZE is DL + 4, where the 4 accommodates the BDW.

     DL > 32756 is considered to be an error condition.
3. If you specify a GOIEXIT with the EXTDATA keyword, then the information on the EXTDATA keyword is passed to the exit in the GOI description control block.

CD=JCS | EXTDATA (OPTIONAL)

indicates that you want DataRefresher to generate a load control deck or column descriptors when it processes your extract request.

The value that you write determines where DataRefresher places the load control deck or column descriptors in the output.

The CD keyword must be included for data in IXF. If the extracted data is going to a data set or file, specify that CD=EXTDATA. If extracted data is going to an output job, specify that CD=JCS to ensure that the IXF header, table, and column records will be included with the extracted data.

The DataRefresher-generated load utility control deck contains statements that are formatted specifically for the load utility that you invoke to load your extracted data into a relational table (DB2 or SQL/DS); the statements include the load commands and descriptions of the data, target table, and target columns. (The USERDECK keyword on the SUBMIT command can be optionally used to override default values otherwise applied to a DataRefresher-generated load control deck.)

The column descriptors contain no load commands but describe the data and the positions of the fields within the data records.

To generate a control deck or column descriptors, write both the CD keyword on the SUBMIT command and the INTO keyword on the EXTRACT command for your extract request.

The values of CD are:

JCS

causes DataRefresher to generate a load control deck or column descriptors and insert them into an output job. The JCS that you named on the JCS keyword must contain a *CD statement so that DataRefresher knows where to place the generated control deck or column descriptors.

EXTDATA

causes DataRefresher to generate a load control deck or column descriptors and inserts them into the beginning of the physical sequential data set that you specified on the EXTDATA keyword. If CD=EXTDATA, you must also include the EXTDATA keyword.

If a GOIEXIT keyword is also specified with CD=EXTDATA, then the generated load control deck or column descriptors are passed to the exit, one line at a time, on a CD call.

The combination of the EXTDATA, JCS and CD keywords determines the destination of your output and the actions to be taken following the extract. The options available may be limited by the target database specified on the DBS keyword. The output options are as follows:

File The extracted data and the control deck are written to the same file. Another job may be started when the extract completes. This job may refer to the file of extracted data.

Usage: This option is valid for every target type. However, the DB2 Load utility requires that the Load Control deck and the data to be loaded are in separate files so this option cannot be used to load the extracted into DB2.

Action: SpecifyEXTDATA= *ddname* or *dsn*, as described for the EXTDATA keyword. Specify CD=EXTDATA. If you want to run a host job afterwards, then specify JCS= *ddname*, as described for the JCS keyword.

Job

The extracted data and the control deck are stored in a job which is started when the extract completes.

Usage: This option is valid for every target type and is useful for routing extracted output to a remote system for loading. However, the data is not stored on a dataset and if the follow-on job fails, the extract may have to be run again to recover.

Action: Do not specify EXTDATA. Specify CD=JCS. Specify JCS=*ddname*, as described for the JCS keyword. The JCS should contain*CD and *EO statements.

File and job

The extracted data is stored in a dataset and the control deck is written to a job that will be started when the extract completes.

Usage: This option is valid for every target type except IXF. As the data is stored on a dataset, if the follow-on job fails the extract does not have to be run again to recover.

Action: Specify EXTDATA=*ddname* or *dsn*, as described for the EXTDATA keyword. Specify CD=JCS. Specify JCS=*ddname*, as described for the JCS keyword. The JCS should contain a *CD statement.

GOI Exit

The extracted data and the control deck (optional) are passed to an exit instead of being written by DataRefresher. A job can be started once the extract is complete.

Usage: This option is valid for every target type.

Action: Specify GOIEXIT=*exitname* as described for the GOIEXIT keyword. If you specify EXTDATA=*ddname* or *dsn*, as described for the EXTDATA keyword the values will be passed to the exit. If you want to pass the control deck to the exit, then specify EXTDATA=*ddname* or *dsn*, as before, and CD=EXTDATA. If you want to run a job when the extract is complete, then specify JCS=*ddname*, as described for the JCS keyword.

Data types: The data types in the control deck vary depending on the source column data type, target database, and the value specified in the FORMAT keyword. Table 2 on page 89 shows the data types when the target database is SQL/DS:

Table 2. Data types when target database is SQL/DS

	Target database is SQL/DS and	
Source field data type is:	**FORMAT= SOURCE**	**FORMAT= EBCDIC**
C	CHAR	CHAR
VC	CHAR	CHAR
G	GRAPHIC	GRAPHIC
VG	GRAPHIC	GRAPHIC
H	FIXED	CHAR
F	FIXED	CHAR
D	FLOAT	CHAR
E	FLOAT	CHAR
P	DECIMAL	CHAR
Z	DECIMAL	CHAR
A	CHAR	CHAR
T	CHAR	CHAR
S	CHAR	CHAR
B	FIXED	CHAR

The following table shows the data types when the target database is DB2:

Table 3. Data types when target database is DB2

	Target database is DB2 and	
Source column data type is:	**FORMAT= SOURCE**	**FORMAT= EBCDIC**
C	CHAR	CHAR
VC	VARCHAR	VARCHAR
G	GRAPHIC	GRAPHIC
VG	VARGRAPHIC	VARGRAPHIC
H	SMALLINT	INTEGER EXTERNAL
F	INTEGER	INTEGER EXTERNAL
D	FLOAT (53)	FLOAT EXTERNAL (53)
E	FLOAT (21)	FLOAT EXTERNAL (21)
P	DECIMAL	DECIMAL EXTERNAL
Z	DECIMAL	DECIMAL EXTERNAL
A	DATE EXTERNAL	DATE EXTERNAL
T	TIME EXTERNAL	TIME EXTERNAL
S	TIMESTAMP EXTERNAL	TIMESTAMP EXTERNAL
B	SMALLINT	INTEGER EXTERNAL

FORMAT=EBCDIC | SOURCE (OPTIONAL)
specifies how to format the extracted data. The values you can specify are:

EBCDIC
converts your extracted data to EBCDIC format.

SOURCE
leaves the extracted data in source format.

There are two instances in which DataRefresher will reformat the source data (even when you specify FORMAT=SOURCE):

- Zoned decimal fields become packed decimal fields.
- Single precision floating point fields become double precision floating point fields.

If DataRefresher leaves your data in source format, it does not generate null indicators (hyphens) for invalid field values (except in the two instances above in which DataRefresher does convert source data).

Note: If DataRefresher encounters a field that is improperly formatted after conversion, DataRefresher substitutes a hyphen in the blank preceding the field.

Numeric data in source format is described in *IBM 370 XA Principles of Operation* or *IBM S/370 Principles of Operations*.

DECIMAL=PERIOD | COMMA (OPTIONAL)
specifies the format of the decimal point character in your extract output. Write:

- DECIMAL=COMMA if you want the decimal point to be represented by a comma
- DECIMAL=PERIOD if you want the decimal point to be represented by a period.

USERDECK='*optionlist*' (OPTIONAL)
explicitly provides DB2 or SQL/DS load utility options. These options are included in the control deck generated by the DEM for SQL/DS or DB2.

- For the DB2 LOAD statement, DataRefresher generates the CONTINUEIF keyword which indicates a value used for a continuation field in the input record. All other LOAD keywords will use the DB2 default values unless otherwise specified by the USERDECK keyword.
- For the SQL/DS DATALOAD utility, DataRefresher generates the CONTINUED(YES) keyword in the INFILE statement to indicate that the input data may span more than one comtrol file input record. All other DATALOAD utility keywords will use the SQL/DS default value unless otherwise specified by the USERDECK keyword.

'*optionlist*'
represents selected DB2 or SQL/DS parameters you want to include in the DataRefresher-generated LOAD or DATALOAD commands. The DEM does not validate the '*optionlist*' variable. Errors will be detected by the DB2 LOAD or SQL/DS DATALOAD utility. For information on what

options you can specify for the DB2 LOAD utility, see the LOAD statement in *IBM DATABASE 2 Version 2: SQL Reference*. For information on what options you can specify for the SQL/DS DATALOAD utility, see the DATALOAD utility in the *SQL/Data System Database Services Utility for VM/SP and VM/XA SP*.

The USERDECK keyword takes the form:

```
USERDECK='parm-1,...,parm-n'
```

where each of the '*parm-n*' specifications is replaced by a DB2 or SQL/DS load utility option.

RESUME(NO) is the default for the USERDECK keyword.

Note: You cannot specify the USERDECK keyword when DBS=IXF. The header record in IXF output is similar to the LOAD and DATALOAD control decks that DataRefresher generates for DB2 and SQL/DS, but the IXF header must be in a set format and cannot be overridden by a user.

ACCOUNT='*actinfo***' (OPTIONAL)**
writes account information used by the exit routine if your installation uses an accounting exit routine to keep track of DEM use. The value that you write can be anything as long as it is no more than 32 characters and is enclosed in single quotation marks. Contact your DataRefresher administrator for specific information about a local standard accounting scheme.

SELFILE=*dxtfile-name* **(OPTIONAL)**
indicates not to process the SELECT keyword for GDI select exits. The SELECT statement will be passed intact to the exit. This means that the SQL supported by a GDI select exit is not restricted to the DataRefresher subset. This is used only for GDI select exits.

Replace *dxtfile-name* with the DataRefresher file name specified with the CREATE DXTFILE command; this indicates the name of the FDTLIB file description to be used when passing information to a GDI select exit. You can specify up to 16 file names.

If the extract request specifies that data is to be joined from multiple GDI select files, specify a list of DXTFILE names with this keyword. The same GDI select exit needs to be specified for each GDI select file.

If you specify only one *dxtfile-name*, you can omit the parentheses. If there are special characters in the DXTFILE name, then you must enclose the name in double quotes (") as in the following example:

```
SUBMIT EXTID = REQUEST2,
       SELFILE = (FILE1, "¢FILE2", "FILE-3")
```

Note: For remote extracts, the files specified with SELFILE must be the same files specified with the FROM keyword of the EXTRACT statement.

GOIEXIT=*exitname* **(OPTIONAL)**
specifies a Generic Output Interface (GOI) exit routine.

GOIEXIT must be specified if you are using a GOI exit.

The extracted data will be passed, one line at a time, to the exit on a PUT call.

If you do not specify either the JCS keyword or the EXTDATA keyword, you must specify the GOIEXIT keyword.

If you specify EXTDATA and GOIEXIT, the information on the EXTDATA keyword is passed to the exit routine in the GOI description Control Block. If you specify CD=EXTDATA the Control Deck is passed to the exit routine on a CD call.

For more information about the GOI exit routine, see *DataRefresher Exit Routines.*

MAPEXIT=*exitname* **(REQUIRED FOR A DATAPROPAGATOR NONRELATIONAL PROPAGATION REQUEST, OTHERWISE, OPTIONAL)**
specifies a map capture exit routine.

MAPEXIT must be specified if you are using a map capture exit.

MAPEXIT cannot be used if SELFILE is included.

If you want to use a DataPropagator NonRelational Map Capture Exit routine, specify MAPEXIT=EKYMCE00. If you specify this exit routine, you must also supply the parameters listed in the MAPUPARM keyword.

For more information on the DataPropagator NonRelational Map Capture Exit routine, including the MAPUPARM parameters, see "Using a DataPropagator NonRelational Map Capture exit" on page 93.

MAPUPARM='userparms' (REQUIRED FOR A DATAPROPAGATOR NONRELATIONAL PROPAGATION REQUEST, OTHERWISE, OPTIONAL)
passes desired parameters to the map capture exit.

MAPUPARM can only be specified if MAPEXIT was specified.

The value of MAPUPARM can be up to 64K characters in length enclosed in quotes, and it is not examined or validated by DataRefresher.

If you specified MAPEXIT=EKYMCE00 to extract data using a DataPropagator NonRelational Map Capture Exit routine, you must specify certain MAPUPARM parameters. See "Using a DataPropagator NonRelational Map Capture exit" on page 93 for more information.

NULLS=Y | N (OPTIONAL)
lets you optionally suppress the inclusion of all DataRefresher null separator fields in the extract output.

Suppression of these fields lets you extract data for use by application programs and any DBMS that cannot handle the DataRefresher null separator field. Suppression of the null separator fields also reduces the amount of space needed for extract output, which can be a consideration when storing large amounts of extract output.

Y specifies that you want null separators inserted (this is what has always been done).

N specifies that you do not want null separators inserted for any columns under any condition. In this case, DataRefresher cannot indicate null values for any data columns, regardless of the null values you specify on the INTO keyword.

When extract output is directed to a file in IXF, NULLS=N means that only non-nullable IXF data types are put in the output file.

When source data is null or missing: When NULLS=Y and null or missing data is encountered, the DEM supplies nulls as usual.

When NULLS=N and the DEM encounters missing data it supplies default data as it does when NOT NULL columns are missing data, including:

- Blanks for character fields
- Zero length for variable character fields
- Earliest possible date/time for date/time fields
- Appropriate zero values for numeric fields

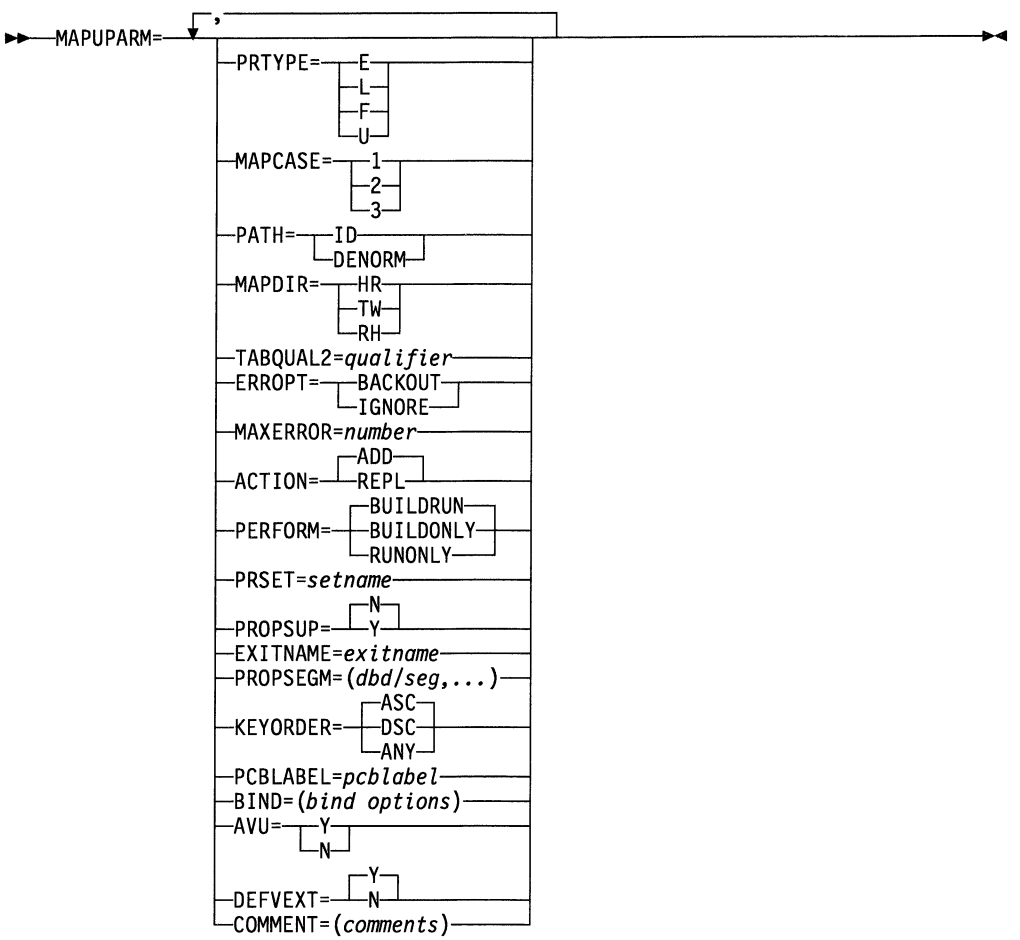
When source data is in error: Data errors are governed by the FLDERR keyword. Data errors can occur only when attempting to convert packed decimal, zoned decimal, date/time, or user data type source data to some target format.

When NULLS=Y, errors in source data will be handled as described for the FLDERR keyword.

When NULLS=N, there is an extension having to do with the FLDERR keyword. The (SUBST(NULL)) and (SUBST(NULL),*n*) options are invalid if a field of a data type capable of error is named for selection with SELECT. The UIM will reject the extract in this case.

Using a DataPropagator NonRelational Map Capture exit

It is possible to propagate extracted data using DataPropagator NonRelational. To do this, you need to specify MAPEXIT=EKYMCE00, and supply the necessary propagation parameters. These parameters are listed within the MAPUPARM keyword, and are shown in the following syntax:



For a DataPropagator NonRelational propagation request, the following keywords are applicable:

PRTYPE=E | L | F | U (REQUIRED)
identifies the type of extract/propagation request that you are creating.

The majority of IMS segments in an installation can be propagated using the generalized mapping logic of types E, L, and F. E supports more functions, including DB2-to-IMS synchronous propagation, and is therefore preferred to L or F.

There are four valid values:

E Extended function

An extended function request supports hierarchical-to-relational propagation, relational-to-hierarchical synchronous propagation, and two-way synchronous propagation.

L Limited function

An limited function request supports only hierarchical-to-relational propagation.

F Full function

PRTYPE=F is only supported by releases of DataPropagator NonRelational earlier than R1V2. In DataPropagator NonRelational R1V1 and earlier,

PRTYPE=F is handled the same as PRTYPE=L with the additional benefit of having CCU support. In DataPropagator NonRelational R1V2 and later, all PRTYPES have CCU support, thus PRTYPE=F is treated the same as PRTYPE=L.

U User mapping

Specifying PRTYPE=U indicates that you require mapping other than that provided in the generalized mapping cases supported by types E, L, and F, and that you are providing your own mapping case.

MAPCASE=1 | 2 | 3 (REQUIRED FOR PRTYPE= E, F, AND L)
identifies the mapping case to which the request belongs. If you selected PRTYPE=U this keyword is ignored. If specified it must contain a valid value.

1 Generalized mapping case 1. This provide mapping between IMS data of one IMS segment type (entity segment) and its concatenated key to/from one relational table.

2 Generalized mapping case 2. This provide mapping between IMS data of one IMS segment type (entity segment) and its concatenated key, and one or more immediately subordinated IMS segment types (extension segments) to/from one relational table. Each type of extension segment can occur once or not at all beneath the entity segment.

3 Generalized mapping case 3. This provide mapping between IMS data of one internal segment (entity segment) and its concatenated key to/from one relational table.

PATH=ID | DENORM (REQUIRED)
identifies the kind of non-key path data selected (from parent segment) of the request. The default is no path data selected.

The values you can specify are:

ID
includes as path data only IMS ID fields; that is, fields that cannot change their value.

DENORM
includes as path data fields that can change their value. For PRTYPE=E, you **cannot** specify PATH=DENORM.

MAPDIR=HR | TW | RH (REQUIRED)
indicates the direction of data propagation.

The values you can specify are:

HR Hierarchical-to-relational only.

TW Hierarchical-to-relational and relational-to-hierarchical.

RH Relational-to-hierarchical only.

Note: You **cannot** specify MAPDIR=TW or HR in the following cases:

- PRTYPE=L
- PRTYPE=F
- Asynchronous propagation
- User asynchronous propagation

TABQUAL2=*qualifier* (OPTIONAL)

If you specify an unqualified table name on the INTO clause of the EXTRACT statement for the propagated table, provide a TABQUAL2 keyword.

If you specify an unqualified table name on the INTO clause, DataPropagator NonRelational must find the description of a "model" table in the DB2 catalog. The model table can be (but does not have to be) the table being propagated. The model table should have the same attributes as the propagated table. DataPropagator NonRelational generates mapping information based on DB2 catalog descriptions of the model table.

You can specify the table name qualifier of the model table on the optional TABQUAL2 keyword (it must be a valid DB2 authorization ID). If you do not provide a TABQUAL2 keyword, then the user ID of the job is used as the default table name qualifier of the model table.

The name of the model table is comprised of:

- The qualifier specified on the TABQUAL2 keyword (or the user ID of the job).
- The unqualified table name specified on the INTO clause of the EXTRACT statement.

If you specify an unqualified table name on the INTO clause of the EXTRACT statement, DataPropagator NonRelational assumes that the model table has been created and has the same attributes as the table propagated by the request being defined.

ERROPT=BACKOUT | IGNORE (REQUIRED)

indicates the error option. The values you can specify are:

BACKOUT

If propagation cannot be successfully completed, and you want DataPropagator NonRelational to back out all changes since the last commit point, select BACKOUT.

IGNORE

Helps to minimize operational risks associated with propagation for IMS and DB2 applications. If propagation cannot be successfully completed (due to something other than unavailable DB2 resources) and you want to ignore the error and return to the caller, select IGNORE.

When you select ERROPT=IGNORE, DataPropagator NonRelational does not abend the application; instead it writes diagnostic information and returns to the program without indicating errors. While this reduces failures that could impact existing applications, it results in inconsistencies between the IMS and DB2 data.

You might want to use ERROPT=IGNORE during the early phases of propagation in a production system when you are first learning how to define a request. Later you can change to ERROPT=BACKOUT with the Status utility.

MAXERROR=*n* (OPTIONAL)

If you select ERROPT=IGNORE, the MAXERROR value indicates how many propagation failures can be reported for this request within a 15-minute interval. It also indicates how many errors will be documented with detailed information

in the trace. When this limit has been exceeded, propagation continues without error messages or detailed trace records being written.

Valid values for MAXERROR are between 0 and 32767 inclusive. The default value of 0 means *no limit*

This keyword limits only those messages written to the workstation and to the audit trail. It does not limit messages written to the optional data set and to the DataPropagator NonRelational trace.

ACTION=ADD | REPL (OPTIONAL)
indicates whether you want to add a new request or replace an existing one.

The values you can specify are:

ADD
When you select ADD, the DataPropagator NonRelational Map Capture Exit, EKYMCE00, first checks to see that the PR does not already exist in the mapping tables. If it does not a new request is created in the DataPropagator NonRelational directory and the SQL module libraries. Otherwise, EKYMCE00 issues an error message and returns a nonzero return code.

REPL
When you select REPL, the EKYMCE00 either replaces the already existing request or, if the request does not exist on the mapping tables, adds the request.

If you replace an existing request for synchronous propagation, the existing request must be inactive or DataPropagator NonRelational must be emergency stopped.

PERFORM=BUILDRUN | BUILDONLY | RUNONLY (OPTIONAL)
Indicates whether you want to:

- Create a request and run the data extraction (BUILDRUN)
- Create a request without extracting any data (BUILDONLY)
- Run the data extraction only (RUNONLY)

The values you can specify are:

BUILDRUN
DataPropagator NonRelational creates a propagation request and DataRefresher builds an extract request. If no errors are encountered during the creation of the propagation request, DataRefresher stores the extract request in the EXTLIB for the subsequent data extraction by the DEM.

BUILDONLY
A propagation request is created without extracting any data. BUILDONLY creates or replaces the propagation request in the DataPropagation NonRelational directory, but the UIM does not build an extract request. After the propagation request is generated, EKYMCE00 returns with a return code greater than 0, resulting in the data not being extracted.

For example, for synchronous propagation, you can code PERFORM=BUILDONLY for requests having a mapping direction of RH. For such requests, you will usually not extract the IMS data.

You can also use BUILDONLY when a DBD has been modified and receives a new version ID. If this modification does not affect the mapping (and therefore does not require another data extract), the database administrator can generate all requests propagating this database without performing an extract.

RUNONLY
Extracts data without replacing the existing request.

With this option, DataPropagator NonRelational's map verification and generation, (MVG), feature checks that the PR already exists and that it exactly matches the new request. If they match, and no other errors are encountered, DataRefresher stores the extract request definition into the EXTLIB for the subsequent data extraction. If they do not match, EKYMCE00 returns with a return code greater than 0, and DataRefresher does not extract the data. If you select RUNONLY, the MVG requires that the request be inactive, and ACTION=REPL.

PRSET=*setname* (OPTIONAL)
Identifies the set of requests to which this particular request belongs. If you do not provide a set name here, the name defined at the DataPropagator NonRelational generation will be used as a default.

Valid *setname* values are alphanumeric/special character names, as described in "DataRefresher naming conventions" on page 18.

PROPSUP= N | Y (OPTIONAL)
Indicates whether propagation should be suppressed when a Segment Exit routine issues a return code of 8.

N Select N if you do not want to allow Segment Exit routines to return a code of 8 during data propagation. When Segment Exit routines return with a return code of 8, it is considered an error, and DataPropagator NonRelational issues an abend.

Y If you select Y and a Segment Exit routine results in inconsistencies between the IMS data copy and the DB2 data copy. These inconsistencies might eventually result in propagation failures of other updates. Therefore, suppress propagation with caution.

If you choose Y and you do not specify the USE keyword of the CCU CHECK control statement, CCU executions could report inconsistencies where none likely exist.

EXITNAME=*exitname* (REQUIRED FOR USER MAPPING)
Identifies the Propagation Exit routine performing propagation.

Do not specify this keyword for a request belonging to a generalized mapping case.

PROPSEGM=*(dbd/seg)* (REQUIRED FOR USER MAPPING)
Identifies the IMS segments to be propagated by the request being defined. Whenever one of the segments that you identified changes, the Propagation Exit routine you specified on EXITNAME is called by the Relational Update Program, (RUP), to process the changed segments.

Select your segments as follows:

```
(dbd/seg,seg,seg,....,dbd/seg,seg,seg,...)
```

Specify the database first and its segments next, followed by a second database, if any. Do not specify PROPSEGM for a request belonging to a generalized mapping case.

If multiple requests (with the same or different Propagation Exit routine), identify the same segment on this keyword, the RUP issues multiple calls to the Propagation Exit routines whenever that segment is changed.

This propagation parameter is used only by the RUP; that is, when the data is propagated from IMS to DB2. For DB2-to-IMS synchronous propagation, this parameter has no effect; the Propagation Exit routine is called whenever the table specified on the INTO clause of the EXTRACT statement is updated.

KEYORDER=ASC | DSC | ANY (REQUIRED)
Indicates the column sequence of the primary key index of the propagated table. The values you can specify are:

ASC All columns composing the primary key of the propagated table are in ascending sequence.

DSC All columns composing the primary key of the propagated table are in descending sequence.

ANY Indicates that MVG must access the DB2 catalog to get the defined ordering sequence of each column composing the primary key of the propagated table.

If you specify ANY, DataPropagator NonRelational's access to the DB2 catalog is typically lengthy and inefficient. Therefore, if you know that all primary key columns have been defined in the DB2 index as ASC or DSC, define them as ASC or DSC here.

If you incorrectly specify ASC or DSC, DataPropagator NonRelational's Consistent Check Utility, (CCU), might report inconsistencies that do not exist.

PCBLABEL=*pcb name* (OPTIONAL)
This keyword is only valid for SYNCHRONOUS propagation.

Identifies the name of the IMS PCB that the Hierarchical Update Program, (HUP), uses to perform DB2-to-IMS synchronous propagation.

If this parameter is omitted and the request is used to perform DB2-to-IMS synchronous propagation, the name of the DXTPCB is used to identify the name of the IMS PCB that the HUP uses.

BIND=*text* (OPTIONAL)
Contains BIND options for the SQL update module. The default is no bind options.

If you want DataPropagator NonRelational to automatically bind the package of the SQL update module, you have to specify bind options, including the collection ID where the package is to be bound.

The bind options are provided in the form of a string of DB2 specifications, up to a maximum of 254 bytes. Enclose the DB2 specifications in parentheses, without a continuation character.

DataPropagator NonRelational ensures that the following keywords, which are not supported, are not specified in the string:

- MEMBER

- BIND
- PLAN
- COPY
- LIBRARY

If you do not specify the collection ID in the PACKAGE keyword, the other bind options are ignored and no package is created.

Recommendations for BIND options

- For performance reasons, specify VALIDATE(BIND) rather than the DB2 default of VALIDATE(RUN). This prevents DB2 from performing a package validation each time the RUP invokes an SQL update module.
- If you have specified an unqualified propagated table name in your propagation request, you should use the QUALIFIER operand to set the correct qualifier.

AVU=Y | N (OPTIONAL)

Avoids unnecessary updates. With this parameter, you can influence the performance of IMS-to-DB2 propagation. This parameter specifies whether or not DataPropagator NonRelational should determine (during a segment replace) whether at least one propagated field has changed.

Y The replace of an IMS segment results in a propagating SQL update, only if at least one propagated field has changed.

N The replace of an IMS segment results in a propagating SQL update, even if no propagated field has changed.

AVU=N can reduce the total path length when, for example, all fields of a segment are propagated. When you specify AVU=N, DataPropagator NonRelational does not compare each field to determine if at least one of them has changed.

If you do not allow AVU, DataPropagator NonRelational determines a default setting at propagation time, as follows:

- AVU=Y is the default value:
  - When propagating internal segments with mapping case 3 requests
  - When propagating IMS segments with mapping case 1 and 2 requests, and at least one non-key byte of the segment is propagated
- AVU=N is the default in all other cases.

DEFVEXT=Y | N (OPTIONAL)

This keyword is only valid for SYNCHRONOUS propagation.

Identifies how DataPropagator NonRelational handles extension segments propagated exclusively from default values and NULLs during DB2-to-IMS synchronous propagation.

This parameter is used only for mapping case 2 requests performing DB2-to-IMS synchronous propagation. It tells DataPropagator NonRelational what to do with an extension segment during DB2-to-IMS synchronous propagation of an SQL INSERT or UPDATE:

- When all the target columns of an extension segment are either null or contain default values, and

- When the SQL INSERT/UPDATE changes at least one column mapped to the extension segment.

Y This is the default value. When DEFVEXT=Y, the propagation of the SQL UPDATE/INSERT results in zero occurrences of the extension segment type.

N When DEFVEXT=N, the propagation of the SQL UPDATE/INSERT results in zero occurrences of the extension segment type, even if all of its fields are propagated from columns having a NULL or default value.

COMMENT=*(comments)* (OPTIONAL)
Specify up to 254 bytes of comments to document the request. These comments are stored in the request mapping table.

EXTRACT statement

EXTRACT (REQUIRED)
extracts data from a DEM data source.

WITH CREATE | REPLACE (OPTIONAL)
is specified if you want your extracted data to be used to either:

- Create a new DB2 or SQL/DS table (WITH CREATE)
- Replace all the data in an existing DB2 or SQL/DS table (WITH REPLACE)

Do not use the WITH keyword if you want your extracted data:

- Added to an existing DB2 or SQL/DS table
- Loaded into a physical sequential data set (refer to the EXTDATA keyword of the SUBMIT command)
- Loaded into an empty DB2 table.

When you write the WITH keyword, the new or replaced DB2 or SQL/DS table has the name you specified with the INTO keyword; the columns in the resulting table have the names you specified following the table name with the INTO keyword. If you specify FORMAT=SOURCE on the SUBMIT statement, then the data type of a target column depends on the data type of the source column. For more information about the data types of source columns, see your database administrator or the DB2 catalog tables. If you specify FORMAT=EBCDIC on the SUBMIT statement, then the data types of target columns depend on the data types in the control deck as defined at Table 2 on page 89 and Table 3 on page 89.

The following table gives the input/output data types for a DB2 table that is created.

If the source field is type ...	The target column data type is...
A	DATE
B	SMALLINT
C, bytes=n	CHAR(n)
VC, bytes=n <= 254	VARCHAR(n)
VC, bytes=n > 254	LONG VARCHAR
D	FLOAT
E	FLOAT
F	INTEGER
G, bytes=n	GRAPHIC (n/2)
VG, bytes=n <= 254	VARGRAPHIC (n/2)
VG, bytes=n > 254	LONG VARGRAPHIC
H	SMALLINT
P, bytes=n, scale=m	DECIMAL (2n-1,m)
S	TIMESTAMP
T	TIME
Z, bytes=n, scale=m	DECIMAL (n,m)

Notes:

1. If you specify WITH CREATE, CD=JCS and DBS=DB2, the *TC marker must be in your JCS to tell DataRefresher where to write the CREATE TABLE command.
2. If you specify WITH CREATE, CD=EXTDATA and GOIEXIT, then the Table Create statements will be passed, one line at a time, to the GOI exit on a CD call.

The following table gives the input/output data types for a SQL/DS table that is created.

If the source field is type ...	The target column data type is...
A	DATE
B	SMALLINT
C, VC, bytes=n	CHAR(n)
D	FLOAT
E	FLOAT
F	INTEGER
G, VG	GRAPHIC
H	SMALLINT
P, bytes=n, scale=m	DECIMAL (2n-1,m)
S	TIMESTAMP
T	TIME
Z, bytes=n, scale=m	DECIMAL (n,m)

If you are going to produce a control deck, and need to know the target data types for different source field data types, use the previous table.

If you create a data table in an extract request using WITH CREATE, you can specify whether its columns can contain null values with the absence or presence of NOT NULL on the INTO keyword.

INTO *tableid (columnname(s))* **(OPTIONAL)**
identifies the name of the table and the table columns into which the extracted data is to be inserted. You need the INTO keyword if the extracted data is to be loaded into a relational database table.

If an extract request contains the INTO keyword, the DEM generates the load command necessary to load the extracted data into a table.

tableid
specifies the ID of the relational database table into which you want your data inserted.

tableid has the form:

authid.tablename

authid
is the authorization ID. It must be an alphameric/special characters name.

You do not have to include the *authid*. However, if you do not do so for DB2, the load utility will assume that it is the same as the user ID written in the JOB statement of the JCS data set associated with your extract request. If no user ID was specified on the JOB statement, DB2 will use the default user ID for batch jobs that was defined when DB2 was installed.

For SQL/DS, do not specify an authorization ID. SQL/DS will use the user ID of the CONNECT statement in the JCS data set that is associated with your extract request.

If you do not write an authorization ID, do not write the period in front of *tablename.*

tablename
specifies the relational database table into which the extracted data is loaded. The *tablename* must be written. It can be either an SQL name, an SQL quoted name, or a DBCS name.

columnname(s)
specifies the name(s) of the column(s) into which you want the extracted data inserted.

You can write up to 750 *columnnames*, the maximum number of columns allowed in a relational database table. A *columnname* can be either an SQL name, an SQL quoted name, or a DBCS name. If you write more than one column name, separate them with commas.

Specifying nulls

If one or more columns cannot contain null values, you must follow the column name with NOT NULL. If this is not coded, the DEM assumes that the column may contain null data and will produce a NULLIF statement in the generated control statement for the column.

DB2 also allows a column to be created as NOT NULL WITH DEFAULT. When you code DBS=DB2, the DEM allows NOT NULL WITH DEFAULT for a given column definition on the INTO keyword.

If a column is defined as NOT NULL WITH DEFAULT, the DEM includes a DEFAULTIF keyword as part of the description of the relevant column when generating the DB2 LOAD control deck. This causes the DB2 LOAD utility to load zeros into numeric fields and blanks into character fields wherever DEM null indicators are found.

Ordering columns

The order of your column names is important and depends on how you write the SELECT statement and INTO statements.

The order you write the field names on the SELECT statement is the order they are inserted in the columns named on the INTO statement.

The number of columns you name on the INTO statement must be equal to the number of fields you name on the SELECT statement. You can select duplicate fields with the SELECT statement, but you cannot specify duplicate fields on the INTO statement.

If you do not know the order in which the source columns are defined in their relational tables, see your database administrator or query the relational database catalog tables.

Ordering columns when specifying SELECT *

Specifying * for the SELECT statement will select all the columns from a source table. In this case, your target columns are matched with the columns contained in the relevant source table(s).

The number of columns you name on the INTO statement must be equal to the number of columns listed in the relevant source table(s).

Specifying columns for IXF

The IXF table record gets the value of IXFTNAME (name of data) from:

- INTO *tableid*, if specified
- EXTID keyword of the SUBMIT command, if INTO was not specified

The value of IXFTNAML (length of name specified in IXFTNAME) is always 18.

The IXF column descriptor record gets the value of IXFCNAME (name of a column) from:

- INTO *columnname(s)*, if specified
- source data names

The value of IXFCNAML (length of name specified in IXFCNAME) is always 18.

NOT NULL WITH DEFAULT may not be specified for data in IXF.

Reasons for termination or cancellation

If the number of columns specified on the INTO statement does not equal the number of columns of extracted data, the request will not be queued for execution.

If the source data has null values and you are entering data into a table with the NOT NULL option, the DEM will give an error message and terminate.

OPTIONS (OPTIONAL)
affect:

- How much output your extract request produces (OUT)
- When the DEM executes your extract request (PRI)
- How the DEM handles field errors when executing your extract request (FLDERR)
- How much diagnostic information the DEM maintains when executing your extract request. (DEBUG,FLDMSG)

OUT(NONE | *n*) (OPTIONAL)
controls the number of output rows the DEM can generate when executing your extract request.

(NONE)
indicates that there is no limit on the number of output rows that the DEM can generate when executing your extract request. If you specify OUT(NONE) your extract request can only execute when the DEM also has an output limit of NONE, specified on the INITDEM command.

(*n*) indicates the maximum number of output rows you want generated when the DEM executes your extract request.

If this number is reached, the DEM will stop writing your output, even if your output is not yet complete.

Replace *n* with a number from 1 through 10000000, written without commas.

The purpose of the OUT keyword is to let your installation schedule execution of your extract requests on the basis of size. For example, your installation could set up the DEM to run small extract requests during the day and large extract requests overnight.

PRI(0 | *n*) (OPTIONAL)
controls when your extract request will be executed. Replace *n* with a number from 0 through 255.

Generally, the larger the priority number, the more quickly an extract request will be executed. The DEM can execute an extract request only if the extract request's priority number falls within its priority range. See the *DataRefresher MVS and VM User's Guide* for more information about DEM priority ranges.

FLDERR(*value*)
controls what the DEM does if it encounters a field with an invalid format while running your extract request. The DEM can encounter an incorrectly formatted field in a field named on the SELECT statement or a field named on the WHERE statement.

If a field named on the SELECT statement is incorrectly formatted, the DEM responds based on the information you specify on the FLDERR keyword. If a field named in the WHERE statement is incorrectly formatted, the DEM:

- Considers the WHERE condition false
- Adds 1 through *n* (if you specified a form of FLDERR containing *n*)
- Does not substitute a value (if you specified a form of FLDERR containing SUBST).

The values you can specify with the FLDERR keyword are:

SUBST(NULL)
the DEM substitutes nulls when it writes the contents of the fields causing the field errors.

HALT
the DEM stops running your extract request when it encounters a field error.

SKIP
the DEM skips the record containing the field error and continues running your extract request.

SKIP, *n*
the DEM stops after skipping a given number of field errors while executing your extract request.

Replace *n* with the number (1 through 10000) of field errors that you want to allow the DEM to skip. If the DEM encounters *n*+1 field errors, it will stop executing your extract request.

SUBST(ZERO)
the DEM substitutes zeros when it writes the contents of numeric fields causing the field errors. The DEM will write blanks for any other type of field (character, graphic, and so on).

SUBST(NULL), *n*
the DEM substitutes nulls when it writes the contents of the fields causing the field errors and stop executing your extract request after a given number of substitutions.

Replace *n* with the number (1 through 10000) of field errors that you will allow the DEM to substitute in your extract request. If the DEM encounters *n*+1 field errors, it will stop executing it your extract request.

SUBST(ZERO), *n*
the DEM substitutes zeros when it writes the contents of numeric fields causing the field errors and to stops executing your extract request after a given number of substitutions. The DEM will write blanks for any other type of field (character, graphic, and so on).

Replace *n* with the number (1 through 10000) of field errors that you will allow the DEM to substitute in your extract request. If the DEM encounters *n*+1 field errors, it will stop executing your extract request.

Notes:

1. If the execution of an extract request is stopped because of a field error, the DEM writes all of the output resulting from the extract request up to the point of the field error and removes the extract request from the EXTLIB.
2. If you indicate that a target column cannot contain a null value, (with NOT NULL on the INTO statement) and if the selected field is packed decimal, zoned decimal, date/time, or user-defined, you must specify a FLDERR value other than (SUBST(NULL)) or (SUBST(NULL),*n*). (These types of data are the only types for which conversion errors are possible.) If you specify that the DEM substitute nulls in that situation, the DEM will generate nulls, causing the load utility to terminate with an error.
3. If you specify FLDERR(SUBST(NULL)) or FLDERR(SUBST(NULL),*n*), and the column allows nulls, the DEM will put a NULL IF statement for each target column into the load command that it generates. If, however, one of the columns in the target table cannot contain a null the value, NULL IF statement will cause the load utility to terminate with an error. Therefore, if a target column cannot contain a null value, you must indicate a FLDERR value other than (SUBST(NULL)) or (SUBST(NULL),*n*).

When your job stops because of field errors, you can resubmit the job after:

- Fixing the error (if you specified FLDERR(HALT))
- Increasing the value of *n* (if you specified FLDERR(SKIP,*n*))
- Changing the keyword to FLDERR(SKIP) (if you specified FLDERR(HALT) or FLDERR(SKIP,*n*)).

DEBUG(*n*) (OPTIONAL)
affects what diagnostic information the DEM maintains and returns when executing your extract request.

Replace *n* with a number from 1 through 4. See Appendix E, "Diagnostic information" on page 257 for a description of the type of diagnostic information associated with each debugging level.

The default value for *n* is 1.

FLDMSG(*n*) (OPTIONAL)
limits the number of field format errors to be diagnosed with messages. If more than *n* field format errors occur, no details will be presented on the *n+1*st and succeeding errors. Instead, a summary message telling how many field errors were encountered will be written.

FLDMSG(*n*) is written as a string of digits without commas, where *n* is any number from 0 through 1000000 (one million).

The default value for *n* is 100.

SELECT * | *(fieldname(s))* (REQUIRED)
identifies the field(s) from which you want to extract data.

The set of conditions that are permitted in this statement are a subset of the conditions accepted by SQL and DB2. The rules that apply when specifying the *search condition* on the WHERE statement are explained in "SELECT statement conditions" on page 110.

In the case of a GDI select exit, however, the SELECT statement is passed intact to the exit (without the semicolon). This means that the SQL supported by a GDI select exit is not restricted to the DataRefresher subset; your SELECT statement can contain other SQL statements.

*** (asterisk)**
indicates that you want to extract data from all of the fields contained in the DXTVIEW(s) named in the EXTRACT statement's FROM keyword.

If you write * in the SELECT statement and the names of two DXTVIEWs with common field names on the FROM keyword, data will be extracted from all of the fields contained in the first DXTVIEW, but only from the non-overlapping fields contained in the second DXTVIEW.

fieldname
indicates the name(s) of the field(s) from which you want to extract data. The field(s) must be contained in the DXTVIEW(s) named on the FROM keyword. If your name contains special characters, you must enclose it in double quotes ("). For more information on naming conventions, see "DataRefresher naming conventions" on page 18.

Write *fieldname* as:

viewname.fieldname

where *fieldname* is the name (or alias) of the field from which you want to extract data and *viewname* is the name of the DXTVIEW containing the field. Use *viewname* if:

- You want to extract data from fields contained in two different DXTVIEWs

  and

- The name (or alias) of the field you want to extract data from is identical to the name (or alias) of one of the fields contained in the other DXTVIEW.

If you do not write the view name, do not write the period in front of the field name.

You can write up to 750 *fieldnames* (separate them with commas).

FROM *viewname(s)* **(REQUIRED)**
identifies the DXTVIEW(s) containing the fields from which you want to extract data.

Replace *viewname* with the name(s) of the relevant DXTVIEW(s).

You can specify up to 16 views.

Separate multiple view names with commas. If your DXTVIEW name contains special characters, you must enclose it in double quotes ("). See "DataRefresher naming conventions" on page 18 for more information on naming conventions.

Specifying multiple views: You can specify views based on different DXTPCB descriptions or different DataRefresher file descriptions, using the same FROM keyword.

When you specify two or more views in a FROM keyword, DataRefresher extracts data from each view and combines it to form one output row. If you do

not want DataRefresher to generate all the possible combinations (which could result in a great number of unnecessary reads, unnecessary output, and could possibly cause the extract to fail if an output limit is exceeded), you can improve DataRefresher's performance by writing a WHERE statement. A WHERE statement compares a sequence, sort, or key field in one view to a field in another view, and joins the data based upon this match. See the WHERE statement discussion for more information about extracting from multiple views.

The views you specify with the FROM keyword can be a combination of views to which you have access authorization. The rules to follow when specifying more than one view with one FROM keyword include:

1. You cannot specify two or more views of multiple record segments that exist in one DXTFILE. This does not apply to internal segments within one parent segment, only to two or more different multiple record segments.
2. You cannot specify views of PCBs in different PSBs. You may specify views of different PCBs in the same PSB however.

WHERE *search condition* **(OPTIONAL)**
specifies the conditions that the data must satisfy to be extracted.

Notes:

1. Data returned from a data exit as a result of an EOD call will be subject to the conditions specified with WHERE. For more information about EOD calls, see *DataRefresher Exit Routines*.
2. If you use the HSSR to access IMS data and specify the WHERE keyword, the HSSR can slow down processing performance.
3. Optimal numeric key performance is achieved when the "=" boolean operator is used in the extract request.

ORDER BY *columnname(s)* **ASC** I **DESC (OPTIONAL)**
specifies the sequence of the extracted data within one or more named columns. The DEM implements the order you select by invoking the standard MVS Sort Program. You therefore also need to include DD statements for Sort Work files in the JCL which invokes the DEM if you specify a value for the ORDER BY keyword.

columnname
identifies a column from a DXTVIEW named in the SELECT statement. The data within the named column will be ordered in either ascending or descending order. Without the ORDER BY keyword, the data will be sequenced according to the order of the original data. A maximum of 8 column names may be specified in the ORDER BY keyword.

ASC
indicates that the data within the named column will be sequenced in ascending order.

DESC
indicates that the data within the named column will be sequenced in descending order.

SELECT statement conditions

Naming fields

- Any fields you name in your WHERE statement must be contained in the DXTVIEW(s) that you name with FROM, but they do not have to be listed with your SELECT statement.
- When your FROM keyword lists two DXTVIEWs that contain fields with the same name, and you want to include one of those fields on a WHERE statement or SELECT statement, qualify the field with the name of its corresponding DXTVIEW. Instead of specifying only the field name, also specify its view name in order to distinguish it from a field of the same name in the other view.
- When a field you want to include on the WHERE statement has been assigned an alias, use the alias (not the field name).

Multiple conditions

- If you write more than one condition, separate them with boolean operators such as AND, OR, or NOT.
- The usual order of evaluation for boolean operators is shown in the following table:

Order	Operator
1	( )
2	NOT
3	AND
4	OR

Use parentheses to control the order in which boolean expressions will be evaluated.

Consecutive NOT operators

- Do not use consecutive NOT operators. If you use two NOTS before an expression, as in:

```
WHERE NOT NOT A=B
```

the UIM considers the second NOT to be a field name. Because the qualifier (=) does not follow the field name (or what the UIM regards as the field name), the UIM produces an error message. In this example:

```
WHERE NOT NOT=B
```

the second NOT is the name of a field. The qualifier does follow the fieldname, so the UIM will not produce an error message. You can use NOT as a field name in DataRefresher, but for the reason shown in the above example, it is not advisable.

Numeric and character constants

- The value of a numeric or character constant you write with WHERE must fall within the range of allowable values for its associated field type. When you write a constant that falls outside the allowed range for the field, the UIM does

not allow the extract request to be processed. The following table lists the different field types and the allowable ranges for their constants.

Field Type	Sample Field Format	Allowable Range for Constant
A	1994-9-1	8 to 10 bytes
B	12	1 through 255
C, VC	ABC@123	1 to 254 characters
D	(±)123.4E(±)12	5.4E-79 to 7.2E+75
E	(±)123.4E(±)12	5.4E-79 to 7.2E+75
F	(±)123	-2147483648 to +2147483647
G, VG		1 to 127 DBCS characters, (2 to 254 bytes, must be even number of bytes)
H	(±)123	-32768 to +32767
P	(±)123.456	31 significant digits
S	1966-2-5-10.45.23.01	16 to 26 bytes
T	4.01	4 to 8 bytes
Z	(±)123.456	16 significant digits

Comparisons

The comparisons that can be made with WHERE are:

```
=, >, >=, <, <=, ¬=, LIKE or NOT LIKE,
  IN or NOT IN, and BETWEEN
```

- Fields containing date/time data can contain the following comparisons:

  ```
  =, >, >=, <, <=, ¬=, IN, NOT IN, and BETWEEN
  ```

  Date/time constants are considered character data, and must be enclosed in single quotes.

  Date constants can be in the formats:
  'mm/dd/yyyy'
  'dd.mm.yyyy'
  'yyyy-mm-dd'
  'mm/dd/yy'
  'mm—dd—yy'

  Leading zeros can be omitted from the month and the day.

  Time constants can be in the following formats:

  'hh.mm xM where x=A or x=P'
  'hh.mm.ss'
  'hh:mm:ss'

  Leading zeros can be omitted from the hour.

  Timestamp constants must be in the format

  ```
  'yyyy-mm-dd-hh.mm.ss.nnnnnn'
  ```

  Leading zeros can be omitted from the month, day, and hour.

- Fields containing DBCS data can contain the following comparisons:

  ```
  =, >, >=, <, <=, ¬=, IN, NOT IN, or BETWEEN
  ```

Graphic literals: If the WHERE condition contains a graphic (DBCS) literal, the literal must be enclosed in single quotes and be preceded by a 'G'. In addition, the data must be bracketed by shift-out (X'0E') character and a shift-in (X'0F') character, as in:

```
WHERE NAME=G'<D_B_C_S_N_A_M_E_>'
```

(X'0E' is represented by < and X'0F' by >.)

Graphic literals may contain only DBCS data; single character data is not allowed in a graphic literal. In addition, graphic literals may only be compared with graphic fields. In the above example, then, NAME must be a field of type graphic.

Mixed literals: If the WHERE statement contains a mixed literal (both DBCS and single-byte data), the literal must be enclosed in single quotes. The graphic data in the mixed string must be bracketed by a shift-out (X'0E') and shift-in (X'0F') characters, as in

```
WHERE JOB='abcdef<D_B_C_S_>'
```

(X'0E' is represented by < and X'0F' by >.) In this example, abcdef represents single-byte data and D_B_C_S represents graphic data. Mixed literals may only be compared with fields containing character data. In the above example, JOB must be a field of character type data.

Mixed strings are indicated by specifying the UIM keyword MIXED=YES. If this is not specified, all quoted strings will be regarded as simple character strings. X'0E' and X'0F' will have no special significance.

G fields of unequal length: If a comparison is made between two G fields of unequal length, DataRefresher will pad the shorter DBCS field with DBCS blanks (X'4040') when doing the comparison.

User data types: If a comparison is made using a user data type, the comparison is made after the user data type exit has converted the field from its user data type format into its target format in a DataRefresher data type. For example,

```
WHERE UDTFIELD='CHARACTER_OUTPUT'
```

In this WHERE condition, the field called UDTFIELD is a field defined to be in user data type format ZZ. The ZZ data type was defined as having a target data type of character (C) format of length 16 bytes. Before the WHERE condition is evaluated, the user data type exit for data type ZZ is called to convert this field into its target (character) data type. If there are no errors from the data type exit, the field is converted, then the data returned from the data type exit is compared with the WHERE condition 'CHARACTER_OUTPUT'.

Forming the conditions

The following are acceptable forms of a condition:

- *operand1 simple comparison operand2* where:

  operand1 **and** *operand2*
  : are field names, numeric constants, or character strings, but at least one of the two operands must be a field name. The two operands must have compatible data types and compatible lengths. That is, the field must be able to contain the value to which it is being compared.

If both operands are field names, they must both contain numeric data, or they must both contain character data. If only one of the operands is a field name and that field contains numeric data, the other operand must be a numeric value. If only one of the operands is a field name and that field contains character data, the other operand must be a character string (written in single quotes).

simple comparison
is a comparison to be made between the operands.

If the operands represent numeric data, arithmetic comparisons are made. If the operands represent character data, logical comparisons are made. DBCS fields (G fields) can only be compared to DBCS fields, and an EBCDIC collating sequence is used.

Note: Although date/time data type constants and field values may be presented to DataRefresher in a variety of formats, the comparisons are performed in ISO so that they yield the answer you want. For example '01/01/1994' > '12/12/93' because the ISO values are '1994-01-01' and '1993-12-12' respectively.

Examples of a few simple comparisons and their corresponding values are represented in the following table:

You typed	Field1 has	Condition is
field1 = 20	a numeric 10	false
field1 < 'ABCD'	character 'ABC'	false
20 ¬= field1	a numeric 10	true

More comparisons include:

- *fieldname* LIKE / NOT LIKE *characters* where:

  fieldname
  must be the name of a field containing character data.

  characters
  must be a character string (written in single quotes).

  When LIKE is the comparison, the condition is true if the character data in the named field matches the character string. When NOT LIKE is the comparison, the condition is true if the character data in the named field does not match the character string.

  The underscore (_) and the percent sign (%) have special meanings when coded as part of a character string, and LIKE or NOT LIKE is the comparison. The underscore (_) indicates that the corresponding (single) character in the named field is not considered in the comparison. In other words, that character can be any single character. The percent sign (%) indicates that an unspecified number of characters (0 or more) are not considered in the comparison. In other words, that set of characters can be any combination of 0 or more characters.

  The LIKE and NOT LIKE comparisons are not valid for DBCS data.

A few examples and corresponding value are displayed in the following table:

You typed	field1 has	Condition is
field1 LIKE 'ABC'	character 'XYZ'	false
field1 LIKE 'ABC'	character 'ABC'	true
field1 LIKE 'A_C'	character 'ABD'	false
field1 LIKE 'A_C'	character 'ABC'	true
field1 LIKE '%A%C%'	character 'AC'	true
field1 LIKE '%A%C'	character 'FLACK'	false
field1 NOT LIKE '%A%C'	character 'FLACK'	true

- *fieldname* IN / NOT IN *(list)* where:

  fieldname
  is the name of a field. The field can contain either numeric or character data.

  (list)
  can be a series of either numeric constants or of character strings (written in single quotes), depending on whether the named field contains numeric or character data. Write up to 255 numeric constants or character strings (separate them with commas).

  If IN is the comparison, it is true if the data in the named field is equal to one of the numeric constants or character strings in the list. If NOT IN is the comparison, it is true if the data in the named field is not equal to any of the numeric constants or character strings in the list.

  The IN and NOT IN comparisons are not allowed for DBCS data.

  The following table contains a list of IN and NOT IN comparisons and their values.

You typed	field1 has	Condition is
field1 IN (10, 20, 30)	a numeric 10	true
field1 IN (20, 30, 40)	a numeric 10	false
field1 IN ('ABC', 'RST', 'XYZ')	character 'ABC'	true
field1 IN ('ABCD', 'RST', 'XYZ')	character 'ABC'	false

  Note: DataRefresher executes the OR comparison and the IN comparison differently. Thus, in a user error situation, different results may occur.

- *fieldname* BETWEEN *operand1* AND *operand2* where:

  fieldname
  is the name of a field. The field can contain either numeric or character data.

  operand1 **and** *operand2*
  are fields containing numeric or character data, or numeric or character constants.

If the named field contains numeric data, *operand1* can be either a numeric constant or the name of a field containing numeric data, and *operand2* can be either a numeric constant or the name of a field containing numeric data.

If the named field contains character data, *operand1* can be either a character string (written in single quotes) or the name of a field containing character data, and *operand2* can be either a character string (written in single quotes) or the name of a field containing character data.

If this form of the BETWEEN comparison is used, the comparison is true if the data in the named field is greater than or equal to *operand1* and less than or equal to *operand2*. Thus, this comparison is equivalent to:

```
fieldname >= operand1 AND fieldname <= operand2
```

Note that the order of the values is significant.

For DBCS data (G fields), comparisons are made only with other DBCS fields, and EBCDIC collating sequence is used.

The following table contains a list of BETWEEN comparisons, and their corresponding values.

You typed	field1 has	Condition is
field1 BETWEEN 10 AND 20	a numeric 15	true
field1 BETWEEN 'ABB' AND 'ABD'	character 'ABC'	true
field1 BETWEEN 1024 and 2048	a numeric 512	false

- *operand* BETWEEN *fieldname1* AND *fieldname2* where:

  operand
  : can be either a numeric constant or a character string (written in single quotes).

  fieldname1 **and** *fieldname2*
  : are the names of fields. The fields can contain either numeric or character data depending on whether the *operand* is, respectively, a numeric constant or character data.

If this form of the BETWEEN comparison is used, the comparison is true if the *operand* is greater than or equal to the data in *fieldname1* and less than or equal to the data in *fieldname2*. Thus, this qualifier is equivalent to:

```
operand >= fieldname1 AND operand <= fieldname2
```

The order of the field names is significant.

The following table contains a list of BETWEEN comparisons and their corresponding values.

You typed	field1 has	field2 has	Condition is
15 BETWEEN field1 AND field2	a numeric 10	a numeric 20	true
25 BETWEEN field1 AND field2	a numeric 10	a numeric 20	false
'ABC' BETWEEN field1 AND field2	character 'ABB'	character 'ABD'	true

Examples of SUBMIT and EXTRACT

There are two types of examples:

- SUBMIT statement
- EXTRACT statement

The two statements of this command are both very detailed, so there are separate examples for each.

Examples of the SUBMIT command

Example 1

In this example:

- You want to write an extract request having an extract ID of EXTREQ1, and a user ID of SMITH.
- You want messages issued by DataRefresher about your extract request sent to the DB2 table named DXTMSGS.
- The JCS data set having the name DB2JCS can be used by both your extracted data and the messages about your extract request to load data into the desired DB2 tables.
- You want to load your extract output into the DB2 system.
- You want DataRefresher to generate a DB2 LOAD utility control deck and use DB2JCS.
- You want DataRefresher to convert the extracted data to EBCDIC format.

```
SUBMIT EXTID=EXTREQ1,USERID=SMITH,
       MSGS=(DB2JCS,DXTMSGS),DBS=DB2,JCS=DB2JCS,CD=JCS,
       FORMAT=EBCDIC
```

Example 2

In this example:

- You want to write an extract request having an extract ID of EXTREQ1.
- You want messages issued by DataRefresher sent to the DB2 table named DXTMSGS.
- The JCS data set having the name DB2JCS can load both your extracted data and the messages into the desired DB2 tables.
- You want to load your extract output into the DB2 system.
- You want DataRefresher to generate a DB2 LOAD utility control deck and use DB2JCS.
- You want DataRefresher to leave the extracted data in source format.

```
SUBMIT EXTID=EXTREQ1,MSGS=(DB2JCS,DXTMSGS),
       DBS=DB2,JCS=DB2JCS,CD=JCS,FORMAT=SOURCE
```

Example 3

In this example:

- You want to write an extract request having an extract ID of EXTREQ2, and a user ID of JONES.
- You want messages issued by DataRefresher sent to the DB2 table named DXTMSGS.
- The JCS data set having the name DB2JCS can load the messages into the desired DB2 table.
- You do not indicate into which database system you want your messages loaded, so the DBS keyword defaults to DB2.
- You want DataRefresher to generate column descriptors and place them in a physical sequential data set.
- You want all the output from your extract request written to the dynamically allocated data set PSOUT.
- You do not indicate in which format DataRefresher should extract the data that you request. The FORMAT keyword defaults to EBCDIC.

```
SUBMIT EXTID=EXTREQ2,USERID=JONES,
       MSGS=(DB2JCS,DXTMSGS),CD=EXTDATA,EXTDATA=(PSOUT)
```

Examples of the EXTRACT (UIM) statement

Example 1

In this example, you want:

- To insert your extracted data into a DB2 table named DB2TABLE, in the columns named DB2DNUM, DB2ENUM, DB2ENAME, DB2ONUM, and DB2COMM.
- The DEM to stop executing your extract request as soon as it has written 100,000 output rows—even if it has not completed your output.
- Your extract request to have a priority value of 150.
- The DEM to stop executing your extract request as soon as it encounters a field error in an input field.
- The DEM to maintain debugging level 2 diagnostic information while executing your extract request.
- To extract data from fields contained in the DXTVIEW named SALEVIEW (you have been authorized to access SALEVIEW).
- To extract data from the fields contained in SALEVIEW named DNUM, ENUM, ENAME, ONUM, and COMM.
- To extract data only about departments numbered 15, 20, and 38, and only about employees having salaries of at least $25,000.

```
EXTRACT INTO DB2TABLE
(DB2DNUM,DB2ENUM,DB2ENAME,DB2ONUM,DB2COMM)
        OPTIONS (OUT(100000),PRI(150),FLDERR(HALT),DEBUG(2))
        SELECT DNUM,ENUM,ENAME,ONUM,COMM
        FROM SALEVIEW
        WHERE DNUM IN ('15','20','38')
         AND ESALARY >= 25000;
```

Example 2

In this example, you want to:

- Insert your extracted data into a DB2 table named DB2TABLE, in the columns named DB2DNUM, DB2ENUM, DB2ENAME, DB2ONUM, and DB2COMM.
- Allow the OPTIONS keywords (except FDLMSG) to equal their default values. For the FLDMSG keyword, if more than 10 field format errors are diagnosed, the succeeding errors will not be diagnosed.
- Extract data from fields contained in the DXTVIEW named SALEVIEW (you have been authorized to access SALEVIEW).
- Extract data from the fields contained in SALEVIEW named DNUM, ENUM, ENAME, ONUM, and COMM.
- Extract data about all departments except 15, 20, and 38, and only about employees having salaries between $25,000 and $28,000 or salaries between $40,000 and $45,000.

```
EXTRACT INTO DB2TABLE
(DB2DNUM,DB2ENUM,DB2ENAME,DB2ONUM,DB2COMM)
        OPTIONS (FLDMSG(10))
        SELECT DNUM,ENUM,ENAME,ONUM,COMM
        FROM SALEVIEW
        WHERE DNUM NOT IN ('15','20','38')
          AND (ESALARY BETWEEN 25000 AND 28000
          OR ESALARY BETWEEN 40000 AND 45000);
```

Example 3

In this example, you want to:

- Load your extracted data into a physical sequential data set rather than into a relational database table.
- Allow the OPTIONS keywords to equal the default values.
- Extract data from fields contained in the DXTVIEW named SALEVIEW (you have been authorized to access SALEVIEW).
- Specifically extract data from the fields contained in SALEVIEW named DNUM, ENUM, ENAME, ONUM, and COMM.
- Extract data only about employees whose names start with an S.

```
EXTRACT SELECT DNUM,ENUM,ENAME,ONUM,COMM
        FROM SALEVIEW
        WHERE ENAME LIKE 'S%';
```

Example 4

In this example, you want to:

- Insert your extracted data into a DB2 table named DB2TABLE, in the columns named DB2DNUM, DB2ENUM, DB2ENAME, DB2JOB, DB2YEARS, DB2SALARY, DB2BONUS, DB2ONUM, DB2COMM, DB2DATE, DB2HJOB, DB2HYEARS, DB2SSAL, and DB2ESAL.
- Allow the OPTIONS keywords to equal the default values.
- Extract data from fields contained in the DXTVIEWs named SALEVIEW and HISTVIEW (you have been authorized to access both).
- Extract data from the fields contained in SALEVIEW and HISTVIEW named DNUM, ENUM, ENAME, JOB, YEARS, SALARY, BONUS, ONUM, COMM, DATE, HJOB, HYEARS, SSAL, and ESAL.
- Extract data about all employees except those whose names start with an S.

```
EXTRACT INTO DB2TABLE (DB2DNUM,DB2ENUM,DB2ENAME,DB2JOB,
                       DB2YEARS,DB2SALARY,DB2BONUS,
                       DB2ONUM,DB2COMM,DB2DATE,DB2HJOB,
                       DB2HYEARS,DB2SSAL,DB2ESAL)
        SELECT DNUM,ENUM,ENAME,JOB,YEARS,SALARY,BONUS,
               ONUM,COMM,DATE,HJOB,HYEARS,SSAL,ESAL
        FROM SALEVIEW,HISTVIEW
        WHERE ENAME NOT LIKE 'S%';
```

Example 5

In this example, you want to:

- Create a new DB2 table named DB2TAB with columns named DB2ENAME, DB2SALARY, DB2BONUS, DB2ONUM, DB2COMM, DB2DATE, DB2HJOB, DB2HYEARS, DB2SSAL, DB2ESAL.
- Allow the OPTIONS keywords to equal the default values.
- Extract data from fields contained in the DXTVIEW HISTVIEW named ENAME, SALARY, BONUS, ONUM, COMM, DATE, HJOB HYEARS, SSAL, ESAL (you have been authorized to access HISTVIEW).
- Extract data from all of the fields contained in the DXTVIEW named HISTVIEW.
- Order the data according to salary, from lowest salaried employee to the highest.

```
EXTRACT WITH CREATE
        INTO DB2TAB (DB2ENAME,DB2SALARY,DB2BONUS,
                     DB2ONUM,DB2COMM,DB2DATE,DB2HJOB,
                     DB2HYEARS,DB2SSAL,DB2ESAL)
        SELECT ENAME,SALARY,BONUS,ONUM,COMM,
               DATE,HJOB,HYEARS,SSAL,ESAL
        FROM HISTVIEW
        ORDER BY SALARY;
```

Example 6

In this example, you want to:

- Replace all of the data in an existing SQL/DS table named SQLTABLE, in the columns named SQLDNUM, SQLDAREA, SQLDMGR, SQLDDIV, SQLDCITY, SQLENUM, SQLENAME, SQLJOB, SQLYEARS, SQLSALARY, SQLBONUS, SQLDATE, SQLHJOB, SQLHYEARS, SQLSSAL, and SQLESAL.
- Allow the OPTIONS keywords to equal the default values.
- Extract data from fields contained in the DXTVIEW named HISTVIEW (you have been authorized to access HISTVIEW).
- Extract data from all of the fields contained in the DXTVIEW named HISTVIEW.

```
EXTRACT WITH REPLACE
        INTO SQLTABLE (SQLDNUM,SQLDAREA,SQLDMGR,SQLDDIV,
                       SQLDCITY,SQLENUM,SQLENAME,SQLJOB,
                       SQLYEARS,SQLSALARY,SQLBONUS,
                       SQLDATE,SQLHJOB,SQLHYEARS,SQLSSAL,SQLESAL)
        SELECT *
        FROM HISTVIEW;
```

Example 7

In this example, you want to:

- Add your extracted data to an existing SQL/DS table named SQLTABLE, in the columns named SQLDNUM, SQLDAREA, SQLDMGR, SQLDDIV, SQLDCITY, SQLENUM, SQLENAME, SQLJOB, SQLYEARS, SQLSALARY, SQLBONUS, SQLDATE, SQLHJOB, SQLHYEARS, SQLSSAL, and SQLESAL.
- Allow the OPTIONS keywords to equal the default values.
- Extract data from fields contained in the DXTVIEW named HISTVIEW (you have been authorized to access HISTVIEW).
- Extract data from all of the fields contained in the DXTVIEW named HISTVIEW.

```
EXTRACT INTO SQLTABLE (SQLDNUM,SQLDAREA,SQLDMGR,SQLDDIV,
                       SQLDCITY,SQLENUM,SQLENAME,SQLJOB,
                       SQLYEARS,SQLSALARY,SQLBONUS,
                       SQLDATE,SQLHJOB,SQLHYEARS,SQLSSAL,SQLESAL)
        SELECT *
        FROM HISTVIEW;
```

Example 8

In this example, you want to identify the second line managers of the employees in your company who are nearing their 25th service anniversary.

Information about your employees is contained in the following columns of a DB2 table called EMP99 that you created:

- EMPNAME—employee name
- EMPNUM—employee number
- EMPYRS—number of years with company
- EMPDEPT—employee department number
- EMPMGRNO—employee number of employee's manager

Information about the managers is contained in an IMS/VS DL/I hierarchy over which a DXTVIEW named HISTVIEW is defined.

- You will use a GDI record exit that can read rows from EMP99 and return them to the DEM. (Your GDI record exit interprets a table as a file.)
- You must first create a DXTFILE description of the DB2 table EMP99, which you name EMP99.

- You then create a DXTVIEW description of this DXTFILE (EMP99) which you name VEMP99.

You want to:

- Reduce coding by using the default values for the OPTIONS keywords
- Join the EMP99 table with HISTVIEW
- Load your extracted data into another DB2 table named SECONDLN.

```
EXTRACT INTO SECONDLN (ENAME, ENBR, EDEPT, M2NBR)
        SELECT VEMP99.EMPNAME, VEMP99.EMPNBR, VEMP99.EMPDEPT,
               HISTVIEW.DMGR
        FROM VEMP99, HISTVIEW
        WHERE VEMP99.EMPYRS = 24
          AND HISTVIEW.ENUM = VEMP99.EMPMGRNO;
```

DataRefresher extracts data about the employees and the employee number for the employee's manager from EMP99. For employees with 24 years service with the company, DataRefresher extracts data from HISTVIEW about the employee's manager and the manager of that manager.

Chapter 6. Extracting data using SAP commands

The Structures Access Program (SAP) uses user-specified data structures to generate DataRefresher data description statements, and extract requests based on stored information about IMS database definitions (DBDs), VSAM files, and physical sequential files.

For IMS database definitions, the SAP generates:

- `CREATE DXTPSB` statement
- `CREATE DXTVIEW` statement
- `SUBMIT/EXTRACT` statement

This chapter describes the tasks that the DataRefresher administrator must perform to create the data description statements and an extract request. It also describes the SAP conventions and restrictions.

Conventions and restrictions

This section contains information about the rules and limitations of using the SAP with COBOL and PL/I structures.

Conventions for SAP data structure conversion

The following table describes the conventions the SAP observes in converting data structures to DataRefresher `CREATE` and `SUBMIT/EXTRACT` statements:

If a *picture field* contains	• Edit characters (for example, Z), the field is treated as a *character field* • Only the P, S, V, and 9 symbols, the field is treated as a *numeric field*
If any attributes are	• Not provided for a field; the SAP uses the standard default rules for PL/I and COBOL • Not supported (such as LIKE or any dynamic sizes); they may cause error conditions
In *PL/I*	• The data structure must be defined between columns 2 and 72 of the PL/I declare library.
In *COBOL*	• The data definitions must fall within standard COBOL input areas. • A hyphen (-) in COBOL data names is changed to an underscore (_).

Restrictions to SAP processing

To generate data description statements and an extract request for your DL/I database, VSAM file, or sequential file, the SAP has to process the data definitions using PL/I or COBOL structures. If you are generating statements for a DL/I database, the SAP also needs information from an IMS DBD.

The following information outlines the restrictions as they apply to these libraries and DBDs:

General restrictions:

1. The PL/I and COBOL structures must be able to be compiled successfully as individual objects.
2. The IMS DBD generation program must be able to process the database definitions.
3. Any data type not included as either PL/I or COBOL is ignored, but its length is accounted for.
4. If either character or picture data exceeds the maximum lengths, the SAP ignores that field; however, the SAP calculates the length and alignment to position later items correctly.
5. If any field names are duplicated, the first occurrence of the name is used as provided. Each occurrence of the name after the first is altered in the output to make the name of the field a unique name.
6. If the COBOL or PL/I structure defines one or more fields that overlay the key field defined by the IMS DBD, the SAP generates both the DBD key field and the structure's field names.

IMS:

1. The SAP supports logical DBDs. The structure name must match the name on the DBD. For a logical DBD, this structure must represent the logical view of the data returned by DL/I.
2. Secondary indexes are not supported.
3. The segment names in the DBD must match the member names in a COBOL or PL/I structure partitioned data set.
4. The SAP supports bidirectional extracts.

PL/I structures:

1. The PL/I compiler must be able to process the PL/I structures. The PL/I structures must begin with a `DECLARE` statement and end with a semicolon (;).
2. DCL (DECLARE) members can include other members using the `%INCLUDE` statement only if they exist in the same data set.
3. The SAP analyzes only the first DECLARE (at the beginning of the data structure). The SAP ignores all succeeding DCLs.
4. SAP use the default PL/I rules for calculating field offsets. Take care when trying to align data types to byte and word boundaries.
5. `SKIP`, `PAGE`, `PRINT`, and `NOPRINT` statements are permitted, but they are ignored.
6. The `LIKE` attribute is not supported.

7. Precompiler functions are not supported. If they are used, they either cause an incorrect analysis or a serious problem is reported.
8. MARGINS are assumed to be 2 and 72.
9. Standard defaults are assumed. The `DEFAULT` statement is not permitted.
10. Only the following data types are explicitly supported:

Data type	Restrictions
CHARACTER(n)	n ≤ 254
CHARACTER(n) varying	n ≤ 32767
GRAPHIC(n)	n ≤ 127
GRAPHIC(n) varying	n ≤ 16383
BINARY FLOAT(n)	22 ≤ n ≤ 53
DECIMAL FLOAT(n)*	n ≤ 16
BINARY FIXED(a,b)	16 ≤ a ≤ 31, b = 0
DECIMAL FIXED(a,b)	1 ≤ a ≤ 15, b ≤ 15
PICTURE	Total length ≤ 254
REAL numeric items	None

* SAP will accept a DECIMAL FLOAT value without *n*, by defaulting to a half-word (four byte) field.

COBOL structures:

1. The COBOL compiler must be able to compile the COBOL structures as an object. The COBOL structures must begin with a 01 level and represent one database segment.
2. Elements that follow a COBOL `OCCURS DEPENDING ON` statement, but are not subordinate to the OCCURS clause, cannot have a SYNCHRONIZE (ALIGN) clause.
3. The USAGE INDEX and the USAGE POINTER are not supported. INDEX (POINTER) items are omitted from the output.
4. P picture items are ignored.
5. COMP-3 items are limited to 18 digits. COMP items are limited to 9 digits. Items longer than these limits are omitted from the output.
6. Leading signs are not supported for numeric items. Numeric items are changed to CHAR if a leading sign is found.
7. A nonzero scale (the use of the V picture) is omitted for COMP items.
8. The RENAMES (level 66) option is not supported. Processing ends if one is found.
9. *CBL, *CONTROL, SKIP, and EJECT are permitted but ignored.
10. "D" debug lines are always treated as comments.
11. Only the following data items are explicitly supported:

Data type	Restrictions
DISPLAY	Length ≤254
DISPLAY-1	Length ≤ 127
COMP	Length ≤ 9 (digit positions)
COMP-1	none
COMP-2	none
COMP-3	Length < 18 (digit positions)

Specifying SAP execution information

Before you can specify SAP execution information, you must determine which data definition describes the source data for the DataRefresher extract and whether the data definition is in the IMS DBD libraries or a data set containing COBOL or PL/I language structures.

Starting SAP

Two DataRefresher JCL skeletons can be used to start SAP:

DVRXKJFA Invokes the SAP Command Generator to generate DataRefresher `CREATE DXTFILE` statements from COBOL or PL/I language structures.

DVRXKJDA Invokes the SAP Command Generator to generate DataRefresher `CREATE DXTPSB` statements from IMS DBD elements and the underlying COBOL or PL/I structures.

SAP generates accompanying CREATE DXTVIEW and SUBMIT/EXTRACT commands using either JCL skeleton. Both skeletons are stored in the DataRefresher Dialogs skeleton library and should be tailored for your specific installation during installation. For information about tailoring these JCL skeletons, see the DataRefresher Program Directories.

You must specify the information about the IMS DBD elements and data structures that you are using in the skeletons. To incorporate the specifications, you can use:

- SAP Dialogs panels
- An editor to modify a copy of the skeleton JCL

These methods are described in the following sections.

Using SAP Dialog panels

You can start the SAP Dialogs from:

- DataRefresher Administrative Dialogs
- ISPF Main Menu (if an option has been set up by the system administrator).

This section takes you through SAP Dialogs panels using the DataRefresher Administrative Dialogs. The sample data shown in this section, is provided with DataRefresher.

Step 1. Select option 2 DESCRIPTION from the Administrative Dialogs Main Menu.

The Build and Maintain Data Description Requests panel is displayed.

Step 2. Select option 5, from the Build and Maintain Data Description Request# panel, to start SAP Dialogs.

The Create DataRefresher Design panel, shown in Figure 1 on page 127, is displayed.

```
                                CREATE DataRefresher DESIGN
COMMAND ===>


      Select the type of data
      description to be built.        ===> 1
            1  FILE
            2  PSB
```

Figure 1. Create DataRefresher design panel

Step 3. From the Create DataRefresher Design panel, select one of the following options:

1 FILE Generates the table design for a VSAM or sequential file (a `CREATE DXTFILE` statement). When you select this option the Create DXTFILE for VSAM or SEQ File panel, shown in Figure 2, is displayed.

2 PSB Generate the table design for an IMS DL/I database (a `CREATE DXTPSB` statement). When you select this option the Create DXTPSB for DL/I Database panel, shown in Figure 3 on page 128, is displayed.

Step 4. The action taken in this step depends on the option selected in Step 3.

- If you selected option **1**, type the following information, shown in Figure 2 and press ENTER:
  - Copy library name
  - Name of member containing the structures
  - Type of source file
  - Language to be accessed

```
                    CREATE DXTFILE FOR VSAM OR SEQ FILE
 Command ===>

 DECLARE COPY LIBRARY:
    PROJECT  ===> DXTDEV
    GROUP    ===> DVR110
    TYPE     ===> DVRSAMPE
    MEMBER   ===> DVRECSDP
  OTHER PDS  ===>

DXTFILE TYPE    ===> SEQ          (ESDS, KSDS, or SEQ)

LANGUAGE        ===> PLI          (PLI or COBOL)

Do you want to validate the
selected Data Sets and Members?
Enter Yes or No                   ===> YES


Press:   HELP key for information   ENTER to continue   END key to return
```

Figure 2. Create DXTFILE for VSAM or sequential file panel

- If you specified option **2**, type the following information, shown in Figure 3 on page 128 and press ENTER:
  - IMS DBD source data set and member
  - Copy library name containing structures
  - Structure's language

```
                CREATE DXTPSB FOR DL/I DATA BASE
 Command ===>

 IMS DBD SOURCE:
     PROJECT  ===> DXTDEV
     GROUP    ===> DVR110
     TYPE     ===> DVRSAMPE
     MEMBER   ===> DVRECSDD
  OTHER PDS   ===>

 DECLARE COPY LIBRARY:
     PROJECT  ===> DXTDEV
     GROUP    ===> DVR110
     TYPE     ===> DVRSAMPE
  OTHER PDS   ===>

 LANGUAGE        ===> COBOL      (PLI or COBOL)

 Do you want to validate the
 selected Data Sets and Members?
 Enter Yes or No                 ===> YES

Press:   HELP key for information   ENTER to continue   END key to return
```

Figure 3. Create DXTPSB for DL/I database panel

The Create Commands Target Library panel, shown in Figure 4, is displayed. You use this panel to identify the library in which you want the generated CREATE DXTFILE or DXTPSB commands and the SUBMIT/EXTRACT command to be stored and whether you want to replace the existing PDS members.

```
                  CREATE COMMANDS TARGET LIBRARY
 Command ===>


 DXT CREATE COMMAND TARGET LIBRARY:
    PROJECT  ===>
    GROUP    ===>
    TYPE     ===>
    MEMBER   ===>              (CREATE DXTFILE or DXTPSB COMMANDS)
  OTHER PDS  ===>

(Optional - Target Library Member for the CREATE SUBMIT/EXTRACT Request)
    MEMBER    ===>

 REPLACE MEMBERS ===> YES      (YES | NO  Replace members if they exist)


Press:   HELP key for information   ENTER to continue   END key to return
```

Figure 4. Create commands target library panel

Step 5. Type the following information in the Create Commands Target Library panel, and press ENTER:

- DataRefresher Create Command Target Library information:

  To use the sample data supplied with DataRefresher make the following entries depending on the option selected in Step 3 on page 127:

Field	Option 1	Option 2
Project	DXTDEV	DXTDEV
Group	DVR110	DVR110
Type	DVRIMEXE	DVRIMEXE
Member	DVRECSDP	DVRECSDD

- Optional - Target Library Member field

  To use the sample data supplied with DataRefresher make the following entries depending on the option selected in Step 3 on page 127:

	Option 1	Option 2
Member	DVRECSBP	DVRECSUB

Note:

If you make an entry in this field the Submit/Extract and Table Load Details panel, shown in Figure 5, is displayed after you press ENTER.

If you do not make an entry for this field Submit or Save the Generated JCL panel is displayed, and you should continue from Step 7 on page 130.

- Specify whether existing members are to be replaced.

Step 6. If you made an entry for the optional target library member in Step 5 on page 128, the Submit/Extract and Table load Details panel, shown in Figure 5, is displayed.

```
            SUBMIT/EXTRACT and TABLE LOAD Details
Command ===>

   TARGET SYSTEM TYPE          ===>          (DB2,SQLDS, IXF or FILE)
   TARGET NODE ID              ===>
   EXTDATA DDNAME              ===>          (Blank for DB2 or SQL/DS)
   OUTPUT FORMAT               ===>          (EBCDIC or SOURCE)

   REPLACE DB2/SQLDS TABLE     ===>          (YES or NO)
   DB2 OR SQL/DS USER ID       ===>
   DB2 OR SQL/DS TABLE NAME    ===>
   DB2 OR SQL/DS JCS NAME      ===>

Press:   HELP key for information   ENTER to continue   END key to return
```

Figure 5. Submit/Extract and table load details panel

Use the panel to specify the target table and its location and characteristics. If you are using the DataRefresher example make the following entries for this panel:

Field	Entry
Target System Type	DB2
Target Node ID	*nodename*
Output Format	Source
Replace DB2/SQLDS Table	Yes
DB2 or SQL/DS User ID	*userid*
DB2 or SQL/DS Table Name	DB2TABLE
DB2 or SQL/DS JCS Name	DB2DDNAME

The Submit or Save the Generated JCL panel, shown in Figure 6 on page 130, is displayed.

```
            SUBMIT or SAVE the Generated JCL
Command ===>


Do you want to CREATE the DataRefresher commands now
or save the JCL for later execution.          ===> _
     1  Execute the JCL now
     2  Save the JCL for later execution

SAVE JCL ISPF LIBRARY:

   PROJECT  ===>
   GROUP    ===>
   TYPE     ===>
   MEMBER   ===>
 OTHER PDS  ===>

REPLACE MEMBERS ===> YES      (REPLACE EXISTING MEMBER - YES OR NO)

Press:   HELP key for information   ENTER to continue   END key to return
```

Figure 6. Submit or save generated JCL panel

Step 7. The entry you make for this panel depends on whether the job is to be run immediately:

Option 1 Submits the generated JCL to SAP Command Generator.

Option 2 Saves the generated JCL. Specify the library and member where you want to store the JCL and whether you want to replace an existing member.

If you are using the DataRefresher example make the following entries for this panel:

Field	Entry
Project	DXTDEV
Group	DVR110
Type	DVRIMEXE
Member	DVRECSJL
Replace Members	YES

Editing the skeleton JCL

The two JCL skeletons (DVRXKJFA for DataRefresher file statements and DVRXKJDA for DataRefresher PSB statements) are stored in the DataRefresher Dialogs skeleton library.

Note: They skeletons can not be accessed from the DataRefresher Dialogs. You must access them from outside DataRefresher to copy them and edit the copies.

You can use SAP Dialogs, shown in "Using SAP Dialog panels" on page 126 to generate a copy of the JCL containing the information you need to invoke SAP. If you save this JCL, shown in Step 7, you can edit and resubmit the JCL to the SAP to generate data descriptions and extract requests based on other structures.

JCL for generating a CREATE DXTFILE statement:

```
//&ZUSER.# JOB (,,BINR1),'&ZUSER',USER=&ZUSER.,CLASS=A,
//            MSGLEVEL=(1,1),MSGCLASS=X
/*NOTIFY ***NODEID***.&ZUSER
/*ROUTE PRINT ***NODEID***.&ZUSER
//*************************************************************
//******DVRXKJFA*************************************************
//* THE DVRMLIBE  MESSAGE LIBRARY CONTAINS THE SAP MESSAGES USED
//* in the Dialogs and SAP messages used for the sysprint
//* output.
//*
//*           THOSE STATEMENTS WHICH MUST BE MODIFIED ARE
//*           FLAGGED BY ENCLOSING THE PART TO BE MODIFIED
//*           IN TRIPLE  ASTERISKS. FOR EXAMPLE:
//*
//*               DSN=***DXTPRE***
//*
//* THE STEPLIB SHOULD INCLUDE THE FOLLOWING LIBRARIES:
//*  1. PLILINK
//*  2. SIBMLINK
//*
//*************************************************************
//* DVRFL   BUILD DataRefresher COMMANDS FROM A SEQ./VSAM FILE
//*************************************************************
//*
//DVRFL    PROC    1
//*
//ALLOC    EXEC PGM=IEFBR14
//TEMPTAB  DD DSN=&ZUSER..TEMPTAB,DISP=(NEW,PASS),
//            UNIT=SYSDA,SPACE=(TRK,(2,1,2)),
//            DCB=***ISPPRE***.ISPTLIB
//*
//STEP01   EXEC PGM=IKJEFT01,DYNAMNBR=25,REGION=1024K,(0,LT,ALLOC)
//STEPLIB  DD DSN=***PLI.RUNTIME.LIB***.SIBMLINK,DISP=SHR
//         DD DSN=***PLI.RUNTIME.LIB***.PLILINK,DISP=SHR
//SYSIN    DD DUMMY
//SYSTSIN  DD DDNAME=INTSO    2
//VARSIN   DD DDNAME=INVARS   3
//ISPLLIB  DD DSN=***DXTPRE***.DVRSLOAD,DISP=SHR
//         DD DSN=***DXTPRE***.DVRLOAD,DISP=SHR
//ISPMLIB  DD DSN=***DXTPRE***.DVRMLIBE,DISP=SHR
//         DD DSN=***ISPPRE***.ISPMLIB,DISP=SHR
//         DD DSN=***ISPPRE***.ISRMLIB,DISP=SHR
//ISPPLIB  DD DSN=***DXTPRE***.DVRPLIB&LANG,DISP=SHR
//ISPSLIB  DD DSN=***DXTPRE***.DVRSLIBE,DISP=SHR
//ISPTLIB  DD DSN=&ZUSER..TEMPTAB,DISP=SHR
//         DD DSN=***ISPPRE***.ISPTLIB,DISP=SHR
//DVRXOTB  DD DSN=&ZUSER..TEMPTAB,DISP=SHR
//SRCLIB   DD DSN=&SRCLIB,    4
//            DISP=SHR
//ERDREP1  DD SYSOUT=*
//ISPPROF  DD UNIT=SYSDA,SPACE=(TRK,(1,1,1)),
//            DCB=***ISPPRE***.ISPTLIB
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//ISPLOG   DD SYSOUT=*,DCB=(RECFM=VA,LRECL=125,BLKSIZE=129)
//PLIDUMP  DD SYSOUT=*
//DVRFL    PEND
//*
```

Figure 7 (Part 1 of 2). JCL for generating a CREATE DXTFILE statement

```
//ISPFGO   EXEC DVRFL
//STEP01.INTSO  DD  *
       ISPSTART PGM(DVRJ0000) PARM('/')
//STEP01.INVARS DD *
  SRCTYPE = '&SRCTYPE';
  DES2DSN = '&FTDSN';
  DES2MEM = '&FTMEM';
  DES2LAN = '&DES2LAN';
  DES2FTY = '&DES2FTY';
  DES2REP = '&DES2REP';
  LDJCLDN1 = '&FTLDSN';
  LDJCLMEM = '&FTLMEM';
  LDJCLMEX = '&LDJCLMEX';
  LDJCLSTP = '&LDJCLSTP';
  LDNODEID = '&LDNODEID';
  LDJCLDDN = '&LDJCLDDN';
  LDFORMAT = '&LDFORMAT';
  LDTREP   = '&LDTREP';
  LDUSERID = '&LDUSERID';
  LDOUTTBN = '&LDOUTTBN';
```

Figure 7 (Part 2 of 2). JCL for generating a CREATE DXTFILE statement

1 The DVRFL procedure sets up the environment required by SAP. The PROC is executed by the `IPFGO EXEC DVRFL` command.

2 The INTSO inline data set invokes SAP Command Generator.

3 The INVARS inline data set defines the parameters that the SAP uses to generate data description statements and an extract request.

4 The `&SRCLIB` variable identifies the structure library. The user assigns this variable a value either by using the dialogs or by editing the file. This variable should have the same value as DES2DSN.

The definitions of the parameters in the JCL model required to generate a `CREATE DXTFILE` statement are:

SRCTYPE	SRCTYPE=1 for structure (non-DBD) data
DES2DSN	Segment structure data set name (copy lib)
DES2MEM	Structure member name
DES2LAN	Source language (PLI or COBOL)
DES2FTY	Source file type (ESDS, KSDS, or SEQ)
DES2REP	Replace the target data set? (Yes or No)
LDJCLDN1	Output target data set name
LDJCLMEM	Output target member for `DXTPSB/DXTVIEW` commands
LDJCLMEX	Output target member for SUBMIT/EXTRACT data
LDJCLSTP	Output target type (DB2, SQLDS, IXF, or FILE)
LDNODEID	SUBMIT node ID
LDJCLDDN	SUBMIT DD name
LDFORMAT	SUBMIT output format (EBCDIC or SOURCE)
LDREP	Replace target relational table? (Yes or No)
LDUSERID	Create relational table user ID
LDOUTTBN	Create relational table name

If you followed the example in "Using SAP Dialog panels" on page 126 to generate JCL for generating DataRefresher file data descriptions, the values of SAP input in the saved JCL are:

```
//STEP.INVARS DD *
SRCTYPE = '1';
  DES2DSN = '''DXTDEV.DVR110.DVRSAMPE''';
  DES2MEM = 'DVRECSDP';
  DES2LAN = 'PLI';
  DES2FTY = 'SEQ';
  DES2REP = 'YES';
  LDJCLDN1 = '''DXTDEV.DVR110.DVRIMEXE''';
  LDJCLMEM = 'DVRECSDP';
  LDJCLMEX = 'DVRECSBP';
  LDJCLSTP = 'DB2';
  LDJCLDDN = 'DB2DDNAM';
  LDNOTEID = 'nodeid';
  LDFORMAT = 'SOURCE';
  LDTREP   = 'YES';
  LDUSERID = 'userid';
  LDOUTTBN = 'DB2TABLE';
```

JCL for generating a CREATE DXTPSB statement:

```
//&ZUSER.# JOB (,,BINR1),'&ZUSER',USER=&ZUSER.,CLASS=A,
//             MSGLEVEL=(1,1),MSGCLASS=X
/*NOTIFY ***NODEID***.&ZUSER
/*ROUTE PRINT ***NODEID***.&ZUSER
//*****************************************************************
//******DVRXKJDA***************************************************
//* THE DVRMLIBE MESSAGE LIBRARY CONTAINS THE SAP MESSAGES USED
//* in the Dialogs and SAP messages used for the sysprint
//* output.
//*
//*           THOSE STATEMENTS WHICH MUST BE MODIFIED ARE
//*           FLAGGED BY ENCLOSING THE PART TO BE MODIFIED
//*           IN TRIPLE  ASTERISKS. FOR EXAMPLE:
//*
//*               DSN=***DXTPRE***
//*
//* THE STEPLIB SHOULD INCLUDE THE FOLLOWING LIBRARIES:
//*  1. PLILINK
//*  2. SIBMLINK
//*
//*****************************************************************
//* DVRDBD   BUILD DataRefresher COMMANDS FROM AN IMS/VS DBD.
//*****************************************************************
//*
//DVRDBD   PROC    1
//*
//ALLOC    EXEC PGM=IEFBR14
//TEMPTAB  DD DSN=&ZUSER..TEMPTAB,DISP=(NEW,PASS),
//            UNIT=SYSDA,SPACE=(TRK,(2,1,2)),
//            DCB=***ISPPRE***.ISPTLIB
//*
//STEP01   EXEC PGM=IKJEFT01,DYNAMNBR=25,REGION=1024K,COND=(0,LT,ALLOC)
//STEPLIB  DD DSN=***PLI.RUNTIME.LIB***.SIBMLINK,DISP=SHR
//         DD DSN=***PLI.RUNTIME.LIB***.PLILINK,DISP=SHR
//SYSIN    DD DUMMY
//SYSTSIN  DD DDNAME=INTSO    2
//VARSIN   DD DDNAME=INVARS   3
//ISPLLIB  DD DSN=***DXTPRE***.DVRSLOAD,DISP=SHR
//         DD DSN=***DXTPRE***.DVRLOAD,DISP=SHR
//ISPMLIB  DD DSN=***DXTPRE***.DVRMLIBE,DISP=SHR
//         DD DSN=***ISPPRE***.ISPMLIB,DISP=SHR
//         DD DSN=***ISPPRE***.ISRMLIB,DISP=SHR
//ISPPLIB  DD DSN=***DXTPRE***.DVRPLIB&LANG,DISP=SHR
//ISPSLIB  DD DSN=***DXTPRE***.DVRSLIBE,DISP=SHR
//ISPTLIB  DD DSN=&ZUSER..TEMPTAB,DISP=SHR
//         DD DSN=***ISPPRE***.ISPTLIB,DISP=SHR
//DVRXOTB  DD DSN=&ZUSER..TEMPTAB,DISP=SHR
//DBDSRC   DD DSN=&DBDSRC,    4
//            DISP=SHR
//SRCLIB   DD DSN=&SRCLIB,    5
//            DISP=SHR
//ERDREP1  DD SYSOUT=*
//ISPPROF  DD UNIT=SYSDA,SPACE=(TRK,(1,1,1)),
//            DCB=***ISPPRE***.ISPTLIB
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//ISPLOG   DD SYSOUT=*,DCB=(RECFM=VA,LRECL=125,BLKSIZE=129)
//PLIDUMP  DD SYSOUT=*
//DVRDBD   PEND
//*
```

Figure 8 (Part 1 of 2). JCL for generating a CREATE DXTPSB statement

```
//ISPFGO   EXEC DVRDBD
//STEP01.INTSO DD  *
       ISPSTART PGM(DVRJ0000) PARM('/')
//STEP01.INVARS DD *
  SRCTYPE = '&SRCTYPE';
  DES1MEM = '&FTMEMD1';
  DES1DN1 = '&FTDSND1';
  DES1LAN = '&DES1LAN';
  DES1DN2 = '&FTDSND2';
  LDREPMEM = '&LDREPMEM';
  LDJCLDN1 = '&FTLDSN';
  LDJCLMEM = '&FTLMEM';
  LDJCLMEX = '&LDJCLMEX';
  LDJCLSTP = '&LDJCLSTP';
  LDJCLDDN = '&LDJCLDDN';
  LDNODEID = '&LDNODEID';
  LDFORMAT = '&LDFORMAT';
  LDTREP   = '&LDTREP';
  LDUSERID = '&LDUSERID';
  LDOUTTBN = '&LDOUTTBN';
```

Figure 8 (Part 2 of 2). JCL for generating a CREATE DXTPSB statement

1 The DVRDBD procedure sets up the environment required by SAP. The PROC is executed by the `IPFGO EXEC DVRDBD` command.

2 The INTSO inline data set invokes the SAP Command Generator.

3 The INVARS inline data set defines the parameters that SAP uses to generate data description statements and an extract request.

4 The `&DBDSRC` variable identifies the DBD library. The user assigns this variable a value either by using the dialogs or by editing the file.

This variable should have the same value as DES1DN1.

5 The `&SRCLIB` variable identifies the structure library. The user assigns this variable a value either by using the dialogs or by editing the file. This variable should have the same value as DES1DN2.

The definitions for the parameters in the JCL model for the CREATE DXTPSB are:

SRCTYPE	SRCTYPE=2 for DBD data
DES1MEM	IMS DBD source member name
DES1DN1	IMS DBD source data set name
DES1LAN	Source language (PLI or COBOL)
DES1DN2	IMS segment structure data set name (Copy Lib)
LDREPMEM	Replace output target data set? (Yes or No)
LDJCLDN1	Output target data set name
LDJCLMEM	Output target member for `DXTPSB/DXTVIEW` commands
LDJCLMEX	Output target member for SUBMIT/EXTRACT data
LDJCLSTP	Output target type (DB2, SQLDS, IXF, or FILE)
LDJCLDDN	SUBMIT DD name
LDNODEID	SUBMIT node ID
LDFORMAT	SUBMIT output format (EBCDIC or SOURCE)
LDTREP	Replace relational table? (Y or N)
LDUSERID	Create relational table user ID
LDOUTTBN	Create relational table name

If you have followed the example in "Using SAP Dialog panels" on page 126 and saved the generated JCL, SAP input values in the saved JCL are:

```
//STEP01.INVARS DD *
       SRCTYPE = '2';
       DES1MEM = 'DVRECSDD';
       DES1DN1 = '''DXTDEV.DVR110.DVRSAMPE''';
       DES1LAN = 'COBOL';
       DES1DN2 = '''DXTDEV.DVR110.DVRSAMPE''';
       LDREPMEM = 'YES';
       LDJCLDN1 = '''DXTDEV.DVR110.DVRIMEXE''';
       LDJCLMEM = 'DVRECSDD';
       LDJCLMEX = 'DVRECSUB';
       LDJCLSTP = 'DB2';
       LDJCLDDN = 'DB2DDNAM';
       LDNODEID = 'NODEID';
       LDFORMAT = 'SOURCE';
       LDTREP   = 'Y';
       LDUSERID = 'USERID';
       LDOUTTBN = 'DB2TABLE';
```

Submitting the JCL for batch processing

When you have generated or edited the JCL for invoking SAP, submitting the finished JCL in one of the following ways:

- In SAP dialogs, select option 1 on the Submit or Save the Generated JCL panel (see Step 7 on page 130).
- While looking at the JCL with an editor, enter the `SUBMIT` command.

Using SAP output

SAP generates three types of DataRefresher data description statements:

- `CREATE DXTFILE` (for VSAM or sequential files)
- `CREATE DXTPSB` (for IMS DL/I databases)
- `CREATE DXTVIEW`

SAP can optionally generate an extract request for the data source described by the input structure.

Note: Always review SAP output before you use it. You can use the error or warning messages that appear in the SYSPRINT output (the Command Generator report) to correct any deviations in the output data descriptions.

For any type of output member (data description or extract request statement), you can:

1. Edit the partitioned data set member containing the commands.
2. Invoke the UIM to store the commands in the proper library (FDTLIB or EXTLIB).
3. Import the commands into the DataRefresher Dialogs library.

Editing SAP output

You can edit the data description statements or the extract request with any editor that can edit partitioned data set members.

Before you submit the data descriptions or extract request to the UIM, check the output from SAP for:

- Unwanted information

  SAP creates the UIM commands for your designated files and PSBs using all the DBD and language structures information available. As a result, the output may describe more segments, fields, or PCBs than you need for your extract requests.

- Missing information

  Information may be missing from the DBD or language structure because it is not normally carried in these structures. An example is the DDNAME for a file. SAP specifies in the SYSPRINT output (SAP Command Generator report) that information is missing from the structure.

- Erroneous information

  Information from SAP could be erroneous because the structure was created or maintained incorrectly, or includes nonstandard representation of data.

- Information in the wrong format

  When SAP generates DataRefresher data descriptions for segments that include arrays, SAP generates a separate path for each element in the array. In this type of data description, the parent segment is repeated for each element in the array, with only the array element changing for each path. If you want the data description to contain one path containing all occurrences in the array, you must edit the generated data description statement.

Invoking UIM to store SAP output

To store the data descriptions in the FDTLIB, specify the partitioned data set members containing the data descriptions as the DXTIN data set in the JCL used to invoke the UIM.

To store the extract request in the EXTLIB, specify the partitioned data set members containing the extract request as the DXTIN data set in the JCL used to invoke the UIM.

Importing SAP output into the DataRefresher dialog library

You can import the data descriptions and extract request into the DataRefresher Administrative Dialogs environment as model data descriptions or shared objects.

For more information about importing objects, see the DataRefresher MVS and VM User's Guide.

Error handling

Error messages from the SAP Dialogs are displayed on the screen. If you want more information about the error message, press the HELP key. Error messages from the SAP Command Generator are written to ERDREP1. The SAP handles errors based on the following return code:

Severe errors (rc ≥ 12)
Abnormal termination with problem diagnosis information written to ERDREP1.

The following figure is an example ISPF transaction log message for an abnormal termination:

```
*** ISPF TRANSACTION LOG ***                          USERID: USERNAME  DATE: 94/03/14

START OF ISPF LOG - - -  - SESSION # 1 ------------------------------------------------
   TSO  - COMMAND  -  - ALLOC F(ISPFILE) DA('USERNAME.SAP.OUTPUT'
***** DIALOG ERROR ***** - DVRXE900 - SAP SEVERE ERROR
    - A SEVERE ERROR HAS OCCURRED DURING THE EXECUTION OF A SAP DIALOG.
    - INFORMATION DETAILING THE CAUSE OF THE ERROR IS LISTED BELOW.
    - THIS SAP DIALOG WILL BE TERMINATED WHEN YOU PRESS THE ENTER KEY.
    -
    - PLEASE TAKE THIS INFORMATION TO YOUR SAP SUPPORT PERSONNEL.
    -
    -        MODULE :
    -       SERVICE : FTINCL
    -          PARM : DVRXKJFK
    -   RETURN CODE :       20
    -
    - ISPF103
    - IMBED FILE 'LOSTSKEL' DOES NOT EXIST, DVRXKJFK RECORD-47
 END OF ISPF LOG - - - - - SESSION # 1 ------------------------------------------------
```

Other errors (4 < rc < 12)

The SAP Command Generator writes message information to ERDREP1, and the SAP Command Generator terminates processing.

Warning errors (0 < rc ≤ 4)

The SAP Command Generator writes message information to ERDREP1 and continues processing.

The following figure is an example of an exception report that would be written to ERDREP1.

```
          SAP COMMAND GENERATOR REPORT                        94/03/14
    FILE NAME   : COB01
    COPY LIBRARY: 'USERS.COBOL.TESTS'

    DVRXW126 - SCALING CHARACTER FOUND IN PIC SP.  SCALE IGNORED

    DVRXW126 - SCALING CHARACTER FOUND IN PIC SP.  SCALE IGNORED

    DVRXW126 - SCALING CHARACTER FOUND IN PIC SP.  SCALE IGNORED

    DVRXW120 - LEADING SIGN NOT SUPPORTED FOR NUMERIC ITEM ELEM_3.

    DVRXW126 - SCALING CHARACTER FOUND IN PIC SP.  SCALE IGNORED

    DVRXW126 - SCALING CHARACTER FOUND IN PIC SP.  SCALE IGNORED

                0 ERROR MESSAGES WERE PRINTED.
                6 WARNING MESSAGES WERE PRINTED.
                0 INFORMATION MESSAGES WERE PRINTED.
          *** END OF "SAP COMMAND GENERATOR REPORT" ***
```

For more information about SAP messages, refer to the *DataRefresher Messages and Codes*.

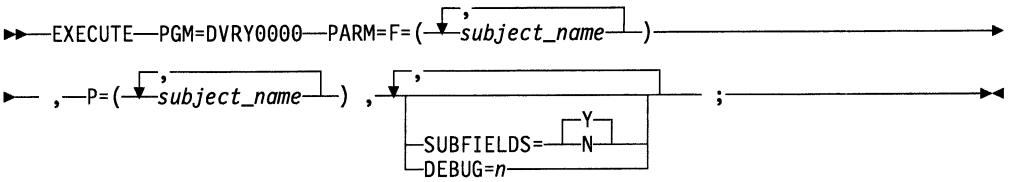
Chapter 7. DAP commands

This chapter contains an alphabetical listing of all Dictionary Access Program (DAP) commands. The DAP is a program for generating data descriptions of source files and non-relational databases from which data can be extracted. The DAP retrieves these descriptions from the IBM OS/VS DB/DC Data Dictionary.

For more information about using the DAP, see the *DataRefresher MVS and VM User's Guide.*

EXECUTE

Use this OS/VS DB/DC Data Dictionary command to start the DataRefresher DAP.



The rules for the EXECUTE command include the general syntax rules for Data Dictionary commands. For more information see the section on command conventions in the *IBM OS/VS DB/DC Data Dictionary Terminal User's Guide and Command Reference*.

The general rules are:

- Use only the first 79 columns (bytes) of a line for your command.
- Use the plus sign (+) to indicate statement continuation. The plus sign indicates where the continuation will start. If you need a blank between the end of your statement on one line and its continuation on the next, place that blank just before the plus sign, or in column 1 of the continuation record.
- Within the single quotation marks on the PARM keyword, write up to 200 characters, including blanks.

Rules peculiar to the DAP:

- Write the keywords in any order you like.
- Place a single comma between successive keywords.
- Write no more than one keyword of each type.
- Specify either the F or the P keyword. Omit the F keyword if you are not identifying files, and the P keyword if you are not identifying PSBS.

EXECUTE (REQUIRED)
starts the DataRefresher DAP.

PGM=DVRY0000 (REQUIRED)
calls the DataRefresher DAP.

PARM=' ' (REQUIRED)
specifies information to be passed to the Data Dictionary.

All information passed to the Data Dictionary is enclosed in one set of single quotation marks and can contain either or both:

F=(*subject_name*) P=(*subject_name*)

Separate the subject names by commas, and enclose the list in parentheses.

Writing subject names:

- A subject name specified with the P keyword identifies a PSB.
- A subject name specified with the F keyword identifies a file.

The subject names can appear in any order. Each identified file or PSB must be represented as a subject in the Data Dictionary databases. Each subject is identified by its subject name, which can be either a primary name or an alias. Every subject name has four qualifiers:

- A status code
- A subject code
- A user name
- An occurrence number.

If you use all four qualifiers in a subject name, you use the form:

```
(status_code,subject_code, user_name,occurrence_number)
```

Examples

```
(A,F,FILEA,002)
```

This identifies a file represented by a subject whose status code is A, whose subject code is F, whose user name is FILEA, and whose occurrence number is 002.

```
(T,P,PSBA,0)
```

This identifies a PSB represented by a subject whose status code is T, whose subject code is P, whose user name is PSBA, and whose occurrence number is 0.

Write the status code and the subject code exactly as they are known to your Data Dictionary databases. The status code for a file or PSB must always be a letter from A through T, or a number from 0 through 9.

Write the user name as it is known to your Data Dictionary databases unless the name contains special characters. In that case, it must be enclosed in double quotes (").

Write occurrence number as a 1-, 2-, or 3-digit integer, provided that you include all significant digits. For example, you can write 002, 02, or 2 for the occurrence number in the first example above, and you can write 000, 00, or 0 for the occurrence number in the second example. The occurrence number for a PSB must be 0.

Writing abbreviated subject names: You can identify a file by the user name alone if the status code for its subject is P and the occurrence number is 0.

You can identify a PSB by the user name alone if the status code for its subject is P.

Examples

```
FILEC could replace the subject name (P,F,FILEC,000).
```

```
PSBC could replace the subject name (P,P,PSBC,000).
```

Keyword examples: Using the preceding rules, the following are legitimate F and P keywords:

```
F=(FILEC,(A,F,FILEA,002))
F=(FILEF,FILEG,FILEH)
F="FILE-X"
P=(PCBC)
P=((T,P,PSBA,0),PSBC,(S,P,PSBD,000))
```

You need not include the outer parentheses around a list of subject names when the list consists of a single user name. For example, you could write either:

```
P=(PCBC)
```

or

```
P=PCBC
```

However, it is an error not to include the outer parentheses when a list consists of a single fully qualified subject name. For example,

```
F=((P,F,FILEA,000))
```

identifies a single file (FILEA).

But,

```
F=(T,F,FILEA,000)
```

identifies four files whose user names for their subjects are T, F, FILEA, and 000.

The first three of these names could be the names of DXTFILE descriptions. The fourth, however, could not, because 000 does not conform to the naming convention set forth in "DataRefresher naming conventions" on page 18. The DAP recognizes it as a syntax error, and does not process the command.

SUBFIELDS=Y | N (OPTIONAL)
indicates to the DAP whether to generate FIELD keywords for subfields within fields.

Y causes the DAP to write FIELD keywords for each subfield within a field.

N causes the DAP to write FIELD keywords only for fields directly related to a segment—not for subfields within a field.

The DAP writes subfields only when their use is described to the Data Dictionary as type R (redefines) or type C (contains). Subfields that redefine a field start at the same byte at which the main field begins. Subfields contained in fields do not start at the same byte as the field, but are completely contained in the main field. The DAP ignores subfields of usage types D, 6, or 8, and does not support arrays or COBOL picture attributes. For more information about how source data is described to the Data Dictionary, see the *IBM OS/VS DB/DC Data Dictionary Applications Guide*.

DEBUG=*n* (OPTIONAL)
determines what diagnostic information the DAP should write when it executes your Data Dictionary EXECUTE command.

Replace *n* with a value of 1, 2, 3, or 4. The larger the indicated number, the more information will be written. The default is 1.

Unless you are trouble-shooting, set *n* to 1. A value of 1 specifies that diagnostic information is to be printed only in the event of a terminating program or system error. For the DAP, unlike the UIM, a debug level of 2 gives information about the Program Access Facility.

For details on what is printed for the different values of *n*, refer to Appendix E, "Diagnostic information" on page 257.

Example of EXECUTE

```
EXEC PGM=DVRY0000+
  PARM='F=(FILEC,(P,F,FILEA,002)),P=PCBC,SUBFIELDS=Y,DEBUG=1';
```

Notes:

1. There are three subject names being passed to the Data Dictionary. Two identify a file and one identifies a PSB.
2. The first file is FILEC. Since the status code and occurrence number are not specified, they default. The subject defaults to P and the occurrence number defaults to 0.
3. The second file is (P,F,FILEA,002). All qualifiers are specified.
4. The PSB is PCBC, indicating that the status code of its subject is P.
5. SUBFIELDS=Y causes the DAP to write FIELD keywords for each subfield within a field.
6. DEBUG = 1 will cause diagnostic information to be printed only in the event of a terminating program or system error.

Chapter 8. DEM commands

This chapter contains an alphabetic listing of all Data Extract Manager (DEM) commands. These commands can be used to extract data from non-relational data sources.

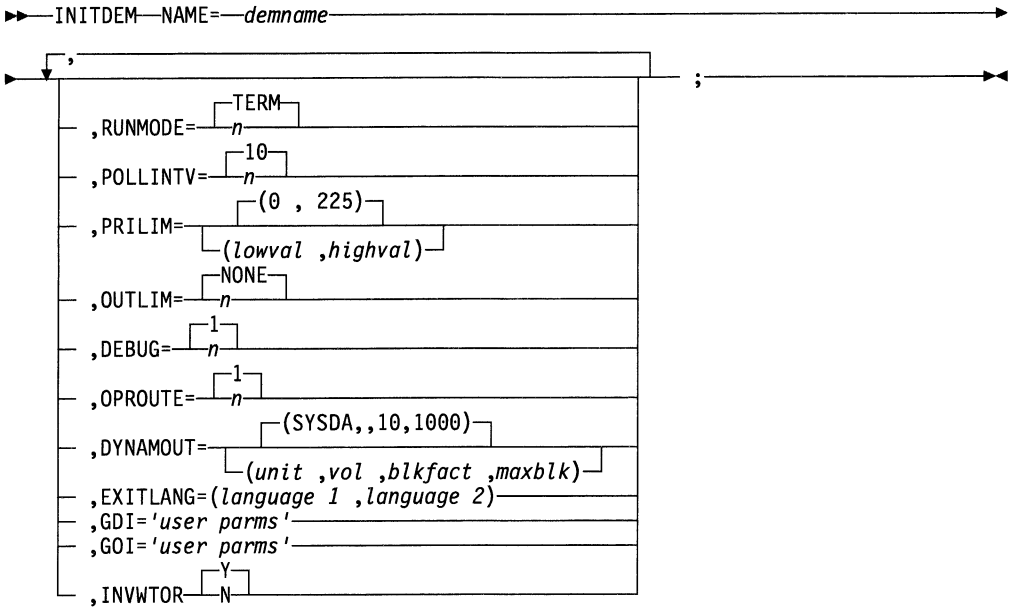
DEM data sources include: IMS databases, physical sequential data sets VSAM data sets, sources accessed with generic data interface (GDI) exits, and sources on remote nodes accessed with DataRefresher.

For more information about using the DEM commands, see the *DataRefresher MVS and VM User's Guide*.

INITDEM

Use the INITDEM command as a DEM initialization command to identify a DEM and specify some of its operating characteristics. It does not establish what data the DEM can access.

Code this command as part of a DEM's DXTIN data set.



INITDEM (REQUIRED)
identifies a DEM and initially specifies some of its operating characteristics.

NAME=*demname* (REQUIRED)
assigns a name to a DEM.

Replace *demname* with the name you want to assign to the DEM. The name must be an alphameric/special characters name, as discussed in "DataRefresher naming conventions" on page 18.

To help eliminate confusion over the identities of a DEM, consider naming a DEM based on the JCL job name. DataRefresher does not let you run DEMs with identical NAMEs at the same time.

RUNMODE=TERM | *n* (OPTIONAL)
determines whether a DEM runs in terminating or continuous (for *n* number of minutes) run mode.

TERM
sets the DEM to terminating run mode.

The DEM executes all qualifying extract requests in the EXTLIB and then stops.

n sets the DEM to continuous run mode.

The DEM will execute all qualifying extract requests in the EXTLIB and then periodically recheck the EXTLIB for new extract requests to execute.

Replace *n* with the time in minutes you want the DEM to run in continuous mode. It can be an integer from 1 to 32767. When this time runs out, the DEM will change to terminating mode.

POLLINTV=*n* (OPTIONAL)
determines how long (in seconds) the DEM should wait after running all qualifying extract requests in the EXTLIB before rechecking the EXTLIB for new extract requests to execute. This period of time is called the *polling interval*.

Replace *n* with an integer from 1 to 3600 to establish the length (in seconds) of the polling interval. The default value is 10 seconds.

If the DEM is running in terminating run mode, the DEM ignores this keyword.

PRILIM=(*lowval,highval*) (OPTIONAL)
determines a DEM's priority range. The DEM will execute only those extract requests having a priority value from the *lowval* through and including the *highval*.

Replace *lowval* and *highval* with whole numbers such that:

$0 \leq lowval \leq highval \leq 255$

The default values are 0 and 255 for *lowval* and *highval*, respectively. If you include the PRILIM keyword, you must code both the *lowval* and the *highval* even if one of them equals its default value.

OUTLIM=NONE | *n* (OPTIONAL)
determines a DEM's output limit. The DEM selects for execution only those extract requests having output row estimates less than or equal to this output limit.

When the DEM starts to execute an extract request having an output row estimate meeting the preceding condition, and then that extract request's output reaches the limit, the DEM immediately stops writing that extract request's output even if the output is not complete. The user receives partial output and a message saying that the output limit was reached for that extract request. The extract request is removed from the EXTLIB. Increase the value assigned to OUTLIM in order to get more output.

NONE
sets the DEM to have no limit on the number of output rows it writes per extract request.

n sets the DEM at a particular output limit.

Replace *n* with an integer from 1 to 10000000 (no commas) to establish the maximum number of output rows the DEM writes per extract request.

DEBUG=*n* (OPTIONAL)
determines the DEM's debugging level. Replace *n* with a number from 1 to 4. For a description of the type of diagnostic information associated with each debugging level see Appendix E, "Diagnostic information" on page 257.

For some system errors, the system prints control area and trace information regardless of what debugging level you choose. If you include

a DXTDUMP DD statement, system errors cause a printout (not a dump) of selected storage areas.

A DataRefresher user assigns a debugging level to a given extract request. That debugging level is ignored unless it is higher than the debugging level you establish for the DEM that executes that extract request. If a DataRefresher user does assign a debugging level higher than 1 to a given extract request, the DEM that executes that extract request will not batch it with any other extract requests.

OPROUTE=*n* (OPTIONAL)
specifies the routing code by which a DEM sends operator messages to an MVS console. This keyword does not specify which MVS operator console is monitoring the DEM.

You can specify that more than one message routing code be sent to a single MVS console. The decision about which MVS consoles receive operator messages occurs at MVS system generation. See *OS/VS2 MVS System Commands* for more information.

Replace *n* with the number (1 through 16) of the desired console.

DYNAMOUT=(*unit,vol,blkfact,maxblk*) (OPTIONAL)
specifies the defaults for dynamic allocation of physical sequential data sets. Defaults may be overridden by the EXTDATA keyword of the SUBMIT command.

unit
specifies the unit of the DASD device.

vol specifies the volume serial. The default value is unspecified to automatically permit the system to assign a volume serial.

blkfact
represents the number of extract records in a block. Replace *blkfact* with an integer from 1 through 5000 inclusively.

maxblk
represents the maximum number of blocks which will be allocated for a data set.

Replace *maxblk* with an integer from 1 through 2147483647.

EXITLANG=(*language1,language2*) (OPTIONAL)
specifies the language environments that exit routines require.

language1,language2
specify the languages in which the exit routines are written.

There are four languages that are valid when using exits:

- PL/I
- COBOL
- LE/370
- ASSEMBLER

Assembler is always assumed, whether or not you include the EXITLANG keyword.

Replace *language1, language2* with PLI, COBOL or LE (for LE/370).

Enclosing parentheses for this operand are optional when specifying only one language.

If you do not know the language in which an exit routine is coded, see your system or database Administrator.

GDI='*user parms*' (OPTIONAL)
passes the necessary parameters to the GDI exit while running the DEM.

'user parms'
is limited to a maximum of 94 characters and that value is passed, unchanged, to the user exit.

GOI='*user parms*' (OPTIONAL)
passes the necessary parameters to the GOI exit while running the DEM.

'user parms'
is limited to a maximum of 255 characters and that value is passed, unchanged, to the user exit.

INVWTOR=Y | N (OPTIONAL)
optionally suppresses the invitational WTOR (Write to Operator with Reply).

Upon successful completion of initialization of the DEM, a message is issued to the operator console:

```
WTOR: DVRM01E DXT DATA EXTRACT MANAGER "name"
IS READY TO ACCEPT OPERATOR COMMANDS.
```

The message was intended to be invitational and did not demand a response. You can use the INVWTOR command to suppress this message.

When INVWTOR=N, the operator is denied the ability to enter commands to that DEM.

Example of INITDEM

This example creates a DEM called DEMTWO that:

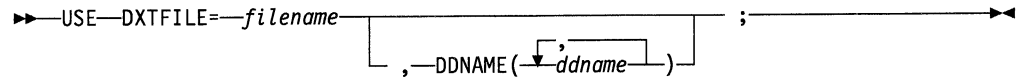
- Runs in continuous run mode for 480 minutes (8 hours)
- Runs with a polling interval of 300 seconds (5 minutes)
- Has a priority range of 200-255
- Has an output limit of 5000 rows per extract request
- Has a debugging level of 2
- Allows an MVS operator to control it through a console that receives messages from routing code 16
- Has a maximum number of 1200 blocks for dynamic allocation of a physical sequential data set
- Specifies that exit routines are written in COBOL
- Passes information to the GDI exit

```
INITDEM NAME=DEMTWO,RUNMODE=480,POLLINTV=300,PRILIM=(200,255),
        OUTLIM=5000,DEBUG=2,OPROUTE=16,DYNAMOUT=(,,,1200),
        EXITLANG=COBOL,GDI='SECURITY INFORMATION';
```

USE DXTFILE

Use this DEM initialization command to provide the DEM with access to the necessary file descriptions in the FDTLIB.

You must code at least one USE DXTFILE command as part of a DEM'S DXTIN data set for each file that the DEM can access. You can code more than one USE DXTFILE command.



USE (REQUIRED)
provides the DEM with access to necessary file descriptions.

DXTFILE=*filename* (REQUIRED)
specifies the name of the DXTFILE associated with the VSAM data set, physical sequential data set, or GDI data source that a DEM can access.

Replace *filename* with the name of the DataRefresher file description in the FDTLIB.

DDNAME=*ddname* (OPTIONAL)
specifies the DD name(s) that is used when allocating the file in the DEM JCL.

For non-GDI files, you must include DDNAME if the required file's DXTFILE description does not include a DD name. You can also use this keyword to override the DD name specified on the DXTFILE description.

You can specify only one DD name for non-GDI data sources.

For GDI data sources, specifying one or more DD name(s) with USE DXTFILE required file's DXTFILE description is optional. You can specify up to eight DD names for GDI data sources.

If you are specifying only one DD name, you can omit the parentheses.

Examples of USE DXTFILE

Example 1

This example accesses the data file associated with the DXTFILE named ACCTSREC.

Assume that the FDTLIB file description contains its DD name and that you do not want to override that name.

```
USE DXTFILE=ACCTSREC;
```

Example 2

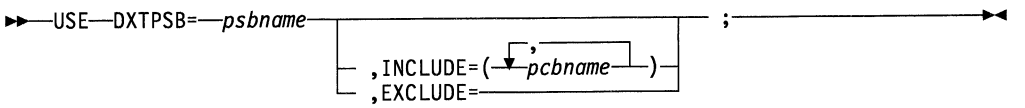
This example accesses the data file associated with the DXTFILE named ACCTSPAY and a DD name of VSAM2. Assume that the FDTLIB file description does not contain its DD name.

```
USE DXTFILE=ACCTSPAY,DDNAME=VSAM2;
```

USE DXTPSB

Use this DEM initialization command to specify a single PSB that a given DEM can use for IMS database access.

If you want a DEM to have access to IMS databases, you must code this command as part of that DEM'S DXTIN data set.



USE (REQUIRED)
provides the DEM with access to necessary file descriptions.

DXTPSB=*psbname* (REQUIRED)
specifies the DXTPSB to which a DEM can have access.

Replace *psbname* with the appropriate DXTPSB's actual name. The *psbname* must be listed in the FDTLIB and must correspond to the IMS PSB being used (passed on the EXEC parameter in the JCL associated with the DEM).

INCLUDE=*pcbname* (OPTIONAL)
specifies which DXTPCBS in the DXTPSB you want this DEM to use.

Replace *pcbname* with the actual name(s) of the DXTPCB(s) to which you want the DEM to have access. Each *pcbname* you code must be listed in the FDTLIB. Parentheses around the *pcbname* are optional if you code just one.

EXCLUDE=*pcbname* (OPTIONAL)
specifies the DXTPCBS in the DXTPSB you do not want a DEM to use.

Replace *pcbname* with the actual name(s) of the DXTPCB(s) to which you do not want the DEM to have access. Each *pcbname* that you code must be listed in the FDTLIB.

You can code either the INCLUDE or the EXCLUDE keyword, but not both.

Examples of USE DXTPSB

Example 1

This example allows a DEM to have access to all of the DXTPCBS in a DXTPSB named BATCHPSB.

```
USE DXTPSB=BATCHPSB;
```

Example 2

This example allows a DEM to have access to only the DXTPCBs named DATA1 and DATA2 in the DXTPSB named BATCHPSB.

```
USE DXTPSB=BATCHPSB,INCLUDE=(DATA1,DATA2);
```

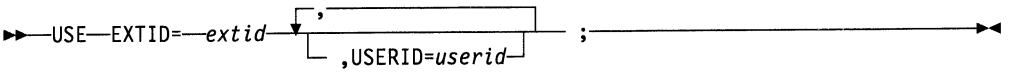
Example 3

This example allows a DEM to have access to all of the DXTPCBS in a DXTPSB named OPSB, except for those named DATA7 and DATA9.

```
USE DXTPSB=OPSB,EXCLUDE=(DATA7,DATA9);
```

USE EXTID

Use this DEM initialization command to enable you to select an extract request (based upon the extract request identifier, and user ID) currently in the EXTLIB awaiting execution. You still must code all the USE DXTFILE, or USE DXTPSB, commands to identify the files, and PSBS, by this extract request. The values specified on the USE EXTID command must match the values that were coded on the SUBMIT command's EXTID, and USERID keywords.



USE (REQUIRED)
provides the DEM with access to a particular extract request.

EXTID=*extid* (REQUIRED)
specifies the name given to an extract request when it was submitted to the UIM. This value must match the value coded on the EXTID keyword of the SUBMIT (UIM) command.

USERID=*userid* (OPTIONAL)
specifies the user ID associated with the extract request when the extract was submitted to the UIM. This provides the DEM with further qualification about the extract request to be run during this execution of the DEM.

This value must match the value coded on the USERID keyword of the SUBMIT command. If the USERID keyword of the SUBMIT command was not included, do not include the USERID keyword here.

Examples of USE EXTID

Example 1

This example notifies the DEM that it can only run extract request(s) which were submitted with the following keywords on the SUBMIT command:

- EXTID=JJLER1
- USERID=MURPHY

```
USE EXTID=JJLER1, USERID=MURPHY;
```

Example 2

This example notifies the DEM that it can only run extract request(s) which were submitted with the following keywords on the SUBMIT command:

- EXTID=JJLER2
- NODE not specified
- USERID not specified

```
USE EXTID=JJLER2;
```

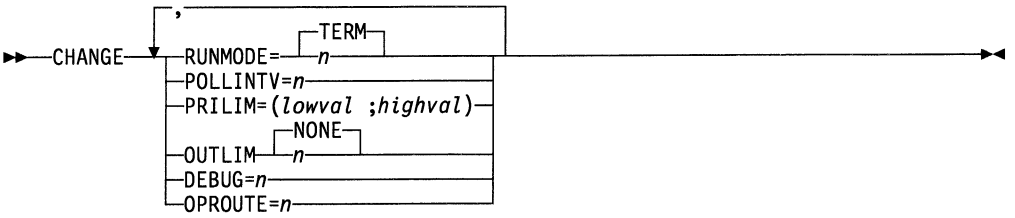
Chapter 9. DEM Operator commands

This chapter contains an alphabetic listing of all Data Extract Manager (DEM) Operator commands. These commands are issued by the MVS operator to control the DEM.

For more information about using the DEM Operator commands, see the *DataRefresher MVS and VM User's Guide.*

CHANGE

Use this DEM operator command to change DEM operating characteristics. It cannot change the characteristics that affect the data accessed by a particular DEM.



CHANGE (REQUIRED)
is issued from the system console by the system operator to initiate a change in the DEM operating characteristics.

RUNMODE = TERM | *n* (OPTIONAL)
can either:

- Change the DEM's run mode from continuous to terminating
- Change the DEM's run mode from terminating to continuous
- Change the amount of time the DEM will remain in continuous run mode

When running in terminating run mode the DEM executes all appropriate extract requests in the EXTLIB and then stops.

When running in continuous run mode the DEM executes all appropriate extract requests in the EXTLIB and then periodically rechecks it for new extract requests to execute.

TERM
changes the DEM's run mode from continuous to terminating.

n changes the DEM's run mode from terminating to continuous and specifies how long it should run in continuous mode, or it changes how long a DEM currently running in continuous run mode will continue to run.

Replace *n* with an integer from 1 through 32767 to establish how many minutes the DEM will run in continuous run mode before changing to terminating run mode.

POLLINTV = *n* (OPTIONAL)
changes the amount of time the DEM running in continuous mode should wait (after executing all selected extract requests in the EXTLIB) before rechecking the EXTLIB for new extract requests to execute. This period of time is called the *polling interval*.

Replace *n* with an integer from 1 to 3600 to establish the new length (in seconds) of the polling interval.

PRILIM = (*lowval,highval*) (OPTIONAL)
changes the DEM's priority range.

The DEM will execute only those extract requests having a priority value

falling inclusively between the low value and the high value of this priority range.

Replace *lowval* and *highval* with whole numbers such that:

```
0 ≤ lowval ≤ highval ≤ 255
```

You must include both a low value and a high value.

OUTLIM = NONE | *n* (OPTIONAL)
changes the DEM's output limit.

The DEM will select for execution only those extract requests having output row estimates that are less than or equal to this output limit.

If the DEM starts to execute an extract request having an output row estimate that meets the preceding condition, then that extract request's output reaches the output limit, the DEM will immediately stop writing that extract request's output.

The values you can specify are:

NONE
changes the DEM so that it will no longer have a limit on the number of rows it writes per extract request.

n sets the new output limit of the DEM.

Replace *n* with an integer from 1 through 10000000 (without commas) to indicate the maximum number of rows the DEM will write per extract request.

DEBUG = *n* (OPTIONAL)
changes the DEM's debugging level.

Replace *n* with a number from 1 through 4. See Appendix E, "Diagnostic information" on page 257 for a description of the type of diagnostic information associated with each level.

A DataRefresher user can also assign a debugging level to a given extract request. This debugging level applies only if it is greater than the debugging level you establish for the DEM executing the extract request. If a DataRefresher user does assign a debugging level to a given extract request, the DEM that executes that extract request will not batch it with any other extract requests.

Note that diagnostic information gets printed in a DXTDUMP data set.

OPROUTE = *n* (OPTIONAL)
changes the routing code by which the DEM sends operator messages to an MVS console.

Replace *n* with the number (1 through 16) of the appropriate console. See *OS/VS2 MVS System Commands* for more information.

Examples of CHANGE

Example 1

This example changes the DEM's run mode from continuous to terminating:

```
CHANGE RUNMODE=TERM
```

Example 2

This example changes the DEM's run mode from terminating to continuous. It will run in continuous run mode for 120 minutes (2 hours):

```
CHANGE RUNMODE=120
```

Note: The previous command would also change how long the DEM currently running in continuous run mode will continue to do so. After issuing the previous command, the DEM would continue to run in continuous run mode for two more hours.

Example 3

This example changes the DEM's polling interval to 600 seconds (10 minutes):

```
CHANGE POLLINTV=600
```

Note: If the DEM is running in terminating run mode, changing its polling interval will have no effect.

Example 4

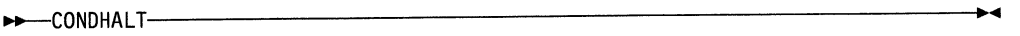
This example changes the DEM so that it:

- Has a priority range of 200-255
- Has no limit on the number of output rows it prints per extract request
- Has a debugging level of 1
- Sends operator messages to an MVS operator console via routing code number 1

```
CHANGE PRILIM=(200,255),OUTLIM=NONE,DEBUG=1,OPROUTE=1
```

CONDHALT

Use this DEMoperator command to temporarily stop a DEM and displays the names and start times of the extract requests that are currently executing.



CONDHALT
is used only by the system operator to temporarily stop a DEM.

After you enter the CONDHALT command, the DEM stops executing. The names and start times of those extract requests currently being executed by the DEM are displayed on the MVS console.

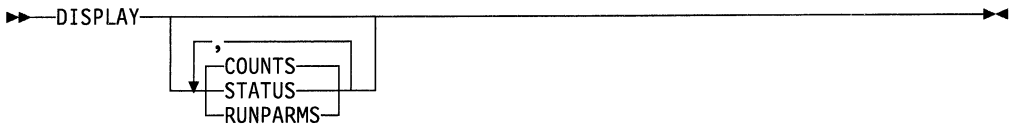
After all the preceding information has been displayed, a prompt requesting that you enter either the RESUME or HALTBATCH command appears. If you enter the RESUME command, the DEM resumes execution. If you enter the HALTBATCH command, the DEM stops executing its current batch of extract requests and queue the extract requests again. They then become eligible for processing by any DEM for which those extract requests qualify.

If you do not respond to the prompt within five minutes, the prompt is repeated. If you do not respond to the third appearance of the prompt, the DEM resumes execution.

DISPLAY

Use this DEM operator command to display:

- The current values of a DEM's operating characteristics
- Information about extract requests in the execution cycle



DISPLAY (REQUIRED)
is used by the system operator to display the DEM's operating characteristics.

COUNTS (OPTIONAL)
displays the following information:

- Number of extract requests in the EXTLIB
- Number of extract requests currently being executed by all DEM jobs
- Number of extract requests currently being executed by the DEM to which you sent the DISPLAY command
- Number of extract requests in the EXTLIB that qualify for execution by the DEM to which you sent the DISPLAY command

STATUS (OPTIONAL)
displays the names, originators, start times, and rows of output of those extract requests currently being executed by the DEM to which you sent the DISPLAY command.

RUNPARMS (OPTIONAL)
displays the current values of the following keywords:

RUNMODE
POLLINTV
PRILIM
OUTLIM
DEBUG
OPROUTE
EXITLANG
DYNAMOUT

Examples of DISPLAY

Example 1

This example displays the COUNTS information. The default of DISPLAY is COUNTS, so you could enter:

```
DISPLAY
```

Example 2

This example displays only STATUS and RUNPARMS information:

```
DISPLAY   STATUS,RUNPARMS
```

Note: COUNTS information is not displayed since other DISPLAY keywords are included and COUNTS is omitted.

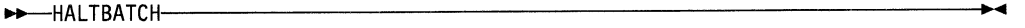
Example 3

This example displays COUNTS information as well as STATUS and RUNPARMS information, enter the following:

```
DISPLAY   COUNTS,STATUS,RUNPARMS
```

HALTBATCH

Use this DEM operator command to stop a DEM's execution of its current batch of extract requests after it has been temporarily stopped by the CONDHALT (DEM) command.

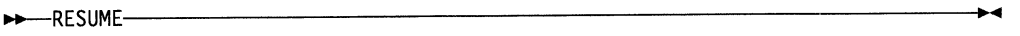


HALTBATCH (REQUIRED)
is used only by the system operator to stop the current batch of extract requests being executed by this DEM. These extracts will then be queued again in the EXTLIB to be processed by any other DEM for which they are eligible.

The DEM must have been temporarily stopped by the CONDHALT command before the HALTBATCH command is used.

RESUME

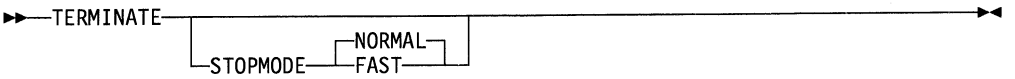
Use this DEM operator command to restart a DEM after it has been temporarily stopped by the CONDHALT command.



RESUME (REQUIRED)
causes a DEM, that has been temporarily stopped by the CONDHALT command, to resume execution. The DEM will resume execution at the point where the CONDHALT command stopped it.

TERMINATE

Use this DEM operator command to stop a DEM.



TERMINATE (REQUIRED)
stops a DEM.

STOPMODE=NORMAL | FAST (OPTIONAL)
determines whether a DEM stops immediately or after it finishes executing its current batch of extract requests.

The values you can specify are:

NORMAL
indicates that the DEM must finish executing its current batch of extract requests before it stops.

FAST
indicates that the DEM must stop immediately. The extract requests will be queued again for execution by a DEM having the same INITDEM NAME.

Examples of TERMINATE

Example 1

This example stops a DEM so that it first finishes executing its current batch of extract requests. STOPMODE=NORMAL is the default value.

```
TERMINATE
```

You can accomplish the same thing by entering:

```
TERMINATE STOPMODE=NORMAL
```

Example 2

This example stops a DEM immediately:

```
TERMINATE STOPMODE=FAST
```

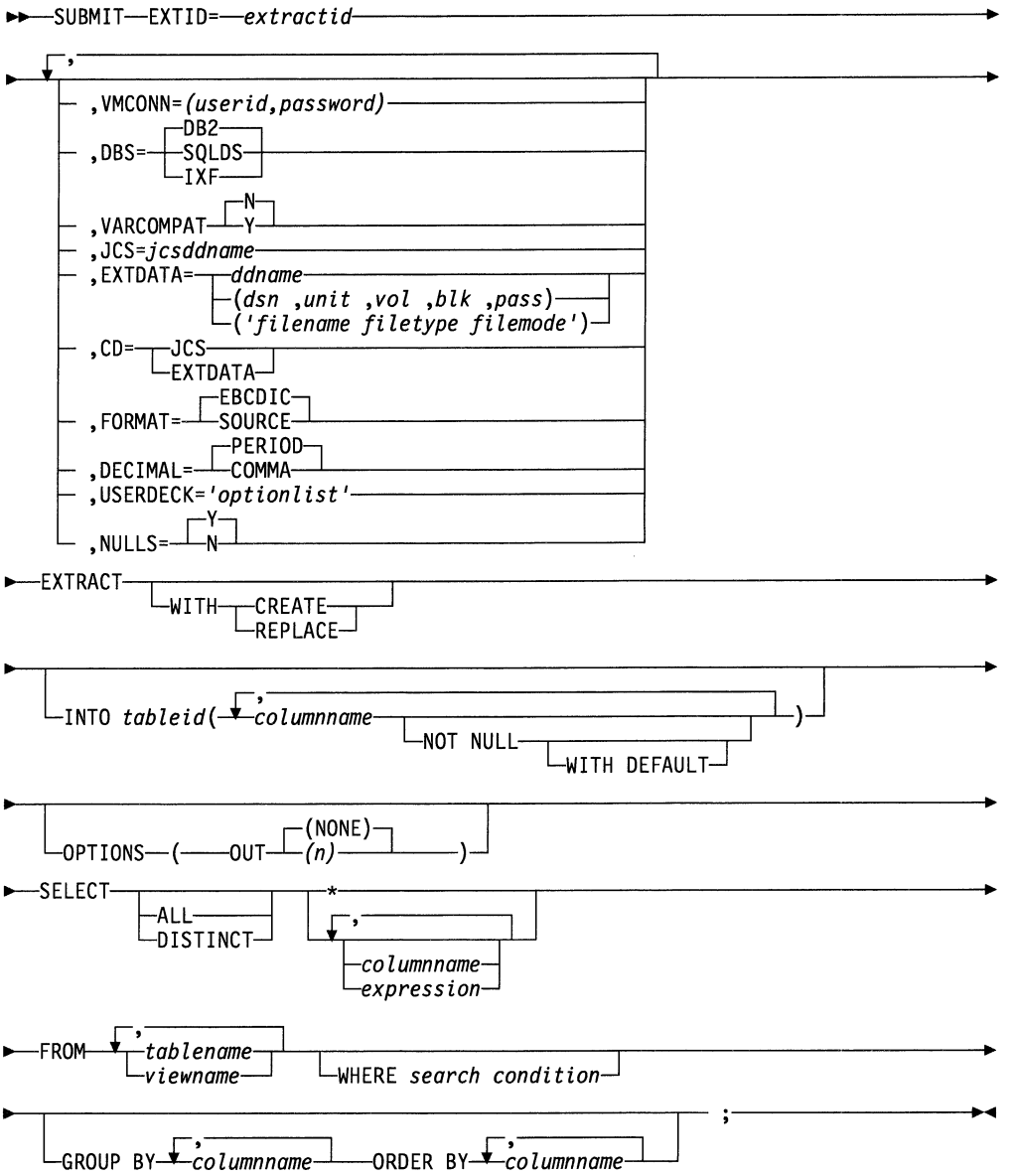
Chapter 10. REM commands

This chapter contains an alphabetic listing of all Relational Extract Manager (REM) commands. These commands can be used to extract data from relational data sources such as DB2 and SQL/DS databases.

For more information about using Relational Extract Manager commands, see the *DataRefresher MVS and VM User's Guide*.

SUBMIT/EXTRACT

The SUBMIT (REM) command and the EXTRACT (REM) statement are used to extract data from relational databases.



SUBMIT (REQUIRED)
submits an extract request.

EXTID=*extractid* (REQUIRED)
identifies the extract ID of your extract request.

Replace *extractid* with an SQL or DBCS name.

VMCONN=(*userid,password*) (OPTIONAL, used in VM only)
identifies the user ID and password for DataRefresher to use to explicitly connect to SQL/DS. VMCONN is only used for a REM running under VM; it is not used for MVS.

If the VMCONN keyword is not specified, implicit connection to the SQL/DS database will occur with the VM logon ID under which the REM is being executed.

userid
is a valid SQL/DS user ID.

password
is the password associated with the SQL/DS user ID.

DBS=DB2 | SQLDS | IXF (OPTIONAL)
identifies the database system into which your extract output is loaded. By writing it, you notify the REM of the type of load control deck it should generate.

DB2
loads your extracted output into a DB2 table.

SQLDS
loads your extracted output into an SQL/DS table.

IXF
writes your extracted output in IXF.

Control information (IXF header records) is always included with the extracted data. If you specify DBS=IXF, you must also specify the CD keyword. The USERDECK keyword is not valid if DBS=IXF.

Note: If you specify DBS=IXF, the data set or file must have a variable length record format. If you specify DBS=DB2 or DBS=SQLDS, the data set or file must have a fixed length record format.

VARCOMPAT= N | Y (OPTIONAL)
indicates how DB2 variable length fields, for example VARCHAR, are to be extracted. The VARCOMPAT keyword is only valid when DBS=DB2.

N The 2-byte length field is included in the extracted data.

Y The 2-byte length field is not included in the extracted data. VARCOMPAT=Y provides compatibility with DXT* Version 2 Release 3 and earlier. Do not use the CD keyword when using VARCOMPAT=Y.

JCS=*jcsddname* (OPTIONAL)
indicates that you want DataRefresher to generate an output job. In addition to starting a load utility, the JCS could start the SORT utility or any other program you may want to use.

Replace *jcsddname* with the DD name of the data set or the FILEDEF containing the JCS associated with your extract request. If you do not include the JCS keyword, you must include the EXTDATA keyword.

EXTDATA=*ddname* | (*dsn,unit,vol,blk,pass*) |
'*filename filetype filemode*' (OPTIONAL)
identifies the data set to which REM writes the output. This is used only if you want to direct some or all of your extract request output to a physical sequential data set or a CMS file. This physical sequential data set must not have a variable record format (RECFM V) except when the extracted output is in IXF format.

The minimum logical record length (LRECL) of a physical sequential data set or CMS file is 20, even if the extracted data has a smaller LRECL.

The maximum LRECL size supported by MVS and VM is 32760. Therefore, the REM's maximum row size is 32760. DataRefresher calculates the truncation of LONG VARCHAR and LONG VARGRAPH columns to produce an output record not greater than 32760 bytes in length.

If the EXTDATA keyword is not included, you must write the JCS keyword and the JCS that you name must contain an *EO statement.

If your job is run under MVS, DataRefresher lets you specify *DSN as a substitution variable in the DD statement in your JCS for the output data set name. When DataRefresher is executed, the data set name specified in the EXTDATA keyword replaces the *DSN in the DD statement.

If your job is run under VM, DataRefresher lets you specify *FN as a substitution variable in the FILEDEF statement for the output file name (in the JCS). When DataRefresher is executed, the CMS file name specified in the EXTDATA keyword will replace the *FN in the FILEDEF statement.

For data in IXF written to a CMS file, DataRefresher will choose unblocked records with RECFM=V if a file is dynamically allocated.

When data in IXF is written into a generated output job, the variable length records are broken up into 80-byte record lengths, with the standard DataRefresher use of the character # to indicate continuation.

When extract requests are batched and extracted to the same tape, the tape should be specified as non-labelled.

The values you can specify for EXTDATA are:

ddname
identifies the name of a DD statement to which the REM writes your extracted output. The DD name that you write must have a corresponding DD statement in the JCL that you use to run the REM in MVS, or a corresponding FILEDEF in VM. In addition, if you write this keyword and the JCS keyword on your extract request, the JCS that you refer to must contain the data set name that you write on this keyword.

Note: You cannot use the name DXTOUT or SYSOUT as a ddname.

(*dsn,unit,vol,blk,pass***)**
indicates to the REM that you want it to allocate (and catalog) a data set for your output while it is processing your extract request. This format is used for the MVS REM only.

The REM saves and catalogs all data sets it dynamically allocates. Before reusing a dynamically allocated data set, first delete and uncatalog it. If you try to reuse a dynamically allocated data set before uncataloging it, MVS sends a message requesting that you cancel or provide a device type.

If you write this form of EXTDATA, include *dsn*, but *unit, vol, blk,* and *pass* are optional. Include commas for every value you do not write, unless the comma (or commas) is not followed by another value.

dsn
assigns a name to the data set that REM dynamically allocates. You must include *dsn*.

The name that you write can be up to 8 characters in length. The first letter of the name must be alphabetic, a special character, or a hyphen; the remaining characters can be alphabetic, a special character, hyphens, or numeric.

dsn can also be a series of up to five names, joined by periods, of no more than 8 characters each. If you concatenate names, the total length of the string can be no longer than 44 characters.

unit
overrides the default unit identification for your data set. If the unit that you write is a device, replace *unit* with a name of 1 to 8 characters in length; the characters can be alphameric/special characters or hyphens. If the unit is the name of a user group, replace *unit* with a name of 1 to 8 alphameric characters in length.

(If the value of *unit* is not compatible with the value of *vol*, MVS requests more information from the operator and waits for a response before continuing.)

The default value of *unit* is **SYSDA**.

vol overrides the default volume serial identification for your data set. You can specify only one volume serial; if you need multiple volumes, you must use the *ddname* form of EXTDATA. Replace *vol* with hyphens or with an alphameric/special characters name of 1 to 6 characters in length.

(If the value of *vol* is not compatible with the value of *unit*, MVS requests more information from the operator and waits for a response before continuing.)

blk overrides the default blocking factor for your extract request. The blocking factor determines the number of extract records that REM includes in each block. Replace *blk* with an integer from 1 to 5000.

pass
assigns a password to the dynamically allocated data set. Replace *pass* with an alphameric/special characters name.

Under MVS, if you choose to protect your data set with a password, you should have an MVS password data set. (See *OS/VS2 MVS Utilities* for more information about the IEHPROGM utility which you can use to maintain your password data set.)

If you want to reuse a data set and it has a password, you must know the password before you can delete and uncatalog the data set.

Note: When you are dynamically allocating a data set and DBS=IXF, DataRefresher overrides the default values for the DCB:

- RECFM is VB.
- LRECL is the length of the longest IXF record type—85 unless the data record is longer than 85.
- BLOCKSIZE (where DL denotes the length of the data record).

If DL <= 4180, BLOCKSIZE is *n* * DL + 4, where *n* is the integer part of 8360/DL (remainder ignored), and the 4 accommodates the block descriptor word (BDW).

If DL > 4180, BLOCKSIZE is DL + 4, where the 4 accommodates the BDW.

DL > 32756 is considered to be an error condition.

'*filename filetype filemode*'**(used in VM only)**
indicates that you want the REM to create a CMS file for your output while it is processing your extract request (VM REM). Replace '*filename filetype filemode*' with the appropriate file descriptors.

The designated *filemode* must be accessible to REM in write mode.

CD=JCS | EXTDATA (OPTIONAL)
indicates that you want the REM to generate a load control deck and/or column descriptors.

The value that you write determines where the REM places the load control deck and/or column descriptors in the output.

The CD keyword must be included for data in IXF. If the extracted data is going to a data set or file then CD=EXTDATA. If extracted data is going to an output job, then CD=JCS. The IXF header, table, and column records must be included with the extracted data.

The REM-generated load utility control deck contains statements that are formatted specifically for the load utility that you start to load your extracted data into a relational table (DB2 or SQL/DS); the statements include the load commands and descriptions of the data, target table, and target columns. (The USERDECK keyword on the SUBMIT command can be optionally used to override default values otherwise applied to a REM-generated load control deck.)

The column descriptors, on the other hand, contain no load commands but describe the data and the positions of the fields within the data records.

For the REM to create a control deck or column descriptors, you must write both the CD keyword on the SUBMIT command and the INTO keyword on the EXTRACT command for your extract request.

The values of CD are:

JCS
identifies the JCS data set to use if the REM is to generate an output job for loading or transmitting extracted data. When you include CD=JCS, replace '*jcsddname*' on the JCS keyword with the name of the data set or file containing the JCS associated with the extract request. If CD=JCS, the REM generates a load control deck and/or column descriptors and inserts them into an output job. That JCS must contain a *CD statement so that the REM knows where to place the generated control deck and/or column descriptors.

EXTDATA
specifies that the REM generates a load control deck (or column descriptors) and inserts it into the beginning of the physical sequential data set or CMS file that you specified on the EXTDATA keyword. If you specify CD=EXTDATA, you must also include the EXTDATA keyword.

- When your data is targeted for a DB2 table, you can direct the control deck and the extracted data in either of two ways:

  The same place (both the control deck and the data to an output job or both to the same output data set or file)

  Different places (the control deck to an output job and the extracted data to an output data set or file)

  You can, therefore, write either JCS or EXTDATA on the CD keyword.

- When your data is targeted for an SQL/DS table, you should direct the control deck and the extracted data to the same place — to an output job or to the same output data set or file. Therefore, when directing your extract request output to an SQL/DS table by way of a data set or file, you must use the value EXTDATA on the CD keyword. Do not use the value JCS.

- When your data is targeted for a data set or file in IXF, CD=EXTDATA must be specified.

  When your output data in IXF is going to an output job, CD=JCS must be specified.

  The control deck must be included with the extracted output data.

The CD keyword determines the destination of your extract output. The three tables below summarize the results when CD is not included, when CD=JCS, and when CD=EXTDATA.

- When CD is not written:

	JCS written?	EXTDATA written?	*EO in JCS?	Results are...
1	YES	YES	--	VALID
2	YES	NO	YES	VALID
3	YES	NO	NO	INVALID
4	NO	YES	--	VALID
5	NO	NO	--	INVALID

In the three valid situations (1, 2, and 4), the output from the extract request is a physical sequential data set containing the extracted data only — no control deck or column descriptors are generated. Note that in the second valid situation, the output replaces the *EO statement in the JCS and remains there.

- When CD=JCS:

	JCS written?	*CD in JCS?	EXTDATA written?	*EO in JCS?	Results are...
1	YES	YES	YES	--	VALID
2	YES	YES	NO	YES	VALID
3	YES	YES	NO	NO	INVALID
4	YES	NO	--	--	INVALID
5	NO	--	--	--	INVALID

In the first valid situation (1), the output from the extract request is a load utility job and a physical sequential data set containing the extracted data. In the second valid situation (2), the extract request output is a load utility job containing the extracted data.

- When CD=EXTDATA:

	JCS written?	EXTDATA written?	Results are...
1	YES	YES	VALID
2	YES	NO	INVALID
3	NO	YES	VALID
4	NO	NO	INVALID

In the two valid situations (1 and 3), the extract request output is a physical sequential data set containing the extracted data and the generated control deck or column descriptors.

Data types: The data types in the control deck will vary depending on the source column data type, target database, and the value specified on the FORMAT keyword. The following table shows the data types when the target database is SQL/DS:

Source column data type is:	Target database is SQL/DS and	
	FORMAT= SOURCE	FORMAT= EBCDIC
CHAR	CHAR	CHAR
VARCHAR	CHAR	CHAR
LONG VARCHAR	CHAR	CHAR
GRAPHIC	GRAPHIC	GRAPHIC
VARGRAPHIC	GRAPHIC	GRAPHIC
LONG VARGRAPHIC	GRAPHIC	GRAPHIC
FLOAT	FLOAT	CHAR
REAL	FLOAT	CHAR
INTEGER	FIXED	CHAR
SMALLINT	FIXED	CHAR
DECIMAL	DECIMAL	CHAR
DATE	CHAR	CHAR
TIME	CHAR	CHAR
TIMESTAMP	CHAR	CHAR

The following table shows the data types when the target database is DB2:

Source column data type is:	Target database is DB2 and	
	FORMAT= SOURCE	FORMAT= EBCDIC
CHAR	CHAR	CHAR
VARCHAR	VARCHAR	VARCHAR
LONG VARCHAR	VARCHAR	VARCHAR
GRAPHIC	GRAPHIC	GRAPHIC
VARGRAPHIC	VARGRAPHIC	VARGRAPHIC
LONG VARGRAPHIC	VARGRAPHIC	VARGRAPHIC
FLOAT	FLOAT (53)	FLOAT EXTERNAL (53)
REAL	FLOAT (21)	FLOAT EXTERNAL (21)
INTEGER	INTEGER	INTEGER EXTERNAL
SMALLINT	SMALLINT	INTEGER EXTERNAL
DECIMAL	DECIMAL	DECIMAL EXTERNAL
DATE	DATE EXTERNAL	DATE EXTERNAL
TIME	TIME EXTERNAL	TIME EXTERNAL
TIMESTAMP	TIMESTAMP EXTERNAL	TIMESTAMP EXTERNAL

FORMAT=EBCDIC | SOURCE (OPTIONAL)
specifies to the REM how to format the data it extracts. The values you can specify are:

EBCDIC
indicates that you want the REM to convert your extracted data to EBCDIC format.

SOURCE
indicates that you want the REM to leave the extracted data in source format.

Numeric data in source format is described in *IBM 370 XA Principles of Operation* or *IBM S/370 Principles of Operations*.

DECIMAL=PERIOD | COMMA (OPTIONAL)
specifies the format of the decimal point character in your extracted output.

Write:

- DECIMAL=COMMA if you want the decimal point to be represented by a comma
- DECIMAL=PERIOD if you want the decimal point to be represented by a period.

USERDECK='*optionlist*' (OPTIONAL)
explicitly provides DB2 or SQL/DS load utility options. These options are included in the control deck generated by the REM for SQL/DS or DB2.

This keyword is invalid if DBS=IXF.

- For the DB2 LOAD statement, DataRefresher generates the CONTINUEIF keyword which indicates a value used for a continuation field in the input record. All other LOAD keywords will use the DB2 default values unless specifically overridden by the USERDECK keyword.
- For the SQL/DS DATALOAD utility, DataRefresher generates the CONTINUED(YES) keyword in the INFILE statement to indicate that the input data may span more than one control file input record. All other DATALOAD utility keywords will use the SQL/DS default value unless specifically overridden by the USERDECK keyword.

'*optionlist*'
represents selected DB2 or SQL/DS parameters the user wishes to include in the REM-generated LOAD or DATALOAD commands. The REM does not validate the '*optionlist*' variable and, therefore, any errors will be found only when the DB2 LOAD or SQL/DS DATALOAD utility is invoked. For information on the options you can specify for the DB2 LOAD utility, see the LOAD statement in *IBM DATABASE 2 Version 2: SQL Reference*.

The USERDECK keyword takes the form:

```
USERDECK='parm-1,...,parm-n'
```

where each of the '*parm-n*' specifications is replaced by a DB2 or SQL/DS load utility option.

RESUME(NO) is the default for the USERDECK keyword.

Note:

The REM will not provide for RESUME(YES) on the DB2 LOAD statement; users who want their extract request output to be loaded into a table that already contains data must specify USERDECK='RESUME(YES)' in their corresponding requests.

Specification of the USERDECK keyword is not allowed when DBS=IXF. The header record in IXF output is similar to the LOAD and DATALOAD control decks that DataRefresher generates for DB2 and SQL/DS, but the IXF header must be in a set format and thus should not be overridden by a user.

NULLS=Y | N (OPTIONAL)
lets you optionally suppress the inclusion of all DataRefresher null separator fields in the extract output.

Suppression of these fields lets you extract data for use by application programs and DBMSs that do not recognize what the DataRefresher null separator field is or how to handle it. Suppression of the null separator fields also reduces the amount of space needed for extract output, which can be a consideration for users or installations that store large amounts of extract output.

Y specifies that you want null separators inserted (as with in previous releases of DataRefresher).

N specifies that you do not want null separators inserted for any columns under any condition. In this case, DataRefresher cannot indicate null values for any data columns, regardless of the null values you specify on the INTO keyword.

When extract output is directed to a file in IXF, NULLS=N means that only non-nullable IXF data types will be used in the output file.

EXTRACT statement

EXTRACT (REQUIRED)
is specified to extract data from a relational database (DB2 or SQL/DS).

WITH CREATE | REPLACE (OPTIONAL)
is included when your target is a DB2 or SQL/DS table. Use it to:

- Create a new DB2 or SQL/DS table (WITH CREATE)
- Replace all the data in an existing DB2 or SQL/DS table (WITH REPLACE)

INTO *tableid (columnname(s))* **(OPTIONAL)**
identifies the name of the table and the table columns where the extracted data is to be inserted. It is needed only if the extracted data is to be loaded into a relational database table.

OPTIONS (OUT (NONE) | (*n*)) (OPTIONAL)
affects how much output (in output rows) your extract request produces.

SELECT (REQUIRED)
identifies the field(s) to be extracted from a DB2 or SQL/DS database. It is coded as if it were to be submitted directly to the database. The SELECT statement is not syntax checked by DataRefresher; it is passed to DB2 or SQL/DS for error checking.

FROM (REQUIRED)
identifies the table or view from which data is to be selected.

WHERE *search-condition* **(OPTIONAL)**
conditionally selects data for the SELECT command. The variable *search-condition* identifies one or more conditions to apply in selecting data.

ORDER BY *columnname* **(OPTIONAL)**
orders selected data by specific columns. Put the most significant column first. This column will be put in order first, the second will be ordered within the limits of the first ORDER BY column, and so on.

Ascending order is the default.

GROUP BY *columnname* **(OPTIONAL)**
is used to group selected table rows by matching values in more than one column. This keyword specifies the definition of the groups.

EXTRACT (REQUIRED)
is specified to extract data from a relational database.

WITH (OPTIONAL)
is included when your target is a DB2 or SQL/DS table. Use it to:

- Create a new DB2 or SQL/DS table (WITH CREATE)
- Replace all the data in an existing DB2 or SQL/DS table (WITH REPLACE)

Do not use the WITH keyword if you want to:

- Add your extracted data to an existing table
- Load your data into a physical sequential data set (refer to the EXTDATA keyword of the SUBMIT command)

When you write the WITH keyword, the new or replaced table that results has the name you specified on the INTO statement; the columns in the resulting table have the names you specified following the table name on the INTO statement. The data type of a target column depends on the data type of the source column. For more information about the data types of source columns, see your database administrator, or the DB2 or SQL/DS catalog tables.

The following table gives the input/output data types for a DB2 or SQL/DS table that is created.

If the DB2 or SQL data type is...	The target column data type is...
CHAR	CHAR (source length)
VARCHAR	VARCHAR (longest source occurrence)
LONG VARCHAR	LONG VARCHAR (SQL defined size)

If the DB2 or SQL data type is...	The target column data type is...
GRAPHIC	GRAPHIC (source length)
VARGRAPHIC	VARGRAPHIC (longest source occurrence)
LONG VARGRAPHIC	LONG VARGRAPHIC (SQL defined size)
FLOAT	FLOAT
INTEGER	INTEGER
SMALLINT	SMALLINT
DECIMAL	DECIMAL (precision,scale)
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP

Note: If a source data type is CHAR, VARCHAR, or LONG VARCHAR having bit data, its target column data type will be those data types with bit data.

If you are going to produce a control deck and need to know the target data types for different source field data types, see the previous data type tables.

Note: If you specify WITH CREATE, CD=JCS and DBS=DB2, the *TC marker must be in your JCS to tell DataRefresher where to write the CREATE TABLE command.

When you create a DB2 or SQL/DS data table in an extract request using WITH CREATE, you can determine whether its columns can contain null values with the absence or presence of NOT NULL on the INTO statement.

INTO *tableid (columnname(s))* **(OPTIONAL)**
identifies the names of the table and the table columns where the extracted data is to be inserted. INTO is needed only if the extracted data is to be loaded into a relational database table.

If an extract request contains the INTO statement the REM generates the load command necessary to load the extracted data into the table.

tableid
specifies the ID of the relational database table into which you want your data inserted.

tableid has the form: *authid.tablename*

authid is an alphameric/special characters name.

You do not have to include the *authid* (authorization ID). However, if you do not do so for DB2, the load utility assumes it is the same as the user ID written in the JOB statement of the JCS data set associated with your extract request. If no user ID was specified on the JOB statement, DB2 will use the default user ID for batch jobs that was defined when DB2 was installed.

If, in the case of SQL/DS, you do not specify an authorization ID, SQL/DS will use the user ID of the CONNECT statement in the JCS data set that is associated with your extract request.

If you do not write an authorization ID, do not write the period in front of *tablename*.

tablename must be written. It can be either an SQL name, an SQL quoted name, or a DBCS name.

columnname(s)
specify the name(s) of the column(s) into which you want the extracted data inserted.

For the maximum number of *columnnames*, see Appendix A, "DataRefresher limits" on page 241. A *columnname* can be either an SQL name, an SQL quoted name, or a DBCS name. If you write more than one column name, separate them with commas.

Specifying nulls

If one or more columns cannot contain null values, you must follow the column name with NOT NULL. If this is not coded, the REM assumes that the column may contain null data and will produce a NULLIF statement in the generated control statement for the column.

DB2 also allows a column to be created as NOT NULL WITH DEFAULT. When you code DBS=DB2, the REM will allow NOT NULL WITH DEFAULT for a given column definition on the INTO statement.

If a column is defined as NOT NULL WITH DEFAULT, the REM will include a DEFAULTIF keyword as part of the description of the relevant column when generating the DB2 LOAD control deck. This causes the DB2 LOAD utility to load zeros into numeric fields and blanks into character fields wherever REM null indicators are found.

Ordering columns

The order of your column names is important and depends on how you write the SELECT and INTO statements.

The order you write the field names on the SELECT statement will be the order in which they are inserted in the columns named on the INTO statement.

The number of columns you name on the INTO statement must be equal to the number of fields you name on the SELECT statement.

If you do not know the order in which the source columns are defined in their relational tables, you need to see your database administrator or query the relational database catalog tables.

Ordering columns when specifying SELECT *

Specifying * for the SELECT statement will select all the columns from a source table. Your target columns are matched with the columns contained in the relevant source table(s).

The number of columns you name on the INTO statement must be equal to the number of columns listed in the relevant source table(s).

Specifying columns for IXF

The IXF table record gets the value of IXFTNAME (name of data) from:

- INTO *tableid* if specified
- EXTID keyword of the SUBMIT command if INTO was not specified

The IXF column descriptor record gets the value of IXFCNAME (name of a column) from:

- INTO *columnname(s)* if specified
- source data names (0 otherwise)

IXFCNAML (length of name specified in IXFCNAME) is always 18.

NOT NULL WITH DEFAULT may not be specified for data in IXF.

Reasons for termination:

If the number of columns specified on the INTO statement does not equal the number of columns of extracted data, the request will be terminated.

If the source data can have null values and you are entering data into a table with NOT NULL or NULLS=N, the REM will give an error message and terminate.

OPTIONS (OUT (NONE|*n*)) (OPTIONAL)
affects how much output your extract request produces.

OUT(NONE | *n*) (REQUIRED if OPTIONS is specified)
controls the number of output rows the REM can generate when running your extract request.

(NONE)
indicates that there is no limit on the number of output rows that the REM can generate when executing your extract request.

(*n*) indicates the maximum number of output rows you want generated when the REM executes your extract request.

When this number is reached, the REM stops writing your output, even if your output is not yet complete.

Replace *n* with a number from 1 to 10000000, written without commas.

SELECT (REQUIRED)
identifies the field(s) to be extracted from a DB2 or SQL/DS database. It is coded the same as if it were to be submitted directly to the database. The SELECT statement is not syntax checked by DataRefresher; it is passed to DB2 or SQL/DS for error checking.

The following discussion of SELECT is a simplified version of the DB2 or SQL/DS SELECT statement.

ALL
specifies that duplicate values are to be selected. If ALL is specified, duplicate rows will not be eliminated from the result of a select.

DISTINCT
specifies that duplicate values are not to be selected. It may be used to eliminate duplicates from the SELECT result, or to eliminate duplicates from a built-in function.

*** (asterisk)**
selects all the columns from a table.

columnname(s)
lists the names of the column(s) from which you want to select data.

List the names of the columns you want from left to right. When specifying more than one column, separate the column names by commas.

For the maximum number of *columnnames*, see Appendix A, "DataRefresher limits" on page 241.

The *columnname* can be an SQL name, SQL quoted name, or a DBCS name.

expression
represents a calculated result. A selection can be made not only on data already in a table, but also on results that can be calculated by using that data.

A calculated result can consist of any combination of column names and constants connected by one or more arithmetic operators. A constant (sometimes called a literal) specifies a value. A constant can be a numeric value, hexadecimal data, or character data. Constants that contain character or hexadecimal data must be enclosed in single quotes.

FROM (REQUIRED)
identifies the table or view from which data is to be selected.

tablename
specifies the name of the table to be accessed.

viewname
specifies the name of the view.

More than one table or view may be selected at a time.

When you select data from two or more tables, DataRefresher extracts data from the specified columns in each table and combines it to form one output row. The WHERE statement compares a column value in one table with the other specified column value (EMPNO in DSN82.TEMPL) and combines the rows based upon this match.

WHERE *search-condition* **(OPTIONAL)**
conditionally selects data for the SELECT command. The variable *search-condition* identifies one or more conditions to apply in selecting data.

Replace *search-condition* with one or more conditions for eliminating unwanted row data. Parentheses can be used to group conditions.

ORDER BY *columnname* **(OPTIONAL)**
orders selected data by specific columns. Put the most significant column first. This column will be put in order first, the second will be ordered within the limits of the first ORDER BY column, and so on.

Ascending order is the default. If you want the rows presented in descending order, write

```
ORDER BY columnname DESC
```

GROUP BY *columnname* **(OPTIONAL)**
is used to group selected table rows by matching values in more than one column. This keyword specifies the definition of the groups.

Replace *columnname* with the name of one or more columns, separated by commas, to be used when forming a group.

Except for the grouping column (the column named with GROUP BY clause), any other column value that is selected must have an SQL built-in function specified.

For example,

```
SELECT WORKDEPT, AVG(SALARY)
  FROM DSN82.TEMPL
  GROUP BY WORKDEPT
  ORDER BY WORKDEPT;
```

gives this result:

```
WORKDEPT  AVG(SALARY)
========  ===============
A00       42833.333333333
B01       41250.000000000
C01       30156.666666666
D11       24667.777777777
D21       25153.333333333
E01       40175.000000000
E11       20998.000000000
E21       23827.500000000
```

The average salary is calculated for each department. The example's GROUP BY keyword specifies that you want the database manager to apply a function (in this case, AVG(SALARY)) to each group of columnname values (in this case, WORKDEPT) and return one row for each group of department numbers. WORKDEPT can be selected without an SQL built-in function because its value determines the group. Every member of each group has the same WORKDEPT value.

Examples of SUBMIT

This section contains:

- SUBMIT statement examples
- EXTRACT statement examples

The two statements of this command are both very detailed, so there are separate examples of each. You can use the statements to create your own SUBMIT command in its entirety.

Examples of the SUBMIT (REM) statement

Example 1

This example describes an extract request with the following characteristics:

- The extract request ID is EXTRACT1.
- The extracted output and JCS will be sent to the userid, USER1.
- A control deck will be generated for DB2 (DBS=DB2).
- The DD name of the input JCS data set is called JCSFILE1.
- The decimal point in the extract output will be a comma.
- The output format will be in EBCDIC.

- The RESUME(YES) keyword will be generated in the DB2 control deck for the LOAD utility.

```
SUBMIT EXTID=EXTRACT1,VMSPOOL=(USER1),
       DBS=DB2,CD=JCS,JCS=JCSFILE1,
       DECIMAL=COMMA,FORMAT=EBCDIC,
       USERDECK='RESUME(YES)'
```

Example 2

This example describes an extract request with the following characteristics:

- The extract request ID is EXTRACT2.
- A load control deck for SQL/DS will be generated.
- A CMS file is created for the extracted data having the name of FILE1, with a type of TEXT, and having a mode of A.
- The control deck is written in the CMS file generated.
- The DD name of the input JCS data set is JCSFILE2.
- The decimal point in the extract output will be a period, which is the default.
- The output format is in source format.
- The SQL/DS DATALOAD control deck has the LIST(NO) keyword generated. This indicates to SQL/DS that input data embedded within the control file is not to be displayed in the message file.

```
SUBMIT EXTID=EXTRACT2,DBS=SQLDS,
       EXTDATA='FILE1 TEXT A',
       CD=EXTDATA,JCS=JCSFILE2,FORMAT=SOURCE,
       USERDECK='LIST(NO)'
```

Example 3

This example describes an extract request with the following characteristics:

- The extract request ID is EXTRACT3.
- The extract output will be written in IXF format.
- A sequential file is created for the extracted data. The ddname FILE2 identifies a DD statement into which the output data will be written.
- The control deck is written in the sequential file generated.
- The output format is in EBCDIC format, which is the default.

```
SUBMIT EXTID=EXTRACT3,DBS=IXF,
       EXTDATA=FILE1,
       CD=EXTDATA,FORMAT=EBCDIC
```

Examples of the EXTRACT (REM) statement

Example 1

This example describes an extract request with the following characteristics:

- An DB2 table (DB2TAB1) is created because of the WITH CREATE keyword.
- The extracted data is inserted into the columns named (PART_NUMBER, DESCRIPTION, QUANTITY_ONHAND).
- All the columns are selected from the SQL/DS INVENTORY sample table (SELECT *).
- Only those part numbers that are greater than or equal to 221 are extracted (PARTNO > = 221).
- The data is ordered by part numbers (ORDER BY PARTNO) in ascending order (the default).

```
EXTRACT WITH CREATE INTO DB2TAB1
        (PART_NUMBER,DESCRIPTION,QUANTITY_ONHAND)
SELECT *
   FROM INVENTORY
   WHERE PARTNO >= 221
   ORDER BY PARTNO;
```

Example 2

This example describes an extract request with the following characteristics:

- The extracted data replaces data in an existing SQL/DS table (SQLDSTAB1) (WITH REPLACE).
- The columns SUPPNO, DELIVERY_TIME, and QONORDER are extracted from the SQL/DS QUOTATIONS sample table.
- The rows that are extracted are the rows where delivery time is greater than 15 (DELIVERY_TIME > 15).
- The rows are ordered by supplier number (SUPPNO) in ascending order (the default).

```
EXTRACT WITH REPLACE INTO SQLDSTAB1
        (SUPPLIER_NUMBER,DELIVERY_TIME,QUANTITY_ONORDER)
SELECT SUPPNO,DELIVERY_TIME,QONORDER
   FROM QUOTATIONS
   WHERE DELIVERY_TIME > 15
   ORDER BY SUPPNO;
```

Example 3

This example describes an extract request with the following characteristics:

- The target table is an SQL/DS table (SUPPLIER_TABLE) which will be created (WITH CREATE operand).
- The column names of the target table are SUPPLIER_NAME, SUPPLIER_ADDRESS, and SUPPLIER_NUMBER.
- Data is being extracted from the SQL/DS table named SUPPLIERS.
- Data is being extracted from the columns named NAME, ADDRESS, and SUPPNO. This is not the order the SUPPLIERS table is in, but will be the order of the target table.
- Only those suppliers whose number is greater than 60 are retrieved.

```
EXTRACT WITH CREATE INTO SUPPLIER_TABLE
        (SUPPLIER_NAME,SUPPLIER_ADDRESS,SUPPLIER_NUMBER)
SELECT NAME,ADDRESS,SUPPNO
  FROM SUPPLIERS
 WHERE SUPPNO > 60;
```

Example 4

This example describes an extract request with the following characteristics:

- The target table is an SQL/DS table named INVENTORY_TABLE. Because we did not specify the WITH CREATE option, the SQL/DS table must be an existing table with the column names defined.
- The column names of the target table are DESCRIPTION, PART_NUMBER, and QUANTITY_ONHAND.
- Data is being extracted from the SQL/DS table named INVENTORY.
- Data is being extracted from the columns named DESCRIPTION, PARTNO, and QONHAND. (This is not the order the INVENTORY table is in, but will be the order of the target table.)
- Only those parts with a description of BOLT are selected.

```
EXTRACT INTO INVENTORY_TABLE
        (DESCRIPTION,PART_NUMBER,QUANTITY_ONHAND)
SELECT DESCRIPTION,PARTNO,QONHAND
  FROM INVENTORY
 WHERE DESCRIPTION = 'BOLT';
```

Example 5

This example describes an extract request with the following characteristics

- The target table is a DB2 table named DB2EMPLOYEE_TABLE with the column names FIRST_NAME, LAST_NAME, and EMPLOYEE_NUMBER. This DB2 table must be an existing DB2 table with the column names defined.
- Data is being extracted from the DB2 table named DSN8.TEMPL.
- Data is being extracted from the columns named FIRSTNME, LASTNAME, and EMPNO. (This is not the order the DSN8.TEMPL table is in, but will be the order of the target table.)
- Data is being extracted for those employees whose number is greater than 250.
- The extracted data is in order of last name.

```
EXTRACT INTO DB2EMPLOYEE_TABLE
        (FIRST_NAME,LAST_NAME,EMPLOYEE_NUMBER)
SELECT FIRSTNME,LASTNAME,EMPNO
  FROM DSN8.TEMPL
 WHERE EMPNO > 250
 ORDER BY LASTNAME;
```

Example 6

This example describes an extract request with the following characteristics:

- The target table is an SQL/DS table (SQLDSTABLE) which is created (WITH CREATE operand).
- The column names of the target table are WORK_DEPARTMENT, FIRST_NAME, and LAST_NAME.
- Extracts data from the DB2 table named DSN8.TEMPL.
- Extracts data from the columns named WORKDEPT, FIRSTNME, and LASTNAME. (This is not the order the DSN8.TEMPL table is in, but will be the order of the target table.)
- Only employees in departments D21, E11, and E21 are selected.
- The extracted data results are ordered first by work department, then by last name.

```
EXTRACT WITH CREATE INTO SQLDSTABLE
        (WORK_DEPARTMENT,FIRST_NAME,LAST_NAME)
SELECT WORKDEPT,FIRSTNME,LASTNAME
  FROM DSN8.TEMPL
 WHERE WORKDEPT IN('D21','E11','E21')
 ORDER BY WORKDEPT,LASTNAME;
```

Chapter 11. Online DataRefresher commands

General-use programming interface

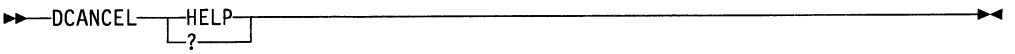
This chapter contains an alphabetic listing of all online DataRefresher commands. The online commands are TSO REXX EXECs that let you build and execute DataRefresher requests online (in the TSO foreground). This is an alternative way to run the UIM, DEM, and REM without using DataRefresher dialogs or JCL to submit the request.

For more information about using the online DataRefresher commands, see the *DataRefresher Administration Guide*.

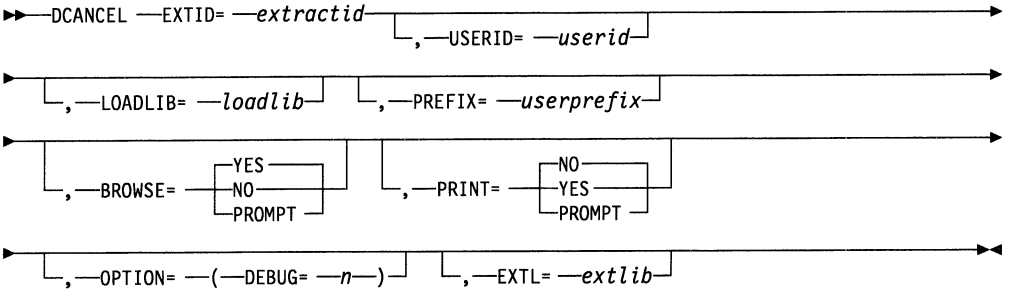
DCANCEL

Use the DCANCEL online command to cancel an extract request that has been submitted to the extract manager for execution. The DCANCEL command does this by issuing the CANCEL (UIM) command.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DCANCEL command is:



DCANCEL (REQUIRED)
cancels an extract request.

EXTID = *extractid* (REQUIRED)
specifies the name of the extract request you want to cancel.

Replace *extractid* with the ID of the relevant extract request that was specified on the SUBMIT command.

USERID=*userid* (OPTIONAL)
specifies the user ID of the extract request you want to cancel.

Replace *userid* with the value of the USERID keyword specified on the SUBMIT command. If the USERID keyword was not used on the SUBMIT command, do not include the USERID keyword on the DCANCEL command.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a fully qualified TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used. DVR110.DVRLOAD is the default shipped with DataRefresher. However your system administrator may have changed this to customize your needs.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all input/output data sets used by the Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

EXTL=*extlib* (OPTIONAL)
specifies the EXTLIB in which the extract request you are cancelling resides.

Examples of DCANCEL

Example 1

This example cancels an extract request.

Assume that:

- EXTREQ1 is the extract ID
- SMITH is the user ID of the extract request
- You want to print the DXTPRINT data set.

```
DCANCEL EXTID=EXTREQ1,USERID=SMITH,PRINT=YES
```

Example 2

This example cancels an extract request.

Assume that:

- EXTREQ1 is the extract ID
- The extract request does not have a user ID
- You want to have DCANCEL prompt for printing the DXTPRINT data set.

```
DCANCEL EXTID=EXTREQ1,PRINT=PROMPT
```

Note: Examples 1 and 2 do not cancel the same extract request because they refer to different user IDs.

DCREATE

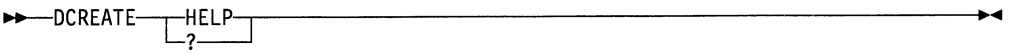
Use the DCREATE online command to create the DataRefresher data descriptions:

- DXTFILE
- DXTVIEW
- DXTPSB/DXTPCB
- DATATYPE

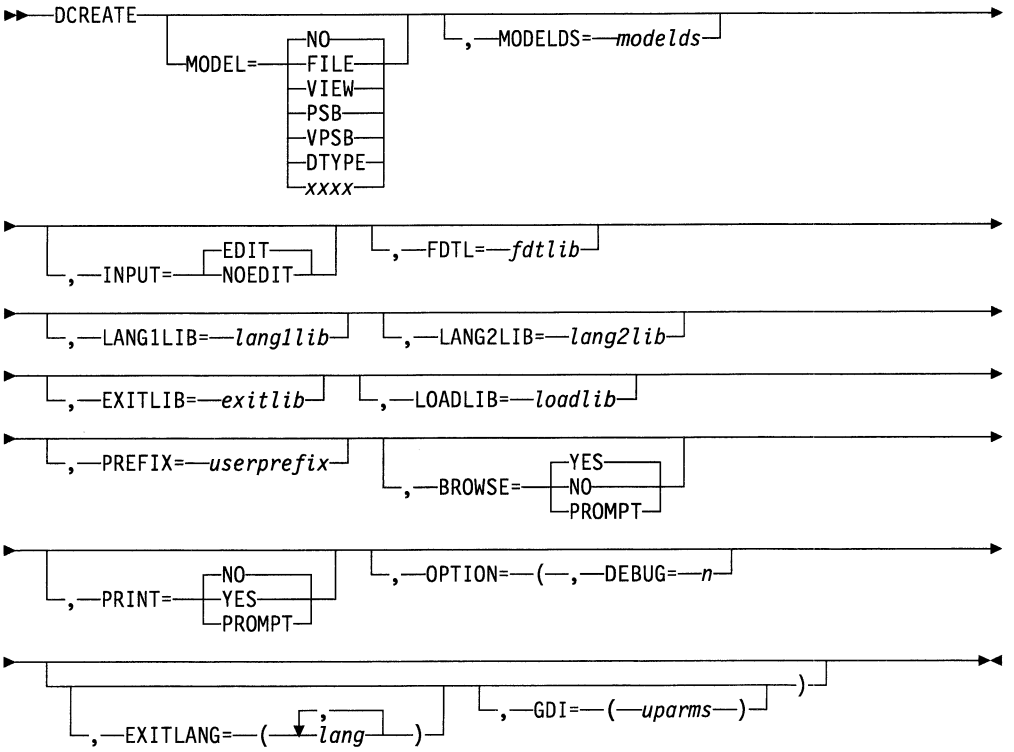
Using the DCREATE command causes the DXTIN data set to be displayed in ISPF EDIT mode so you can enter the DXTCREATE data descriptions. This data set is then used as input to the CREATE (UIM) command. The results can then be displayed at the terminal.

You will need to know how to write the data descriptions once you are in ISPF EDIT mode. See the individual CREATE (UIM) commands for more information about each type of data description.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DCREATE command is:



DCREATE (REQUIRED)
is used to create a data description.

MODEL=NO|FILE|VIEW|PSB|VPSB|DTYPE|*xxxx* **(OPTIONAL)**
specifies the model you want copied into the DXTIN data set.

The values you can specify are:

NO
means a model will not be copied into the DXTIN data set.

FILE
copies the DXTFILE create model DVREDREF into the DXTIN data set.

VIEW
copies the DXTFILE create model DVREDRVF into the DXTIN data set.

PSB
copies the DXTFILE create model DVREDREP into the DXTIN data set.

VPSB
copies the DXTFILE create model DVREDRVP into the DXTIN data set.

DTYPE
copies the DXTFILE create model DVREDCRD into the DXTIN data set.

xxxx
copies any member name specified into the DXTIN data set.

xxxx cannot be one of the reserved model names (FILE, VIEW, PSB, VPSB., DTYPE, or NO).

MODELDS=*modelds* **(OPTIONAL)**
defines the data set containing CREATE models referenced by the MODEL keyword.

INPUT=EDIT | NOEDIT (OPTIONAL)
indicates whether or not an ISPF EDIT screen of DXTIN is shown.

EDIT
means that an ISPF EDIT screen for the DXTIN data set is shown before it is used.

NOEDIT
means that the DXTIN data set will be used without showing it on the screen.

FDTL=*fdtlib* **(OPTIONAL)**
defines the FDTLIB to the UIM.

LANG1LIB=*langlib1* **(OPTIONAL)**
specifies the language library for *language1* of the EXITLANG keyword.

LANG2LIB=*langlib2* **(OPTIONAL)**
specifies the language library for *language2* of the EXITLANG keyword.

EXITLIB=*exitlib* **(OPTIONAL)**
specifies the library the exit is in, if it is not in the LOADLIB.

LOADLIB=*loadlib* **(OPTIONAL)**
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used. DVR110.DVRLOAD is the default shipped with the product, however your System Administrator may have changed this to customize your needs.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all input/output data sets used by Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

EXITLANG=(*language1, language2*) (OPTIONAL)
specifies the language environments that exit routines require.

language1, language2
specify the languages that the exit routines are written in.

There are three languages that are valid when using exits:

- PL/I
- COBOL
- ASSEMBLER

ASSEMBLER is always assumed, whether or not you include the EXITLANG keyword.

Replace *language1, language2* with PLI, COBOL, or both.

Enclosing parentheses for this operand are optional when specifying only one language.

If you do not know the language in which an exit routine is coded, see your System or Database Administrator.

GDI=(*userparms*)
is limited to a maximum of 94 characters and that value is passed, unmodified, to the user exit.

Examples of DCREATE

Example 1

This example creates a DXTFILE data description using all default DCREATE parameters.

```
input:   DCREATE

output:  EDIT screen ---
         ******  TOP OF DATA  ******
         ******  BOTTOM OF DATA  ***

input:   type in or copy CREATE command
         END (F3)

output:  ***  RETURN CODE FROM UIM = 0
         ***

input:   ENTER

output:  BROWSE screen ---
         ******  TOP OF DATA  ******
         DXTPRINT LISTING ---
         . . . . . .
         ******  BOTTOM OF DATA  ***

input:   END (F3)

output:  ***  ONLINE REQUEST COMPLETED  ***
         READY
```

Example 2

This example creates a DXTVIEW the unedited contents of the DXTIN. A user created input model called MYDATA is used. You are prompted for browsing the DXTPRINT data set.

```
input:   DCREATE INPUT=NO, MODEL=MYDATA, BROWSE=PROMPT

output:  ***  RETURN CODE FROM UIM = 0
         ===> BROWSE OUTPUT? (Y|N)

input:   N

output:  ***  ONLINE REQUEST COMPLETED  ***
         READY
```

DDELETE

Use the DDELETE online command to eliminate unwanted data descriptions from the FDTLIB. The DDELETE command issues the DELETE (UIM) command.

When you issue a DDELETE command for a DXTFILE or DXTPSB, the UIM automatically writes a message to the DXTPRINT data set listing any views that are based on the data description you are deleting. This way, your DXTFILEs and DXTPSBS can be easily cross-referenced with their corresponding DataRefresher views.

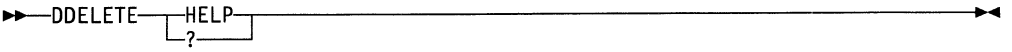
The DDELETE command is also an intermediate step in updating data descriptions. You can update DataRefresher:

- File descriptions
- PSB descriptions
- View descriptions
- User data type descriptions

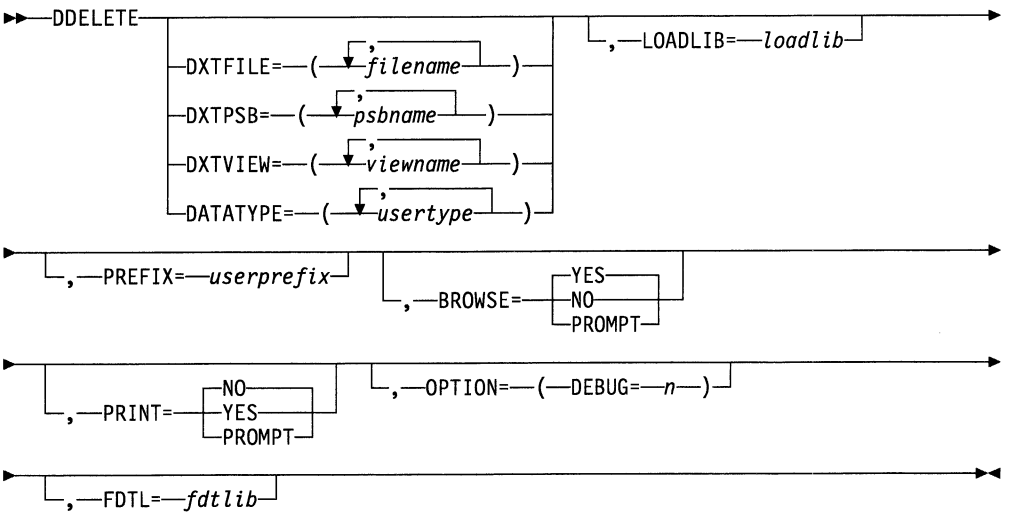
When updating a file, view, PSB, or user data type description, you must first delete the existing version of the data description using the DDELETE command and then create the updated version of the data description.

If you delete the DXTFILE data description of a generic data interface (GDI) select exit, DataRefresher automatically deletes the associated DXTVIEW.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DDELETE command is:



DDELETE (REQUIRED)
specifies one or more data descriptions you want to delete.

DXTFILE=(*filename*) (OPTIONAL)
identifies the DXTFILE description(s) to delete.

Replace *filename* with the names of the DXTFILE descriptions (you are

authorized to use) you want to delete, separating the names with commas. You can delete up to 16 DXTFILE descriptions in one DDELETE command.

If you are deleting only one DXTFILE description, you can omit the parentheses.

DXTPSB=(*psbname*) (OPTIONAL)
identifies the DXTPSB description(s) to delete.

Replace *psbname* with the names of the PSB descriptions you are authorized to use and want to delete, separating the names with commas. You can delete up to 16 DXTPSB descriptions in one DDELETE command.

If you are deleting only one DXTPSB description, you can omit the parentheses.

DXTVIEW=(*viewname*) (OPTIONAL)
identifies the DXTVIEW description(s) to delete.

Replace *viewname* with the names of the view descriptions you are authorized to use and want to delete, separating the names with commas. You can delete up to 16 DXTVIEW descriptions in one DDELETE command.

If you are deleting only one DXTVIEW description, you can omit the parentheses.

DATATYPE=(*usertype*) (OPTIONAL)
specifies the user data type(s) to delete.

Replace *usertype* with the names of the user data types, separated by commas. These names are the 2 character identifiers used on the SRCTYPE keyword of the CREATE DATATYPE command.

A maximum of 16 user data types can be deleted with one DDELETE command.

If you are deleting only one user data type, you can omit the parentheses.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used: DVR110.DVRLOAD is the default shipped with the product, however your system administrator may have changed this to customize your needs.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all input/output data sets used by Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

FDTL=*fdtlib* (OPTIONAL)
defines the FDTLIB data set to the UIM.

Examples of DDELETE

Example 1

This example deletes the DXTPSB descriptions named PSB1 and PSB2, and prints the DXTPRINT data set.

```
DDELETE DXTPSB=(PSB1,PSB2),PRINT=YES
```

Example 2

This example deletes only one DXTFILE description named FILE1 and does not print any output.

```
DDELETE DXTFILE=FILE1
```

DLIST

If you run the DLIST online command, the following output will be written to the DXTPRINT data set and displayed at the terminal:

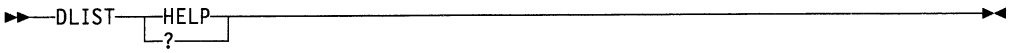
- ID of the extract request
- User ID of the extract request originator
- Name of the PSB from which the extract request retrieves data (if extracting data from an IMS database)
- Name of the file or PCB from which the extract request retrieves data
- Date that the extract request was submitted
- Time that the extract request was submitted
- Number of output rows specified with the OUT keyword of the extract request
- Priority number as specified with the PRI keyword of the extract request
- Disposition of the extract request

The DLIST command invokes the LIST (UIM) command.

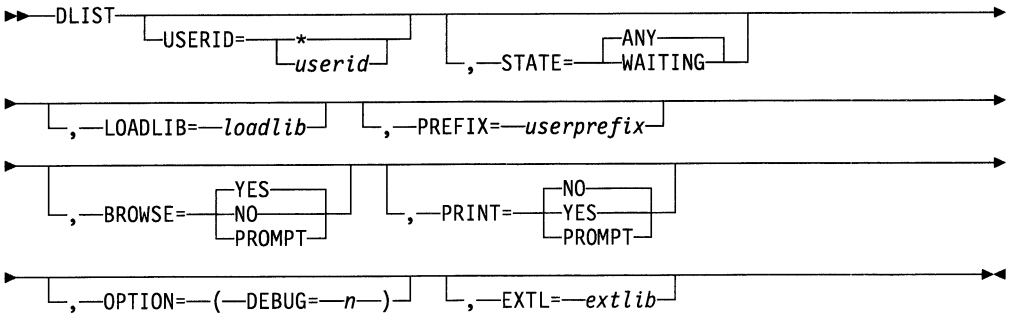
This information is useful in establishing DEM initialization parameters. The DLIST command can provide information on either all of the requests in EXTLIB, regardless of their source or status, or it can provide information on a defined subset of extract requests. These subsets can include only those extract requests:

- Of a named user
- Currently awaiting execution.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DLIST command is:



DLIST (REQUIRED)
writes information about DEM extract requests to the DXTPRINT data set and displays it on the terminal.

USERID=* | *userid* (OPTIONAL)
provides the ID of the user whose requests are to be listed. This user ID must match the user ID provided when the request was submitted. The values you can specify are:

*** (asterisk)**
means that all requests which satisfy the other parameters will be listed, regardless of their USERID values.

userid
is used to list extract requests for a specific user. Replace *userid* with the user ID written on the USERID keyword of the SUBMIT command for the extract request(s) you want to list.

If the USERID keyword is omitted, then all requests that do not have a USERID keyword specified with the extract request, and which satisfy the other parameters, will be listed.

STATE=ANY | WAITING (OPTIONAL)
helps to identify the extract requests to be listed. The values you can specify are:

ANY
means that then all extract requests in EXTLIB which satisfy the other parameters will be listed, regardless of their current status.

WAITING
means that only those extract requests currently awaiting execution and which satisfy the other parameters will be listed.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used; DVR110.DVRLOAD is the default shipped with the product, however, your system administrator may have customized this for your organization.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all data sets used by the DLIST command. If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

EXTL=*extlib* (OPTIONAL)
specifies the EXTLIB to the UIM.

Examples of DLIST

Example 1

The following examples show you two ways to list all extract requests in the EXTLIB.

```
DLIST
```

```
DLIST USERID=*, STATE=ANY
```

Example 2

This example lists all requests currently waiting for execution, and prints the output.

```
DLIST USERID=*, STATE=WAITING, PRINT=YES
```

This is how the DLIST command might be used to help in scheduling the DEM.

Example 3

This example lists only those requests identified with particular USERID value and use DataRefresher load library DXT.LOAD instead of the default load library.

```
DLIST USERID=SMITH, LOADLIB=DXT.LOAD
```

This is how users can check on their requests.

Example 4

This example lists only those requests identified with a particular user and currently waiting for execution.

```
DLIST USERID=USER1, STATE=WAITING
```

This is how users can find out which of their requests in EXTLIB have been validated and are waiting to be run.

Example 5

This example lists only those request identified with a particular user and currently waiting for execution. This time you do not want to display DXTPRINT, you just want to send it to be printed.

```
DLIST USERID=USER1, STATE=WAITING, BROWSE=NO, PRINT=YES
```

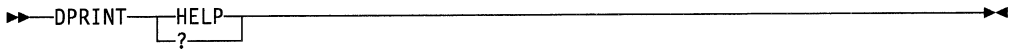
DPRINT

Use the DPRINT online command to display a data description on the terminal and print it in the DXTPRINT data set. The DPRINT command invokes the PRINT (UIM) command. You can use the printed results to examine the contents of the FDTLIB. A printed data description will not contain any comments that may have appeared with the data description when it was created.

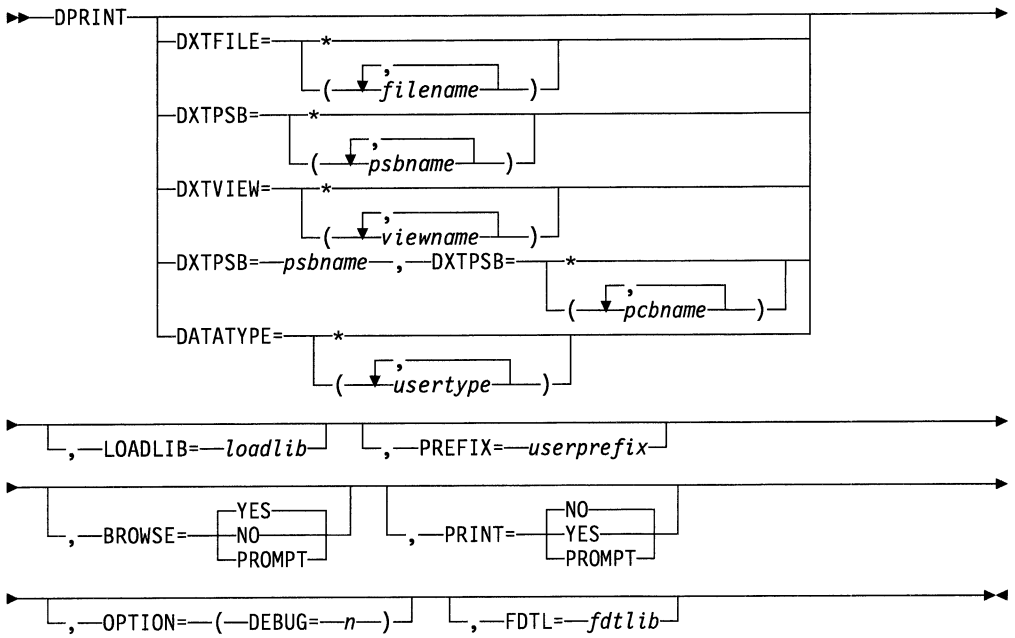
A PRINT (UIM) command for a DXTFILE or DXTPSB automatically issues a message indicating whether any DXTVIEWs exist that are based on the data description you are printing. If your DXTFILE or DXTPSB has one or more associated DXTVIEWs you are authorized to see, the message lists their names. This way, your DXTFILEs and DXTPSBS can be easily cross-referenced with their corresponding DXTVIEWs. The UIM writes these messages to the DXTPRINT data set at the same time it writes the PRINT command.

When the UIM executes a PRINT command for a GDI select file, it does not generate field level information.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DPRINT command is:



DPRINT (REQUIRED)
prints a data description.

DXTFILE=* | (*filename*) (OPTIONAL)
identifies the DXTFILE description(s) to print. The values you can specify are:

*** (asterisk)**
specifies every DXTFILE description you are authorized to use.

(*filename*)
specifies DXTFILE descriptions you want to print, separated by commas.

You can specify up to 16 DXTFILE descriptions in one DPRINT command.

If you are printing only one DXTFILE description, you can omit the parentheses.

DXTPSB=* | (*psbname*) (OPTIONAL)
identifies the DXTPSB description(s) to print. The values you can specify are:

*** (asterisk)**
specifies every DXTPSB description you are authorized to use.

(*psbname*)
specifies the DXTPSB descriptions you want to print, separated by commas.

You can specify up to 16 DXTPSB descriptions in one DPRINT command.

If you are printing only one DXTPSB description, you can omit the parentheses.

DXTVIEW=* | (*viewname*) (OPTIONAL)
identifies the DXTVIEW description(s) to print. The values you can specify are:

*** (asterisk)**
specifies every DXTVIEW description you are authorized to use.

(*viewname*)
specifies the DXTVIEW descriptions you want to print, separated by commas.

You can specify up to 16 DXTVIEW descriptions in one DPRINT command.

If you are printing only one DXTVIEW description, you can omit the parentheses.

DXTPSB=*psbname*, DXTPCB=* | (*pcbname*) (OPTIONAL)
identifies one or more DXTPCB descriptions within a specific DXTPSB description you want to print.

Replace *psbname* with the name of the DXTPSB description that contains the DXTPCB description(s) you want to print. Replace *pcbname* with either:

*** (asterisk)**
specifies every DXTPCB description you are authorized to use (in the DXTPSB description that you have specified).

(*pcbname*)
Replace *pcbname* with the names of the DXTPCB descriptions (in the DXTPSB description you specified) you want to print, separating the names with commas.

You can specify up to 16 DXTPCB descriptions for a given DXTPSB description.

If you are printing only one DXTPCB description, you can omit the parentheses.

DATATYPE=* | *usertype* (OPTIONAL)
prints a previously defined user data type.

*** (asterisk)**
specifies all data types defined to DataRefresher.

usertype
specifies the 2 character identifier specified on the SRCTYPE keyword of the CREATE DATATYPE command.

Up to 16 user data types can be printed with one DPRINT command (enclosed in parentheses and separated by commas).

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used: DVR110.DVRLOAD is the default shipped with the product, however your System Administrator may have customized this for your organization.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all input/output data sets used by Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

FDTL=*fdtlib* (OPTIONAL)
defines the FDTLIB data set to the UIM.

Examples of DPRINT

Example 1

This example prints the DXTFILE descriptions named FILE1 and FILE2 to the screen and causes a hard copy to be printed.

```
DPRINT DXTFILE=(FILE1,FILE2),PRINT=YES
```

Example 2

This example prints the DXTPCB descriptions PCBA, PCBB, and PCBC in the DXTPSB description named PSB1. It also changes the debug level to 2.

```
DPRINT DXTPSB=PSB1,DXTPCB=(PCBA,PCBB,PCBC),
       OPTION=(DEBUG=2)
```

Example 3

This example prints every DXTVIEW description in the input FDTLIB that you are authorized to use. It uses the DataRefresher load library called DXT.LOAD instead of the default program load library.

```
DPRINT DXTVIEW=*, LOADLIB=DXT.LOAD
```

DPUNCH

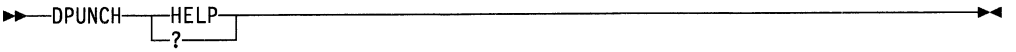
Use the DPUNCH online command to put a data description in the DXTPUNCH and DXTPRINT data sets, and display it on the terminal. The DPUNCH command invokes the PUNCH (UIM) command.

You can use the punched results to examine the contents of the FDTLIB and to update a data description.

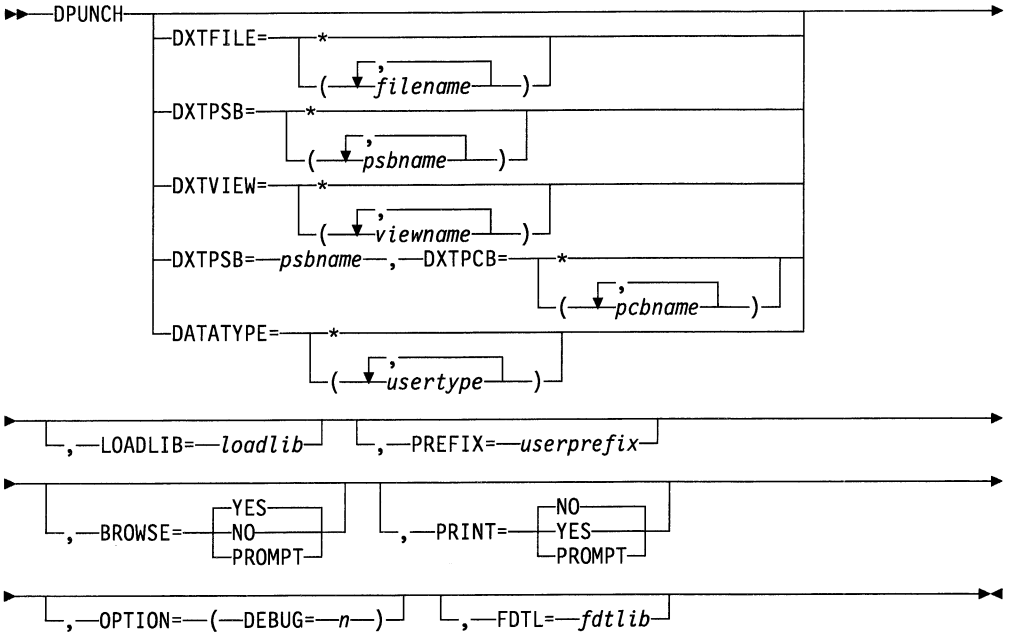
A punched data description will not contain any comments that may have appeared with the data description when it was created.

When the UIM executes a PUNCH command for a GDI select file, it does not generate field level information.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DPUNCH command is:



DPUNCH (REQUIRED)
punches a data description of your choice.

DXTFILE=* | (*filename*) (OPTIONAL)
identifies the DXTFILE description(s) to punch. The values you can specify are:

*** (asterisk)**
specifies every DXTFILE description you are authorized to use.

(*filename*)
specifies the DXTFILE descriptions you want to punch, separated by commas.

You can specify up to 16 DXTFILE descriptions in one DPUNCH command.

If you are punching only one DXTFILE description, you can omit the parentheses.

DXTPSB=* | (*psbname*) (OPTIONAL)
identifies the DXTPSB description(s) to punch. The values you can specify are:

*** (asterisk)**
specifies every DXTPSB description you are authorized to use.

(*psbname*)
specifies DXTPSB descriptions you want to punch, separated by commas.

You can specify up to 16 DXTPSB descriptions in one DPUNCH command.

If you are punching only one DXTPSB description, you can omit the parentheses.

DXTVIEW=* | (*viewname*) (OPTIONAL)
identifies the DXTVIEW description(s) to punch. The values you can specify are:

*** (asterisk)**
specifies every DXTVIEW description you are authorized to use.

(*viewname*)
specifies DXTVIEW descriptions you want to punch, separated by commas.

You can specify up to 16 DXTVIEW descriptions in one DPUNCH command.

If you are punching only one DXTVIEW description, you can omit the parentheses.

DXTPSB=*psbname*, DXTPCB=* | (*pcbname*) (OPTIONAL)
identifies one or more DXTPCB descriptions within a specific DXTPSB description you want to punch.

Replace *psbname* with the name of the DXTPSB description that contains the DXTPCB description(s) you want to punch. Replace *pcbname* with either:

*** (asterisk)**
specifies every DXTPCB description you are authorized to use (in the DXTPSB description that you have specified).

(*pcbname*)
Replace *pcbname* with the names of the DXTPCB descriptions (in the DXTPSB description you specified) you want to punch, separating the names with commas.

You can specify up to 16 DXTPCB descriptions for a given DXTPSB description.

If you are punching only one DXTPCB description, you can omit the parentheses.

DATATYPE=* | *usertype* (OPTIONAL)
punches a previously defined user data type.

*** (asterisk)**
specifies all data types defined to DataRefresher.

usertype
specifies the 2 character identifier specified on the SRCTYPE keyword of the CREATE DATATYPE command.

Up to 16 user data types can be punched with one DPUNCH command (enclosed in parentheses and separated by commas).

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used: DVR110.DVRLOAD is the default shipped with the product, however your System Administrator might have customized this for your organization.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all input/output data sets used by Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPUNCH output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPUNCH data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

FDTL=*fdtlib* (OPTIONAL)
defines the FDTLIB data set to the UIM.

Examples of DPUNCH

Example 1

This example punches the DXTFILE descriptions named FILE1 and FILE2, and prints the DXTPUNCH data set.

```
DPUNCH DXTFILE=(FILE1,FILE2),PRINT=YES
```

Example 2

This example punches the DXTPCB descriptions PCBA, PCBB, and PCBC in the DXTPSB description named PSB1. It also changes the debug level to 2.

```
DPUNCH DXTPSB=PSB1,DXTPCB=(PCBA,PCBB,PCBC),
       OPTION=(DEBUG=2)
```

Example 3

This example punches every DXTVIEW description in your input FDTLIB that you are authorized to use. It uses the DataRefresher load library called DXT.LOAD instead of the default program load library.

```
DPUNCH DXTVIEW=*,LOADLIB=DXT.LOAD
```

DRUN

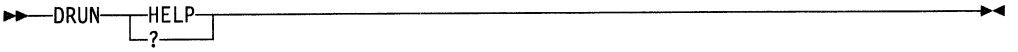
Use the DRUN online command to run the DEM online to process an extract request. Only one extract request can be processed with each DRUN command.

The extract is chosen by a request ID composed of the EXTID, USERID from the SUBMIT command. The command prompts for the data sets needed to run the extract. The DEM is run and the return code is displayed. The data set containing the DEM messages (DXTPRINT) can then be browsed.

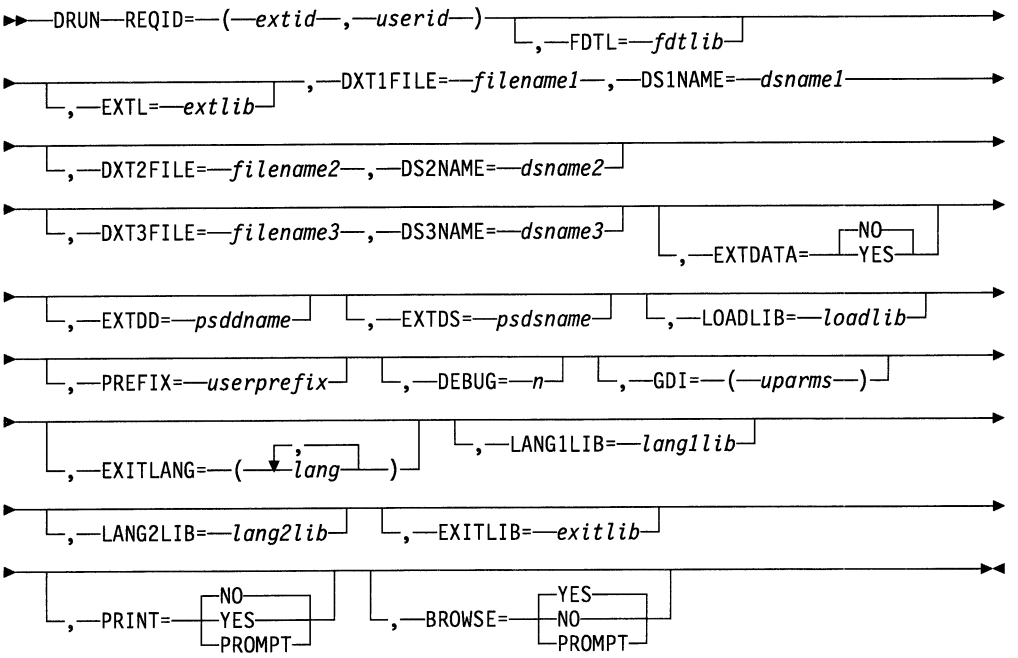
If you used the ORDER BY keyword in the extract request, any diagnostic messages issued by DFSORT* can be viewed in the DXTDIAG data set.

If JCS is used, the JCS data set (DXTOUT) is displayed with the extracted data. The JCS job can then be submitted for execution or saved for later, and its output can be browsed.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DRUN command is:



DRUN (REQUIRED)
is used to run the DEM

REQID=(*extid,userid*) (REQUIRED)
specifies the extract to be run by the DEM. For *extid and userid* use the value specified on the EXTID and USERID, keywords of the SUBMIT (UIM) command. You must specify *extid*. If you do not include *userid* then you do not need parentheses or a comma.

For example,

```
EXT1
(EXT2,USER2)
```

are all correct values of the REQID keyword.

FDTL=*fdtlib* (OPTIONAL)
defines the FDTLIB data set to the UIM.

EXTL=*extlib* (OPTIONAL)
defines the EXTLIB data set to the UIM.

DXT1FILE=*filename1* (REQUIRED)
specifies the DXTFILE description used to describe the data you are extracting from. DRUN allows you to extract from up to three data sets, each one requiring a DXTFILE name and a data set name. You must specify at least one data set.

DXT2FILE=*filename2* (OPTIONAL)
specifies a second DXTFILE description used to describe the data you are extracting from.

DXT3FILE=*filename3* (OPTIONAL)
specifies a third DXTFILE description used to describe the data you are extracting from.

DS1NAME=*dsname1* (REQUIRED)
specifies the data set name corresponding to DXT1FILE. This data set name contains the data to be extracted.

DS2NAME=*dsname2* (REQUIRED if DXT2FILE is specified)
specifies the data set name corresponding to DXT2FILE. This data set name contains the data to be extracted.

DS3NAME=*dsname3* (REQUIRED if DXT3FILE is specified)
specifies the data set name corresponding to DXT3FILE. This data set name contains the data to be extracted.

EXTDATA=NO | YES (OPTIONAL)
is specified if you are using the EXTDATA on the SUBMIT command. Only specify EXTDATA=YES if EXTDATA=*ddname* was coded on the SUBMIT command.

Using the EXTDATA on the SUBMIT command allows you to identify a physical sequential data set to which DataRefresher will write the output.

EXTDD=*psddname* (REQUIRED if EXTDATA=YES is specified)
specifies the DD name of the physical sequential data set to which DataRefresher writes the output. This is the DD name that you coded on the EXTDATA keyword of the SUBMIT command.

EXTDS=*psdsname* (REQUIRED if EXTDATA=YES is specified)
specifies the data set name of the physical sequential data set to which DataRefresher writes the output.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used: DVR110.DVRLOAD is the default shipped with the product, however your System Administrator might have customized this for your organization.

PREFIX=*userprefix* **(OPTIONAL)**
specifies the prefix of the high level qualifier for all input/output data sets used by Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

DEBUG=*n* **(OPTIONAL)**
specifies a debug level of 1, 2, 3, or 4. The default level is 1. For more information about debug levels, see Appendix E, "Diagnostic information" on page 257.

GDI=(*user parms***) (OPTIONAL)**
passes any parameters you want to the GDI exit while running the DEM.

(*user parms***)**
is limited to a maximum of 94 characters; this value is passed, unmodified, to the user exit.

EXITLANG=(*language1,language2***) (OPTIONAL)**
specifies the language environments that exit routines require.

language1,language2
specify the languages in which the exit routines are written.

There are three languages that are valid when using exits:

- PL/I
- COBOL
- ASSEMBLER

ASSEMBLER is always assumed, whether or not you include the EXITLANG keyword.

Replace *language1, language2* with PLI, COBOL, or both.

Enclosing parentheses for this operand are optional when specifying only one language.

If you do not know the language in which an exit routine is coded, see your System or Database Administrator.

LANG1LIB=*langlib1* **(OPTIONAL)**
specifies the language library for *language1* of the EXITLANG keyword.

LANG2LIB=*langlib2* **(OPTIONAL)**
specifies the language library for *language2* of the EXITLANG keyword.

EXITLIB=exitlib (OPTIONAL)
specifies the library the exit is in, if it is not in the LOADLIB.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

Example of DRUN

This example runs the DEM and provides the following information:

- The *extid* is EXT1. The *userid* is not specified.
- The file description that describes the data you are extracting from is called FILE1.
- The data set name corresponding to FILE1 is DXTDXN1.

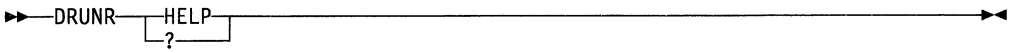
```
input:   DRUN REQID=EXT1,DXT1FILE=FILE1,DS1NAME=DXTDSN1
output:  ***  RETURN CODE FROM DEM = 0
         ===> BROWSE DEM OUTPUT? (Y|N):
input:   Y
output:  BROWSE screen ---
         ******  TOP OF DATA  ******
         DXTPRINT LISTING ---
         . . . . . .
         ******  BOTTOM OF DATA  ***
input:   END (F3)
output:  EDIT screen ---
         ******  TOP OF DATA  ******
         JCS with extracted data
         . . . . . .
         ******  BOTTOM OF DATA  ***
input:   END (F3)
output:  ===> SUBMIT JCS FOR EXECUTION? (Y|N)
input:   Y
output:  ***  ONLINE REQUEST COMPLETED  ***
         READY
```

DRUNR

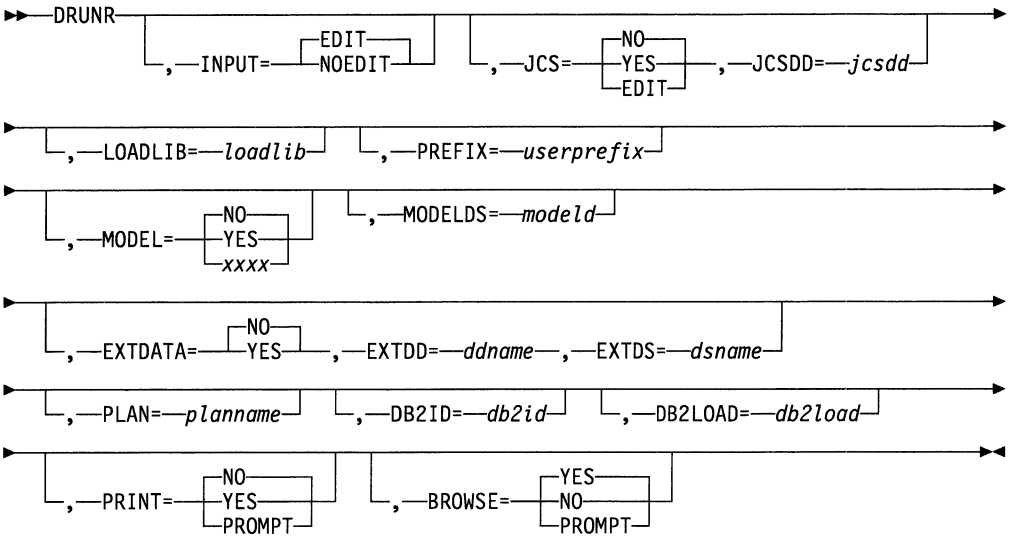
Use the DRUNR online command to run the REM online to process a relational extract request. The command can display the REM input data set in EDIT mode, where SUBMIT and EXTRACT commands can be copied or entered.

If JCS is required, the JCS input data set is displayed for editing. The REM is run and the output (DXTPRINT) can be browsed. If JCS is used, the JCS output with extracted data is next displayed. The JCS job can then be submitted for execution or saved.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DRUNR command is:



DRUNR (REQUIRED)
runs the REM.

INPUT=EDIT | NOEDIT (OPTIONAL)
specifies whether or not to show the EDIT panel of DXTIN.

JCS=NO | YES | EDIT (OPTIONAL)
specifies whether or not JCS will be included.

NO
means that JCS will not be used.

YES
means that JCS will be used, but not edited.

EDIT
means that JCS will be used and edited.

JCSDD=*jcsdd* (REQUIRED if JCS=YES | EDIT is specified)
specifies the DD name of the JCS input that is coded on the EXTRACT statement in the SUBMIT command. If JCSDD is not specified, the default JCSDD is assumed.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used: DVR110.DVRLOAD is the default shipped with the product, however your System Administrator might have customized this for your organization.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all input/output data sets used by the Online DataRefresher commands (for example, DXTPRINT, DXTPUNCH, DXTINOUT). If you do not enter a PREFIX value, it will default to your TSO user ID.

MODEL=NO | YES | *xxxx* (OPTIONAL)
indicates whether the REM model (DVREDEXT) or a user specified model (*xxxx*) is copied into the DXTIN data set from the model data set.

MODELDS=*modelds* (OPTIONAL)
specifies a model data set other than the default model data set. Use this when MODEL=YES or MODEL=*xxxx*.

EXTDATA=NO | YES (OPTIONAL)
indicates whether or not the EXTDATA keyword on the SUBMIT command is coded. Using the EXTDATA on the SUBMIT command allows you to identify a physical sequential data set to which DataRefresher will write the output.

EXTDD=*psddname* (REQUIRED if EXTDATA=YES is specified)
specifies the DD name of the physical sequential data set to which DataRefresher writes the output. If EXTDD is not specified, the default EXTDD is assumed.

EXTDS=*psdsname* (REQUIRED if EXTDATA=YES is specified)
specifies the data set name of the physical sequential data set to which DataRefresher writes the output. If EXTDS is not specified, the default EXTDS is assumed.

PLAN=*planname* (OPTIONAL)
specifies the DB2 plan created when DataRefresher was installed.

DB2ID=*db2id* (OPTIONAL)
specifies the DB2 subsystem ID where your plan and data are stored.

DB2LOAD=*db2load* (OPTIONAL)
specifies the library where the executable DB2 code resides.

For more information about these DB2 keywords, contact your DB2 administrator.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to print the DXTPRINT output data set.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

Example of DRUNR

This example runs the REM, accepting all defaults:

```
input:   DRUNR
output:  EDIT screen ---  (EXTRACT input)
         ******  TOP OF DATA  ******
         ******  BOTTOM OF DATA  ***
input:   type in or copy SUBMIT and EXTRACT commands
         for relational processing
         END (F3)
output:  EDIT screen ---  (JCS input)
         ******  TOP OF DATA  ******
         ******  BOTTOM OF DATA  ***
input:   type in or copy JCS
         END (F3)
output:  ***  REM RETURN CODE = 0
         ===> BROWSE THE OUTPUT? (Y|N)
input:   Y
output:  BROWSE screen ---
         ******  TOP OF DATA  ******
         DXTPRINT LISTING ---
         . . . . . .
         ******  BOTTOM OF DATA  ***
input:   END (F3)
output:  BROWSE screen ---
         ******  TOP OF DATA  ******
         JCS with extracted data
         . . . . . .
         ******  BOTTOM OF DATA  ***
input:   END (F3)
output:  ===> SUBMIT JCS FOR EXECUTION? (Y|N)
input:   Y
output:  ***  ONLINE REQUEST COMPLETED  ***
         READY
```

DSEND

Use the DSEND online command to build an extract request and send it for execution.

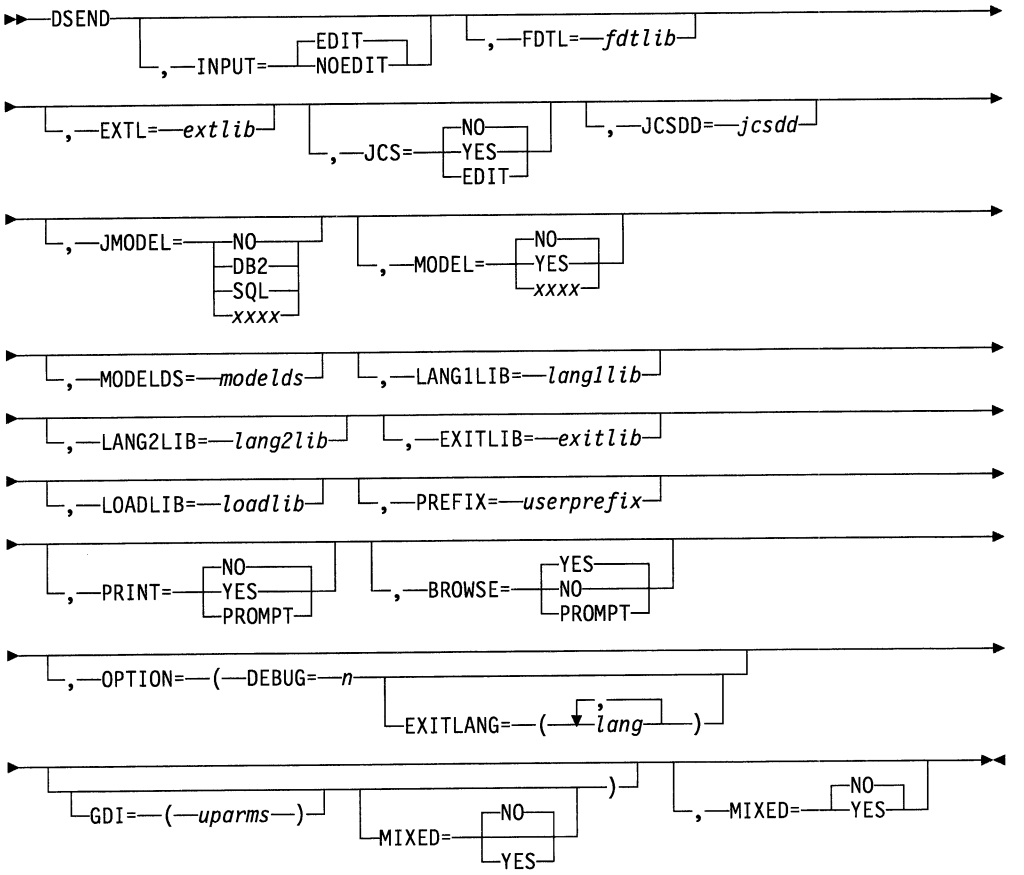
This command displays the EDIT screen for the UIM input data set. You can type in the SUBMIT/ EXTRACT command to perform an extract.

If JCS is to be used, a JCS input EDIT screen is displayed. You can type in the JCS statements.

The UIM then verifies and stores the extract request in the EXTLIB. The output (DXTPRINT data set) is displayed on the terminal and can also be printed.

If you would like to see a brief explanation of this command online, the syntax is:

The complete syntax of the DSEND command is:



DSEND (REQUIRED)
builds an extract request and sends it for execution.

INPUT=EDIT | NOEDIT (OPTIONAL)
indicates whether or not an ISPF EDIT screen of DXTIN is shown.

EDIT
means that an ISPF EDIT screen for the DXTIN data set is shown before it is used.

NOEDIT
means that the DXTIN data set will be used without showing it on the screen.

FDTL=*fdtlib* (OPTIONAL)
defines the data set containing the FDTLIB to the UIM.

EXTL=*extlib* (OPTIONAL)
defines the data set containing the EXTLIB to the UIM.

JCS=NO | YES | EDIT (OPTIONAL)
specifies whether or not the JCS input data set is to be included.

NO
means that JCS will not be used.

YES
means that JCS will be used, but not edited.

EDIT
means that JCS will be used and edited.

JCSDD=*jcsdd* (OPTIONAL)
specifies the DD name of the JCS input data set. This is the name used on the JCS keyword of the SUBMIT command.

JMODEL=NO | DB2 | SQL | *xxxx* (OPTIONAL)
indicates that no model is copied in (NO), or names a model that contains JCS and must be a member of the MODELDS data set.

NO
specifies that no model is copied in.

DB2
specifies a DB2 model for the JCS.

SQL
specifies an SQL model for the JCS.

XXXX
specifies a user model for the JCS. The user model cannot be named SQL, DB2, or NO.

MODEL=NO | YES | *xxxx* (OPTIONAL)
indicates whether or not the UIM/DEM model (NO | YES) or a user model (*xxxx*) is copied into the DXTIN data set.

MODELDS=*modelds* (OPTIONAL)
is used to override the default model data set. This keyword applies when MODEL=YES | *xxxx*, or if JMODEL=DB2 | SQL | *xxxx*. The default model data set is DVR110.DVRJEDIE.

LANG1LIB=*langlib1* (OPTIONAL)
specifies the language library for *language1* of the EXITLANG keyword.

LANG2LIB=*langlib2* (OPTIONAL)
specifies the language library for *language2* of the EXITLANG keyword.

EXITLIB=*exitlib* (OPTIONAL)
specifies the library the exit is in, if it is not in the LOADLIB.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used. DVR110.DVRLOAD is the default shipped with the product, however your System Administrator might have customized this for your organization.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all data sets used by the DSEND command. If you do not enter a PREFIX value, it will default to your TSO user ID.

PRINT=NO | YES | PROMPT (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether you want to browse the DXTPRINT data set that resulted from this command.

YES
presents the DXTPRINT data set for viewing.

NO
does not present the DXTPRINT data set for viewing.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

EXITLANG=(*language1,language2*)
specifies the language environments that exit routines require.

language1,language2
specify the languages in which the exit routines are written.

There are three languages that are valid when using exits:

- PL/I
- COBOL
- ASSEMBLER

ASSEMBLER is always assumed, whether or not you include the EXITLANG keyword.

Replace *language1, language2* with PLI, COBOL, or both.

Enclosing parentheses for this keyword is optional when specifying only one language.

If you do not know the language in which an exit routine is coded, see your System or Database Administrator.

GDI=(*userparms*)
is limited to a maximum of 94 characters; this value is passed, unmodified, to the user exit.

MIXED=NO | YES (OPTIONAL)
specifies that you have values on the WHERE keyword of the SUBMIT command that contain a mixed string of DBCS and single byte characters. If MIXED=NO, all quoted strings will be regarded as single byte character strings and the shift-out and shift-in characters will have no special meaning.

MIXED=NO | YES (OPTIONAL)
specifies that you have values on the WHERE keyword of the SUBMIT command that contain a mixed string of DBCS and single byte characters. If MIXED=NO, all quoted strings will be regarded as single byte character strings and the shift-out and shift-in characters will have no special meaning.

Examples of DSEND

Example 1

This example sends an extract request to the EXTLIB. The SUBMIT commands are already in the DXTIN data set, and the JCS input is to be used but not edited. You do not want to browse the output.

```
input:   DSEND INPUT=NOEDIT, JCS=YES, BROWSE=NO
output:  ***  RETURN CODE FROM UIM=0
input:   ENTER
output:  ***  ONLINE REQUEST COMPLETED  ***
         READY
```

Example 2

This example sends an extract request to the EXTLIB and displays one EDIT screen for copying in SUBMIT command and one for JCS input. The JCSDD default is overridden with MYJCS.

```
input:   DSEND JCS=EDIT,JCSDD=MYJCS
output:  EDIT screen ---  (DXTIN)
         ******   TOP OF DATA  ******
         ******   BOTTOM OF DATA  ***
input:   type in or copy the SUBMIT/EXTRACT commands
         END (F3)
output:  EDIT screen ---   (JCS input)
         ******   TOP OF DATA  ******
         ******   BOTTOM OF DATA  ***
input:   type in or copy the JCS statements
output:  ***  RETURN CODE FROM UIM = 0
         ***
input:   ENTER
output:  BROWSE SCREEN ---
         ******  TOP OF DATA  ******
         DXTPRINT LISTING ---
         . . . . . .
         ******  BOTTOM OF DATA  ***
input:   END (F3)
output:  *** ONLINE REQUEST COMPLETED ***
```

Example 3

This example sends an extract request to the EXTLIB. Assume:

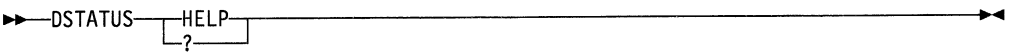
- No edit of the JCS is needed.
- The DD name of the JCS data set as coded on the SUBMIT command is DB2JCS
- The member name of the JCS model is MYJCS
- The SUBMIT/EXTRACT model member name is MYEXT1
- No edit of the input DXTIN data set is needed
- There is a prompt to browse the DXTPRINT data set

```
input:   DSEND JCS=YES, JCSDD=DB2JCS, JMODEL=MYJCS,
               MODEL=MYEXT1, INPUT=NOEDIT, BROWSE=PROMPT
output:  ***  RETURN CODE FROM UIM=0
         ===> BROWSE OUTPUT? (Y|N)
input:   N
output:  ***  ONLINE REQUEST COMPLETED  ***
         READY
```

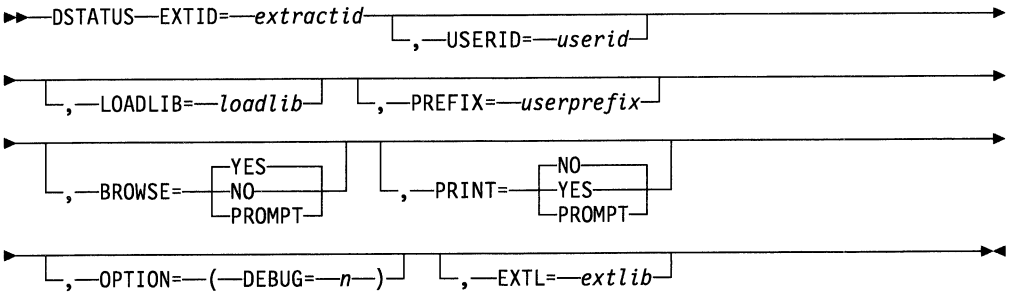
DSTATUS

Use the DSTATUS online command to check the status of an extract request and display the results at the terminal. The DSTATUS command issues the STATUS (UIM) command.

If you would like to see a brief explanation of this command online, the syntax is:



The complete syntax of the DSTATUS command is:



DSTATUS (REQUIRED)
checks the status of a DEM extract request.

EXTID=*extractid* (REQUIRED)
identifies the ID of the extract request whose status you are checking.

Replace *extractid* with the ID of the relevant extract request.

extractid is an SQL name.

USERID=*userid* (OPTIONAL)
identifies the user ID of the extract request whose status you are checking.

Replace *userid* with the value of the USERID keyword of the SUBMIT command for the extract request you are checking. If the USERID keyword was not included in the SUBMIT command, do not write it in your STATUS command.

userid is an alphameric/special characters name.

LOADLIB=*loadlib* (OPTIONAL)
designates the DataRefresher load library to be used for this online program.

loadlib
specifies a TSO data set name corresponding to a DataRefresher load library. Omitting this keyword results in the default loadlib being used; DVR110.DVRLOAD is the default shipped with the product. However, your System Administrator might have customized this for your organization.

PREFIX=*userprefix* (OPTIONAL)
specifies the prefix of the high level qualifier for all data sets used by the DCREATE command. If you do not enter a PREFIX value, it will default to your TSO user ID.

BROWSE=YES | NO | PROMPT (OPTIONAL)
specifies whether or not you want to browse the DXTPRINT data set that resulted from this command.

YES
displays the DXTPRINT data set.

NO
does not display the DXTPRINT data set.

PROMPT
generates a prompt message that asks whether you want to browse the DXTPRINT data set.

PRINT=NO | YES (OPTIONAL)
specifies whether or not to send the DXTPRINT output data set to be printed.

NO
suppresses printing.

YES
causes printing.

PROMPT
generates a prompt message that asks whether you want to print the DXTPRINT data set.

OPTION=(DEBUG=*n*) (OPTIONAL)
specifies the debug level. The default value is 1. For more information about debug levels see Appendix E, "Diagnostic information" on page 257.

EXTL=*extlib* (OPTIONAL)
defines the data set containing the EXTLIB to the UIM.

Examples of DSTATUS

Example 1

This example checks the status of an extract request with the extract ID of EXTREQ1, and user ID of SMITH. The DXTPRINT data set is printed.

```
DSTATUS EXTID=EXTREQ1,USERID=SMITH,PRINT=YES
```

Example 2

This example checks the status of an extract request whose extract ID is EXTREQ1, and which does *not* have a user ID. You are prompted for printing the DXTPRINT data set.

```
DSTATUS EXTID=EXTREQ1,PRINT=PROMPT
```

Note: Because the two examples of DSTATUS commands have different user IDS, they would not check the status of the same extract request.

End of General-use programming interface

Chapter 12. Administrative Dialogs commands

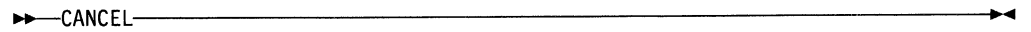
This chapter contains an alphabetical listing of DataRefresher Administrative Dialogs commands. These commands are designed to help the administrator perform several tasks, such as creating and maintaining:

- Data descriptions
- Extract requests
- JCL/JCS.

For more information about using the Administrative Dialogs, see the *DataRefresher Administration Guide* and the *DataRefresher MVS and VM User's Guide.*

CANCEL

Use the CANCEL command to end the current dialog without saving any changes.



CANCEL (REQUIRED)
ends the current dialog and returns you to the menu from which the dialog was initiated.

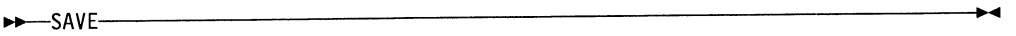
There are two instances of the CANCEL command that require special attention:

- If you are in the ISPF editor and you issue the CANCEL command, the information you have entered up to that point will be undone, but the build/update session will continue.
- When you are updating requests and you have asked for a JCL review in your profile, a CANCEL on the JCL review panel will not undo updates to the edited file, even though all other updated information is undone (for example, the request description).

SAVE

Use the SAVE command to interrupt an incomplete building or updating session without losing any of the information you have entered up to that point.

This command is available to the Build/Update and Profile dialogs only. Using the SAVE command for Send, Import/Export, Status, Cancel, Print, Punch, Erase, Delete, or DAP Dialogs has no effect.



SAVE (REQUIRED)
saves information you entered during a build or update session.

For example, you may interrupt the Profile Dialog and save the information as is, without completing all the associated panels. This gives you the capability of saving a partial dialog. You will be returned to the menu from which you initiated the building or updating dialog.

If you are in the ISPF editor and you type SAVE, the information you have entered up to that point will be saved, but you will remain where you are in the editor.

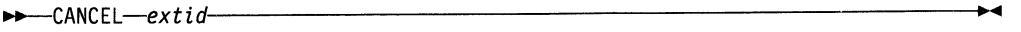
Chapter 13. End User Dialogs commands

This chapter contains an alphabetical listing of all End User Dialogs commands. These commands are intended to help the less experienced DataRefresher user perform data extracts.

For more information about using the End User Dialogs, see the *DataRefresher Administration Guide* and the *DataRefresher MVS and VM User's Guide.*

CANCEL

Use the CANCEL command to create a job to cancel an extract request (using the CANCEL (UIM) command) that has been submitted to the DEM for execution by the End User Dialogs. The CANCEL command in the End User Dialogs applies to DEM extract requests in EXTLIB only.



CANCEL (REQUIRED)
cancels an extract request that has been submitted to the DEM for execution.

extid
identifies the extract request to be cancelled.

Replace *extid* with the relevant extract ID. The extract ID is an 18-character system-generated identifier automatically assigned to the request by End User Dialogs when it is sent to the UIM for processing. It has the form EXTID_*yyddd_hhmmss*, where *yyddd* is the Julian date and *hhmmss* the time when the extract request was sent for execution.

The extract ID is used to distinguish between multiple submissions of the same request name, permitting the user to selectively choose the extract to be cancelled.

If *extid* is not specified, a list of extract requests that have been sent to the UIM in the last seven days will be displayed for selection. Each item in the list is associated with a name that is the same as the NAME specified by the user on the SAVE command. This name helps the user select the correct extract ID for cancellation.

After entering the CANCEL command, the current dialog will be temporarily interrupted to process it. Once the command has been executed, the panel from which it was entered is again displayed.

Examples of CANCEL

Example 1

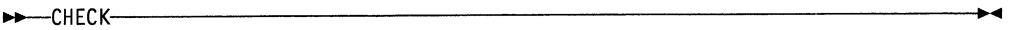
This example cancels an extract request named EXTID_85175_164347:

```
CANCEL EXTID_85175_164347
```

CHECK

Use the CHECK command to ensure that the information necessary for the successful execution of an extract request has been provided during the dialog session.

It is not necessary to issue the CHECK command prior to issuing the SEND command. An implicit check is automatically performed before the request is sent. It is a good idea, however, to use the CHECK command prior to saving (with the SAVE command) an extract request.



CHECK (REQUIRED)

checks various conditions about the extract request you are working on. CHECK is only issued in the End User Dialogs.

For example, you might use the CHECK command if:

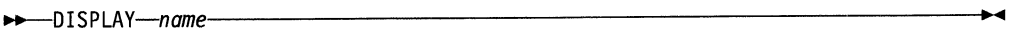
- Multiple tables or DXTVIEWs have been selected
- Columns have been selected
- A join has been specified on the JOIN panel
- Access information has been specified for the location of tables or DXTVIEWs that have been specified

If missing or invalid information is encountered, the panel where the first omission or error occurred will be displayed, with the cursor positioned at the field in question.

If no omissions or errors are located, the panel from which the command was entered is again displayed. A message will be issued indicating if there was missing information or if there was a normal completion.

DISPLAY

Use the DISPLAY command to display the input fields of a saved extract request.



DISPLAY (REQUIRED)
specifies an extract to display.

name
identifies the extract request to be displayed. Replace *name* with the name of the relevant saved extract request (the name may contain up to 18 characters).

If *name* is not specified, a list of saved extract requests is displayed and one can be selected. If you are in an object sharing session, only the extract requests in the common library will be displayed. If you are not in an object sharing session, the extract requests in your personal DataRefresher library will be displayed.

The particular extract request panel initially displayed depends on which panel you are on when you issue the DISPLAY command. DataRefresher fills in all information pertinent to that panel and subsequent extract request panels you access. Navigate between the panels to view this information.

The DISPLAY command is cancelled when you use the END command.

Examples of DISPLAY

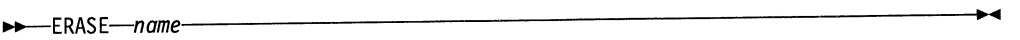
Example 1

This example displays an extract request named EXTRACT1:

```
DISPLAY EXTRACT1
```

ERASE

Use the ERASE command to erase a saved extract request from the DataRefresher dialogs table library.



ERASE (REQUIRED)
erases an extract request.

name
identifies the extract request to be erased. Replace *name* with the name of the relevant saved extract request.

If *name* is not specified, a list of your saved extract requests is displayed so you can select one. If you are in an object sharing session, this list will contain the extract requests in the common library. You must have WRITE access to the common library to erase a shared request.

Before the ERASE command executes, a confirmation panel is displayed:

- If NO is entered on the confirmation panel, the ERASE command is terminated and the request is not erased.
- If YES is left on the panel and ENTER is pressed, the designated extract request is erased.

After entering the ERASE command, the current dialog will be temporarily interrupted to process it. Once the command has been executed, the panel you were in when you entered the command appears again.

Examples of ERASE

Example 1

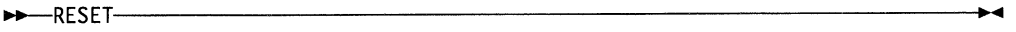
This example erases an extract request named EXTRACT1:

```
ERASE EXTRACT1
```

RESET

Use the RESET command to reset entry fields on a panel to the defaults.

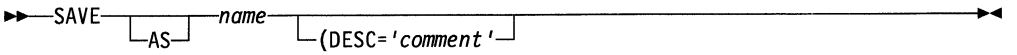
If you issue the RESET command to reset the TABLES panel, the whole End User Dialogs environment is reset.



RESET (REQUIRED)
resets the entry fields of the panel displayed to the default values.

SAVE

Use the SAVE command to save the current extract request under the specified name, and store it in the DataRefresher dialogs table library.



SAVE (REQUIRED)
saves the current extract request.

AS (OPTIONAL)
may be specified for clarification.

name
identifies the extract request to be saved.

Replace *name* with the name of the relevant extract request to be saved. The value of *name* can be a name up to 18 characters.

If an extract request of the same name already exists, a confirmation panel for replacing the request is displayed. If the *name* parameter is not specified, an error message is displayed.

If you enter SAVE while in an object sharing session, and you do not have WRITE access to the common library, the object will be saved in your personal DataRefresher dialogs library.

(DESC='*comment*' (OPTIONAL)
saves a comment associated with the request. A maximum of 41 characters is allowed. It will subsequently appear, and may be changed, each time the request is displayed.

The comment must be enclosed in single quotes; only the enclosed text will be stored. If single quotes are to be part of the stored text, each quote must be doubled so that it will not be misinterpreted as a string terminator.

After entering the SAVE command, the current dialog will be temporarily interrupted to process the command. If the extract request to be saved already exists in the DataRefresher dialogs library, a confirmation panel is displayed:

- If NO is issued, the SAVE command will not be executed and the extract request is not replaced.
- If YES is issued, the designated extract request is replaced and saved.

Once the SAVE command has been executed, the panel from which it was entered is displayed again.

The JCL and JCS associations for the extract request are made according to the following scheme:

Note: You may use the Profile Review Panel to override some fields specified in your dialogs profile:

- If the request extracts data from IMS, a VSAM or physical sequential data set, or a data source accessed by a GDI exit, the JCL named UIM JCL in the user's dialogs profile is used.
- If the request extracts data from DB2, the JCL named REM JCL in the user's dialogs profile is used.

- If the request extracts data from SQL/DS, the EXEC named as REM EXEC in the user's dialogs profile is used.
- If the extract output or control deck routing is JCS:

  When the target type specified in the request is DB2, the JCS named DB2 JCS in the user's dialogs profile is used.

  When the target type specified in the request is SQL/DS, the JCS named as SQL/DS JCS in the user's dialogs profile is used.

  When the target type specified in the request is IXF, the JCS named Other JCS in the user's dialogs profile is used.
- If the extract output and control deck routing are physical sequential, the attributes of the physical sequential data set or the *ddname* in the execution JCL is used.
- If JCL or JCS files are not named in the dialogs profile, a message indicating missing information is displayed after the SAVE command is executed. The JCL or JCS will be retrieved at send time if the information was specified.

Examples of SAVE

Example 1

The following examples save the current extract request under the name EXTRACT1:

```
SAVE AS EXTRACT1 (DESC='extract from MYFILE1 to MYFILE2
SAVE EXTRACT1 (DESC='extract from MYFILE1 to MYFILE2
```

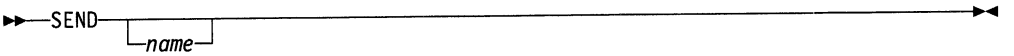
SEND

General-use programming interface

Use the SEND command to direct an extract request to the UIM or REM for processing. You can also use it from other products that have bridged to DataRefresher

If the specified request extracts data from IMS, a VSAM or physical sequential data set, or a data source accessed by a GDI exit, it is sent to the UIM. When a request is sent to the UIM, an extract ID is generated and saved for use by the DataRefresher Dialogs STATUS and CANCEL commands.

If the specified request extracts data from DB2 or SQL/DS, it is sent to the REM.



SEND (REQUIRED)
directs an extract request for processing.

name **(OPTIONAL)**
identifies the extract request to be sent. Replace *name* with the name of the relevant extract request to be sent. The name must consist of no more than 18 characters.

If *name* is not specified and an extract request is currently being built or updated, that request will be sent. Otherwise, a list of saved extract requests is displayed for selection purposes. If you are in an object sharing session, this list contains the names of the extract requests in the common library. Otherwise, the requests in your personal DataRefresher dialogs library are listed.

When the ENTER key is pressed, a confirmation panel is displayed. If you confirm the SEND, the job stream is built using the JCL and JCS associated with the request at save time. If you did not save the request, or the information was missing at save time, the JCL and JCS association is made using the following scheme:

- If the request extracts data from IMS, a VSAM or physical sequential data set, the JCL named as UIM JCL in the user's dialogs profile is used.
- If the request extracts data from DB2, the JCL named as REM JCL in the user's dialogs profile is used.
- If the request extracts data from SQL/DS, the EXEC named as REM EXEC in the user's dialogs profile is used.
- If the extract output or control deck is JCS and the target type specified in the request is:
  - DB2, the JCS named as DB2 JCS in the user's dialogs profile is used.
  - SQL/DS, the JCS named as SQL/DS JCS in the user's dialogs profile is used.
  - IXF, the JCS named as OTHER JCS in the user's dialogs profile is used.
- If the extract output or control deck is physical sequential, the attributes of the physical sequential data set or the *ddname* in the execution JCL is used.

Note: You may use the Profile Review Panel to override the JCL/JCS files specified in your Dialogs profile

After issuing SEND, the current dialog is temporarily interrupted to process the command. Once the SEND command executes, the panel from which it was entered is displayed again.

Examples of SEND

Example 1

This example sends an extract request named EXTRACT1 from within DataRefresher for processing:

```
SEND EXTRACT1
```

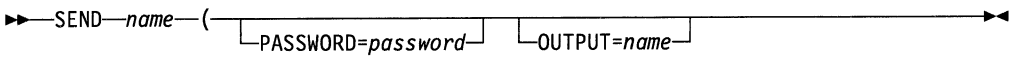
Using SEND from other products

SEND can be passed as a parameter from products bridging to the End User Dialogs. This lets the user of a product with a DataRefresher bridge send a named extract request (previously created and saved) to the UIM or REM for processing without seeing any End User Dialogs panels. For more information about using ISPF commands to invoke the End User Dialogs from a user-written program, see the *DataRefresher Administration Guide*.

Using SEND from other products

SEND is the only command that can be passed as a parameter from products bridging to the End User Dialogs. This lets the user of a product with a DataRefresher bridge send a named extract request (previously created and saved) to the UIM or REM for processing without seeing any End User Dialogs panels.

The syntax for SEND when passed as a parameter is:



Enclose the entire string, including the command name SEND, between single quotes.

SEND (REQUIRED)
sends an extract request to DataRefresher from another product.

name **(REQUIRED)**
identifies the extract request to be sent. This must be a name of no more than 18 characters. The extract request named must have been previously saved in the End User Dialogs environment.

PASSWORD=*password* **(REQUIRED only for SQL/DS)**
specifies the password required to access the relational data.

OUTPUT=*name* **(OPTIONAL)**
indicates the name of a file or data set to be substituted in the JCS. This file name will be substituted in the JCS only if you have coded the &OUTPUT variable in the JCS.

Note: For more information about using the SEND command from other products, see the *DataRefresher Administration Guide*.

SEND (REQUIRED)
sends an extract request from another product to DataRefresher.

name **(REQUIRED)**
identifies the extract request to be sent, and must be a name of no more than 18 characters. The extract request named must have been previously saved in the End User Dialogs environment.

PASSWORD=*password* **(REQUIRED only for SQL/DS)**
specifies the password required to access the relational data.

The password is required when the data source is SQL/DS. When the source is DB2 or DataRefresher (a DEM source), the access password is optional and depends on the job card in the execution JCL. For security reasons, the password is not saved in the DataRefresher End User Table across sessions. Replace *password* with the relevant password.

OUTPUT=*name* **(OPTIONAL)**
indicates the name of a file or data set to be substituted in the JCS. This file name will be substituted in the JCS only if you have coded the &OUTPUT variable in the JCS as in this DRU JCS example:

```
//OUTDRU DD DSN=&OUTPUT
```

name may be a data set name or CMS file ID. When the name is a CMS file ID, you must enclose it in single quotes. The name you specify will not be validated by DataRefresher End User Dialogs

Products bridging to DataRefresher End User Dialogs may need to obtain a list of saved extract requests. This list of saved extract requests and related information is available in the DataRefresher dialogs output library table, DVRRPXRQ. To create this list, the bridging product should use the following field variables:

- REQNAMET — extract request name
- REQFORMT — extract request format (always prompt)
- STABTYPE — extract request source data type (DB2, SQL/DS, or DXT)

Note: For more information about using the SEND command from other products, see the *DataRefresher Administration Guide*.

Example

This example sends an extract request named EXTRACT2 for processing from within QMF*:

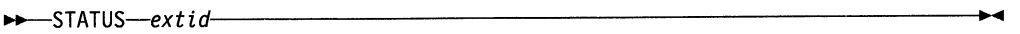
```
'SEND EXTRACT2 (PASSWORD=userpass OUTPUT='extrct output f''
```

End of General-use programming interface

* QMF is a trademark of the International Business Machines Corporation.

STATUS

Use the STATUS command to check the status of a DEM extract request previously sent to the UIM for processing.



STATUS (REQUIRED)
checks the status of a DEM extract request.

extid **(REQUIRED)**
identifies the ID of the extract request whose status you are checking. Replace *extid* with the relevant extract ID.

The extract ID is an 18-character system-generated identifier automatically assigned to the request when it is sent to the UIM for processing:

EXTID_*yyddd_hhmmss*

where *yyddd* is the Julian date and *hhmmss* is the time when the extract request was sent for execution. The extract ID is used to distinguish between multiple submissions of the same request name, thus permitting the user to check the status of or cancel a given submission. The user-assigned extract name does not accommodate this differentiation. The *name* keyword of the SAVE command is used to assign a unique identifier to the extract request being saved. This name lets the user distinguish between the current extract request and other extract requests the user may have created.

If *extid* is not specified, a list of extract IDS for all extract requests sent in the last seven days to the UIM will be displayed for selection purposes.

After entering the STATUS command, the current dialog will be temporarily interrupted to process the command. Once the command has been executed, the panel from which it was entered is displayed again.

Examples of STATUS

Example 1

To display the status of an extract request with the system-generated name EXTID_85175_16437, enter:

```
STATUS EXTID_85175_164347
```

Appendix A. DataRefresher limits

Data description command limits

Data description limits

Data Description	Minimum	Maximum
Segments and fields allowed in a DXTFILE/DXTPCB	1	1530
DXTPCBs, segments and fields allowed in a DXTPSB	1	65535
DXTPCBs allowed in a DXTPSB	1	255
DXTPCBs allowed in an MSDB DXTPSB	1	1
DXTPCBs allowed in a DEDB DXTPSB	1	127
Segments allowed in a DXTFILE	0	255
Segments allowed in a DXTPCB	0	255
Level of segment nesting (hierarchy depth)	1	16
Fields allowed in a DXTVIEW	1	1530
FREQ keyword value	0.01	16777215.00
XBYTES keyword value	1	32760

Data type limits

Data Type	Minimum	Maximum
Timestamp data type	16	26
Date data type	1	10
Character data type	1	254
Graphic data type	2	254
Packed Decimal data type	1	16
Variable Character data type	1	32767
Variable Graphic data type	2	32766
Zoned Decimal data type	1	16
Single Byte Binary data type	1	1
Double Floating Point data type	8	8
Single Floating Point data type	4	4
Halfword Integer data type	2	2
Fullword Integer data type	4	4

Submit command limits

Submit and Extract limits

Description	Minimum	Maximum
Output limit keyword value	1	10000000
Priority keyword value	1	255
FLDERR(SKIP) keyword value	1	10000
FLDERR(SUBST(NULLS)) keyword value	1	10000
FLDERR(SUBST(ZERO)) keyword value	1	10000
FLDMSG keyword value	1	1000000
DEBUG keyword value	1	4
Selected fields when output format is IXF	1	750
Selected fields when output format is DB2	1	750
Selected fields when output format is DB2 and WITH REPLACE specified on EXTRACT	1	750
Selected fields when output format is DB2 and WITH CREATE specified on EXTRACT	1	450
Selected fields when output format is SQL/DS	1	255
Selected fields when source data is from SQL/DS	1	255
INTO fields when output format is IXF	1	750
INTO fields when output format is DB2	1	750
INTO fields when output format is DB2 and WITH REPLACE specified on EXTRACT	1	750
INTO fields when output format is DB2 and WITH CREATE specified on EXTRACT	1	450
INTO fields when output format is SQL/DS	1	255
300		
INTO fields when source data from SQL/DS	1	255
Views specified on FROM clause	1	16
GDI data sources for a GDI SELECT exit	1	16
Fields specified on the ORDER BY clause	1	8
Offsets of fields specified on the ORDER BY clause	1	4092
Number of constants specified in the WHERE clause	1	255

Value ranges for constants used on a WHERE clause

Constant type	Minimum	Maximum
Single Byte Binary constants	1	255
Character and Variable Character (SBCS) constants	1 character	254 characters
Graphic and Variable Graphic (DBCS) constants	1 character	127 characters
Single Floating Point constants	5.4E-79	7.2E+75
Double Floating Point constants	5.4E-79	7.2E+75
Fullword constants	-2147483648	+2147483647
Halfword constants	-32768	+32767
Packed Decimal constants	1 digit	31 digits
Zoned Decimal constants	1 digit	16 digits
Timestamp constants	16 bytes	26 bytes
Time constants	4 bytes	8 bytes
Date constants	8 bytes	10 bytes

Other DataRefresher command limits

Command	Minimum	Maximum
Data descriptions deleted on one command	1	16
Data descriptions printed on one command	1	16
Data descriptions punched on one command	1	16
Extract requests cancelled per command	1	1
Running time (in minutes) of DEM using RUNMODE keyword	1	32767
Poll interval (in seconds) of DEM	1	3600
Number of USE DXTPSBs in DXTIN dataset	0	1
Number of USE EXTIDs in DXTIN dataset	0	1
Number of records in REM DXTIN dataset	1	400
Number of SUBMIT commands in REM DXTIN dataset	1	16

Structure Access Program (SAP) limits

Characteristic	Minimum	Maximum
Length of field name	1 character	18 characters
PL/I and COBOL compiler limits	Set to DXT limit	Set to DXT limit

Dialogs limits

Characteristic	Minimum	Maximum
Tables and/or views selected on an extract request	1	4

Output dataset limits

Characteristic	Minimum	Maximum
LRECL of output data set	20	32760
Blocking factor of output data set	1	5000
Number of blocks written for output data set	1	2147483647
Length of data set name on EXTDATA keyword	System Dependent	44
Length of UNIT identifier	System Dependent	8
Length of VOLUME SERIAL identifier	System Dependent	6
Length of data set password	System Dependent	8

DataRefresher restrictions

Restriction
LE/370 compiled exits may only be run with other LE/370 exits or Assembler exits. CLE cannot be specified with COBOL or PLI on the EXTILANG parameter.
The DEM must be linked in 24 bit addressing mode when using HSSR access and any release of IMS Version 2 (XA).
The DEM must be linked in 31 bit addressing mode when using HSSR access and any release of IMS Version 3 (ESA).
Variable blocked not supported for output data set unless output format is IXF
DXTOUT and SYSOUT should not be used as an EXTDATA keyword value

Valid signs for decimal numeric key fields

Data Type	Valid Sign
Positive packed decimal	C
Negative packed decimal	D
Positive zoned decimal	F
Negative zoned decimal	D

Appendix B. DataRefresher return codes

DAP return codes

The DAP function sets one of these return codes and sends it to the DB/DC Data Dictionary:

0 No errors detected.

4 This is usually issued for DAP user errors, including missing or invalid (for DataRefresher) information in the Dictionary. The user should refer to the listing of error messages. The DAP output is incomplete.

8 This is usually a DAP system error. Remaining commands in the job step do not execute if the Dictionary is currently running with FLUSH=YES. The problem could be related to an inability to obtain storage or to open the DXTDUMP data set. The DB/DC Data Dictionary also issues the following message:

```
DBD0112  Unable to complete requested function
```

16 Sufficient automatic storage to run the DAP cannot be obtained.

24 Write error. The DB/DC Data Dictionary cannot generate output.

If the return code is the highest one during the current job step, the DB/DC Data Dictionary then terminates with that return code.

DEM return codes

For the DEM, there are two sets of return codes. The return code you receive depends upon the subfunction that issued it. However, certain general characteristics of the return codes issued by all of the subfunctions within the DEM are described here.

The return codes expected during an extract-related global function for an existing set of extract requests represented by the DVRXCIUT (this "In-Use Table" set might contain as few as only one extract request) are:

0 No errors encountered.

4 Noncritical errors or warning situations encountered, and/or one or more (but not all) of the extract requests were stopped. For the stopped extract request, a user message has already been sent by the subfunction, detailing the error condition, and the disposition of the extract request is coded in the DVRXCIUT entry for the extract request.

8 Either the entire set of extract requests (that is, the entire run) was stopped as a whole, in which case the reason code and extract request disposition information is put into the DVRXCIUT by the subfunction (individual extract requests can stop prior to stopping the run), or, all of the extract requests stop individually. For individually stopped extract requests, the statement given for **RC=4**

above applies. In any event, the subfunction is still functioning and can be called again.

12 Environment problem. For some reason related to the environment, the subfunction was not able to complete the requested processing. It is not able to continue processing, and does not expect to be called again. Reason code and extract request disposition are placed in the DVRXCIUT. An example of this type of error is a GETMAIN failure.

16 System or program problem. An unexpected situation has developed, and the subfunction was unable to complete the requested processing and cannot be called again. Reason code and extract request disposition are placed in the DVRXCIUT. Examples of this type of error are an unexpected option passed by a calling module, and an unexpected return code passed from a called module.

20 Insufficient storage in DEM internal automatic storage pool. A message provides more information.

24 An error was encountered while writing to the DXTPRINT data set. No further DEM message is provided, although a system message might have been issued.

28 An invalid return code was passed to the DVRXSHEX module internally. A message provides more information.

32 DXTPRINT data set has an invalid logical record length. An LRECL of 121 must be used.

36 The CVT environment bits are not recognizable. No message accompanies this code.

The return codes related to processes that are not for the entire set of extract requests, nor for any specific extract request, are:

0 No errors encountered.

4 Noncritical errors or warning situations encountered.

8 Subfunction did not execute the requested processing due to errors, but it can continue processing. It can be called again.

12 Environment problem. For some reason related to the environment, the subfunction was not able to complete the requested processing. It is not able to continue processing, and it does not expect to be called again. Reason code and extract request disposition are placed in the DVRXCIUT when applicable. (Whenever an extract request is involved, there is an DVRXCIUT.) An example of this type of error is a GETMAIN failure.

16 System or program problem. An unexpected situation has developed, and the subfunction was unable to complete the requested processing. It is not able to continue processing, and it does not expect to be called again. Reason code and extract request disposition are placed in the DVRXCIUT when applicable. (Whenever an extract request is involved, there is an DVRXCIUT.) Examples of this type of error are an unexpected option passed by

a calling module, and an unexpected return code passed from a called module.

DRU return codes

DRU return codes are written to the DXTPRINT data set:

0 Normal exit.

4 Noncritical error during DRU processing.

8 DRU processor terminated because the INDRU data set was empty.

12 DRU processing terminated because the LRECL of the DXTPRINT data set was not 121.

16 Unexpected end of DRU processing.

32 DRU processing failed because the open of the DXTPRINT data set failed.

FMU return codes

The FDTLIB Migration Utility function sets one of the following return codes:

0 Normal completion. No errors were encountered.

4 Minor errors were encountered, such as:

- Open error on DXTDUMP data set
- Input/output files could not be closed

8 Critical errors were encountered, such as:

- Storage could not be acquired
- Initialization incomplete
- VSAM control blocks could not be generated
- Input/output files could not be opened
- Read/write errors detected

12 Unable to open DXTPRINT data set.

32 Bad DXTPRINT LRECL after open.

MIT return codes

0 No errors encountered.

4 Minor errors were encountered, such as:

- Unable to end processing of table
- Unable to sort table
- Unsuccessful scan of table
- Unable to free program storage

8 Critical errors were encountered, such as:

- Unable to read record from VM
- Unable to read record from input
- Unable to erase table
- Unable to add entry to table
- Unable to end processing on table

12	Processing completed with severe errors.
36	Unrecognizable environment.

REM return codes

0	No errors encountered.
4	Noncritical errors or warning situations encountered, and/or one or more (but not all) of the extract requests were terminated. For each terminated extract request, a user message has already been sent by the subfunction, detailing the error condition. The disposition of the extract request is coded in the DVRXCEXB entry for the extract request. Extract request(s) is terminated.
8	Either the entire set of extract requests (that is, the entire run) terminate as a whole, in which case the reason code and extract request disposition information is put into the DVRXCEXB by the subfunction (individual extract requests can terminate prior to stopping the run), or, all of the extract requests terminate individually. For individually terminated extract requests, the statement given for **RC=4** above applies. In any event, the subfunction is still functioning and can be called again. Extract request(s) is terminated.
12	Environment problem. For some reason related to the environment, the subfunction was not able to complete the requested processing and cannot be called again. Reason code and extract request disposition are placed in the DVRXCEXB. An example of this type of error is a GETMAIN failure. REM is terminated.
16	System or program problem. An unexpected situation has developed, and the subfunction was unable to complete the requested processing. It is not able to continue processing and cannot be called again. Reason code and extract request disposition are placed in the DVRXCEXB. Examples of this type of error are an unexpected option passed by a calling module, and an unexpected return code passed from a called module. REM is terminated.

UIM return codes

0	Normal completion. No errors were encountered.
4	User errors were encountered (such as syntax errors or errors that are a result of commands that do not reflect the contents of the FDTLIB or EXTLIB). All problems are detailed in message(s).
8	Critical errors were encountered, such as: • All system errors that are not associated with other return codes • All problems that are recognized as DataRefresher errors All problems are detailed in messages.

12	DXTPRINT or DXTDUMP data set could not be opened. No message accompanies the failure to open DXTPRINT.
16	Storage for initialization of the UIM could not be opened. In some cases, a message accompanies this return code.
20	Insufficient storage in UIM internal automatic storage pool. A message provides more information.
24	An error was encountered while writing to the DXTPRINT data set. No further UIM message is provided, although a system message might have been issued.
28	An invalid return code was passed to the DVRXSHEX module internally. A message provides more information.
32	DXTPRINT data set has an invalid logical record length. An LRECL of 121 must be used.
36	The CVT environment bits are not recognizable. No message accompanies this code.

Appendix C. DataRefresher dialogs models

DataRefresher models provide you with examples for several tasks. You can find the models:

- In a data set called DVR110.DVRJEDIE on MVS
- In a macro library called DVRJEDIE MACLIB on VM

You can view a model in several ways:

- Use the dialogs to access the model
- Browse it as you would any other data set
- Print the model in the same way that you would print a data set
- Browse and edit the model while using Online DataRefresher commands.

The models contain instructions explaining how to change any model to suit your needs.

Data description models

MODEL	Description
DVREDREF	Creates a DataRefresher file description.
DVREDREP	Creates a DXTPSB description.
DVREDCRD	Creates a DataRefresher data type description.
DVREDRVF	Creates a DXTVIEW (of a file) description.
DVREDRVP	Creates a DXTVIEW (of a PCB) description.

JCL/JCS models

MODEL	Description
DVREDREM	Starts the REM when extracting from a DB2 source.
DVREDREV	Starts the REM on VM.
DVREDDAP	Starts the DAP.
DVREDDXT	Starts the DXTINPUT procedure.
DVREDDJC	Loads extracted data into DB2.
DVREDDBC	Creates and loads into DB2.
DVREDSJC	Loads extracted data into SQL/DS.
DVREDRUM	Executes the DRU in MVS.
DVREDRUV	Executes the DRU in VM.
DVREDEJM	Executes DataRefresher's MIT utility in MVS.
DVREDEJV	Executes DataRefresher's MIT utility in VM.

Extract request models

MODEL	Description
DVREDRXV	Submits REM extract requests on VM using the SUBMIT(REM) command.
DVREDRXM	Submits REM extract requests on MVS using the SUBMIT(REM) command.
DVREDEXT	Submits DEM extract requests using the SUBMIT(UIM) command.

SAP Skeletons

NAME	Description
DVRXKJDA	SAP dialog (CREATE DXTPSB)
DVRXKJFA	SAP Dialog (CREATE DXTFILE)

Appendix D. Coding the Description

Several DataRefresher commands have an optional keyword DESC= *comment*. The rules for coding that keyword are:

- The description can include purely character data, or a mixed string of character and DBCS data.
- The DBCS portion of a mixed string must be bracketed by a shift-out (X'0E') and shift-in (X'0F') character, as in this example:

  ```
  'characterdata<D_B_C_S>'
  ```

  (X'0E' is represented by < and X'0F' by >.) A maximum of 50 bytes is allowed for both character and mixed descriptions, whether entered in DataRefresher Dialogs or in batch mode.
- Enclose descriptions, character or mixed, in single quotes. If single quotes are part of the stored text, double each quote so that it will not be misinterpreted as a string terminator.

Appendix E. Diagnostic information

Diagnostic information is maintained by DataRefresher for each debug level.

There are four DEBUG levels that determine the type of diagnostic information DataRefresher provides. You can specify one of these levels in the following:

- EXEC statement in the JCL for the UIM or REM
- INITDEM (DEM) command
- EXECUTE (DAP) command
- EXTRACT (UIM) command
- CHANGE (DEM) command

The following table shows the type of diagnostic information DataRefresher provides depending on the level specified.

Type of Diagnostic Information	Level 1	Level 2	Level 3	Level 4
REM, DEM, UIM, or DAP maintains an internal wrap around trace table showing module entry and exit points.	X	X	X	X
DEM writes trace information about all data access method and source file (IMS/VS DL/I, physical sequential, and VSAM) calls.		X	X	X
REM, DEM, UIM, or DAP writes trace data whenever it enters or exits a module, or whenever it calls to or returns from an exit routine.			X	X
REM, DEM, UIM, or DAP writes control area information whenever it enters or exits a module or other internally specified location, and dumps the exit routine control block before and after calls to an exit.				X

The greater of the two debugging level specifications specified on SUBMIT command or invocation parameter will be used.

The higher the debugging level, the greater the volume of diagnostic information maintained and written as output. Also, the higher the number the more it will impact a DEM's (or other DataRefresher component's) performance.

Appendix F. Output data records and fields

What is a data record

A data record is a line of data in a file. Data Records either replace the *EO statement in the JCS associated with your extract request or are inserted in the data set named on the EXTDATA keyword of your request. As such, they represent the extracted data.

The extracted data, or data records, can be in either SOURCE or EBCDIC format. Use the FORMAT keyword of the SUBMIT command to specify the format you require.

In the data records, each extracted field is preceded by a one-byte null indicator. If a particular value in a field is null, the byte contains a hyphen. If the value is not null, the byte is blank.

Note: Data records also represent messages if the JCS you use for loading your extracted data can also be used for loading DataRefresher messages into a message table.

Data records in an output job

When your extracted data is embedded in an output job, each data record represents all or part of a row of the target table or message table. All the data records embedded in an output job (to send data across networks) are 80 characters (bytes) long. If all the data in a given row cannot be represented by a single data record, the row is spread across a succession of additional data records:

- Within a data record for a given table row, characters 2 through 80 can represent data.
- If more than one data record is needed, columns 2 through 80 on the additional records can also represent data; column 1 never represents data.
- Column 1 of a data record is used to indicate continuation. It carries a hash sign (#) if it and the next record represent parts of the same table row; otherwise, it contains a blank.

In the example below, the first column of the first record contains a pound sign; one table row, therefore, consists of two data records.

```
# 10 HEAD-OFFICE  160 CORPORATE  NEW YORK  160 MOLINARE
  MGR  000007 0229 59.20 000000.00
```

When more than one record per row is needed, fields may wrap around from one record to the next. The wrap-around is continuous. For example, a 4-byte field that begins in column 79 of one record has its first 2 bytes in columns 79 and 80 of that record, and its last 2 bytes in columns 2 and 3 of the next record.

Data records in data sets and files

When your extracted data is written to a physical sequential data set or CMS file, each data record represents a row of the target table or message table. Data records in data sets or files can be any number of characters (bytes) long, depending on the lengths of the fields being extracted. Within a data record for a given table row, all the characters (that is, characters 1 through the last character) can represent data.

How individual fields are represented in EBCDIC format

This section gives a detailed account of how fields are represented in DataRefresher output when the fields are converted from source to EBCDIC format.

source field For a given row of output, the source fields are the fields in the source or sources of data that are listed in the SELECT keyword of the EXTRACT statement.

target field A target field is the representation of some source field in a data record.

Within a data record, the target fields for a row of output appear in the order in which their source fields are named in the SELECT statement of the corresponding EXTRACT statement.

There is no space between target fields. The first target field for a row begins in column 2 of its data record, the second immediately follows the first, and so on.

A target field begins with a one-column *nulls indicator*:

- If the nulls indicator contains a hyphen (-), the target field is null (value unknown) and is set to blank.
- If the column is blank, the target field is not null.

Because most fields are not null, the nulls indicator also acts as a separator between columns when the output data set or file is printed or is displayed at your terminal.

The data portion of a target field follows the nulls indicator. When the field is not null and you have indicated to Data Refresher that it should convert the source data to EBCDIC format, the data portion represents the data as an EBCDIC character string, no matter what the data characteristics of the source field. If, for example, the source field is an H-type field with a hexadecimal content of X'8000', the data portion of the target field would contain -32768, which is the decimal representation of the integer contained in the source field. When the nulls indicator contains a hyphen, the data portion of the field is set to blanks.

Note: If a target field is to be loaded into a column and if the source field is zoned or packed decimal, assign a value other than (SUBST(NULL)) or (SUBST(NULL),n) to the FLDERR keyword on the OPTIONS keyword of the EXTRACT statement. With either of these two options, DataRefresher substitutes nulls and generates NULL IF statements for the fields in your extract request that contain errors; this would result in the extract request being rejected.

The following subsections discuss how the data portion of a target field represents its data in EBCDIC format. This depends on the data characteristics and length of the source field.

B-type fields

When the source field is a B-type field, it represents an integer as a one-byte (8-bit) unsigned binary integer in the range of 0 through 255. Since the higher order bit is interpreted as data, not as the sign, B-type fields are always positive. When extracted for a relational or target in IXF, B-type fields map to the corresponding two-byte signed binary (H-type) columns or fields in the target.

C-type fields

When the source field is a C-type field, the data portion of the target field is merely a copy of the source field. Normally, a byte in the source field (and its copy in the target field) has an EBCDIC representation. However, this need not be true; a byte in a C-type field can have any one of the possible 256 values, whether or not that value represents an EBCDIC character.

A-type, T-type, and S-type fields

A-type (date) data, T-type (time) data, and S-type (timestamp) data is collectively known as *date/time data*.

Note: When extracting date/time data with the DEM, the DEM produces date/time output in standard ISO format. This is not the case when using the REM to extract data. User-supplied date/time conversion exits used in the DEM are also expected to produce date/time data in standard ISO.

The standard ISO format for each type of date/time data output is shown in the following table:

Data Type	Format	Description
DATE	yyyy-mm-dd	A date is always a ten-byte character string
TIME	hh.mm.ss	A time is always an eight-byte character string
TIMESTAMP	yyyy-mm-dd-hh.mm.ss.nnnnnn	A timestamp is always a 26-byte character string

where:

yyyy = year
mm = month
dd = day
hh = hour
mm = minute
ss = second
nnnnnn = nanosecond

Examples:

```
1986-12-30                  (Date data)
03.59.30                    (Time data)
03.59.30                    (Time data)
1963-07-29-11.59.58.000001  (Timestamp data)
```

D-type and E-type fields

D-type fields

When the source field is a D-type field, it represents a number in long (8-byte) floating point format. Its target field represents this number in the E-form (that is, exponent form) used by PL/I. The data portion of this field is 23 bytes long. It contains a mantissa and an exponent, which are separated by an uppercase 'E'.

- The mantissa, to the left of the E, is a 17-digit, signed number with a decimal point between its first and second digits. Unless the number represented is zero, the leftmost digit in the mantissa is never zero. If the number is negative, this digit is preceded by a hyphen. If it is positive, the digit is preceded by a leading zero.
- The exponent is a two-digit, signed integer that indicates how many places to the right or left the decimal point in the mantissa must be moved to obtain the number represented. A positive value indicates movement to the right. A negative value indicates movement to the left. A value of zero implies that the number represented is the mantissa.

Examples:

```
-1.2345678901234567E+10 represents -12345678901.234567
 1.2345678901234567E-03 represents  .0012345678901234567
 0.0                    represents  0
```

Note: Zero, as shown above, has a special representation that does not follow the rules stated above. All other numbers have representations that do follow the rules.

E-type fields

When the source field is an E-type field, it represents a number in short (4-byte) floating point format. As with a D-type field, its target field represents this number in E-form notation, but the representation is shorter. The data portion for an E-type field is only 14 bytes long to accommodate an eight-digit mantissa.

Examples:

```
-1.2345678E+04  represents -12345.678
 1.2345678E-3   represents  .0012345678
 0.0            represents  0
```

F-type and H-type Fields

F-type fields

When the source field is an F-type field, it represents an integer as a fullword (32-bit) signed binary integer. The target field represents this number in decimal. Its data portion is 11 bytes long to accommodate the largest negative integer (including sign) that the source field can represent. A hyphen always appears at the left edge of the data field if the number represented is negative. If the number is positive, this leftmost byte contains a leading zero.

Examples:

```
 0000012345   represents the integer 12345.
-2147483648   represents the largest negative
              integer that the source field
              can hold.
```

H-type fields

When the source field is an H-type field, it represents an integer as a halfword (16-bit) signed binary integer. The target field represents this integer in decimal exactly as it would for an F-type field, except that the data portion of the field is only 6 bytes long.

Examples:

```
 00123   represents the integer 123.
-32768   represents the largest negative integer
         that the source field can hold.
```

G-type fields

When the source field is a G-type field, it represents double-byte character set (DBCS) data. This means that two bytes are used to represent one DBCS character. The length is specified in bytes, and must be an even number in the range of 2 through 254. The DBCS data input cannot be delimited by shift-out (X'0E') and shift-in (X'0F') characters in storage.

If the INTO keyword is omitted from the EXTRACT command, DataRefresher assumes that the data is targeted for a physical sequential data set or CMS file. In this case, when DataRefresher extracts data from a G-type field, shift-out and shift-in characters are included for all DBCS data. The source data from a G-type field is assumed to be totally DBCS data (not mixed).

For relational databases, the shift-out and shift-in characters are not included with DBCS data, because SQL/DS and DB2 assume that G-type fields are totally DBCS data and not mixed. If DBS=IXF, shift-out (X'0E') and shift-in (X'0F') characters are placed around all extracted GRAPHIC data, even if an INTO keyword is specified.

P-type and Z-type fields

P-type fields

When the source field is a P-type field, it represents a number in packed decimal format. The target field represents this number in fixed decimal format; that is, as a sign followed by a string of digits and a decimal point.

Example:

```
-123.45
```

The decimal point can lie within the string of digits or immediately to the right or left of it, the position having been determined by the scale attribute of the source field. The scale attribute is a non-negative integer that indicates how many decimal digits lie to the right of the decimal point. For example, the source field represented by -123.45 has a scale attribute of 2.

The scale attribute of the source field is contained in the field's underlying DataRefresher file or PSB description. It was specified by the FIELD keyword that defined the source field.

The data portion of the target field is just large enough to hold the largest negative number that the source field can represent. For an *n*-byte source field, this requires 2x *n*+1 bytes. Non-significant leading zeros are represented as zeros, instead of blanks. If the number represented is negative, the leftmost byte in the data portion of the target field contains a hyphen. If the number is positive, this byte contains a leading zero.

Examples:

```
The target field for a 3-byte source field
might contain:
 -12345. (scale factor of zero)
  005.43 (scale factor of two)
  .02178 (scale factor of five)
```

Z-type fields

When the source field is a Z-type field, it represents a number in zoned decimal format. The target field represents this number in fixed decimal format, just as it would if the source field were a P-type field. The representation differs from that of a P-type field only in the relative lengths of the source and target fields. Because a Z-type field represents one decimal digit per byte instead of two, the data portion of the target field for an *n*-byte, Z-type field is *n*+2 bytes long.

VC-type and VG-type fields

VC-type fields

When the source field is a VC-type field, the data portion of the target field is merely a copy of the variable-length source field. Normally, a byte in the source field (and its copy in the target field) has an EBCDIC representation. However, this need not be true; a byte in a VC-type field can have any one of the possible 256 values, whether or not that value represents an EBCDIC character.

VG-type fields

When the source field is a VG-type field, it represents variable-length double-byte character set (DBCS) data. This means that two bytes are used to represent one DBCS character. For a VG-type field, give the length in an even number of bytes from 2 through 254. DBCS data cannot be delimited by shift-out (X'0E') and shift-in (X'0F') characters in storage.

When the INTO keyword is omitted from the EXTRACT command, DataRefresher assumes that the data is targeted for a physical sequential data set or CMS file. In this case, when DataRefresher extracts data from a VG-type field, shift-out and shift-in characters are included for all DBCS data. The source data from a VG-type field is assumed to be all DBCS data (not mixed).

For relational databases, the shift-out and shift-in characters are not included with DBCS data, because SQL/DS and DB2 assume that VG-type fields are totally DBCS data and not mixed. If DBS=IXF, shift-out (X'0E') and shift-in (X'0F') characters are placed around all extracted GRAPHIC data, even if an INTO keyword is specified.

When extracted for a relational or target in IXF, VC-type and VG-type fields map to the corresponding variable-length columns or fields in the target.

A VC-type or VG-type field may have an accompanying length field of type F, H, B, P, or Z (zero scale for P or Z) that specifies the size in bytes of the VC-type or VG-type field. The length field is coded on the LFIELD keyword in the DataRefresher file or DXTPSB description.

The BYTES keyword, coded in the DataRefresher file or DXTPSB description, specifies the maximum size of a VC-type or VG-type field. If not indicated by a length field, the size of a VC-type or VG-type field is computed as the record or segment length plus one minus the starting position of the VC-type or VG-type field.

How fields are represented in source format

When you request that DataRefresher extract data and keep it in source format (rather than convert it to EBCDIC), the data is extracted and presented in exactly the same way as it was entered in source. There are, however, three exceptions to this rule:

- Zoned decimal fields are *always* converted to packed decimal.
- When the target database is SQL/DS, single precision floating point fields are converted to double precision floating point.
- Date, time, and timestamp type fields are always output in full character ISO (if you are using the DEM to extract data), regardless of the format specified on the EXTRACT statement of the SUBMIT (UIM) command. (See "A-type, T-type, and S-type fields" on page 261 for the definition of full character ISO.)

How IXF records are represented

There are seven different record types defined for IXF. DataRefresher generates the four types supported in IXF Version 0, the first IXF implementation. Three of these record types are informational or descriptive. They define the file in IXF and the data it contains. These records are always in character format, and numeric values are right-justified.

The data itself may be in character or machine (mixed binary and character) format. DBCS data is not converted to character format. See Appendix G, "Integration Exchange Format (IXF) record descriptions" on page 267 for a detailed explanation of each record type, and a discussion of IXF Version 0.

Appendix G. Integration Exchange Format (IXF) record descriptions

Integration Exchange Format (IXF) is an output format that lets other application programs have access to extracted data.

The advantage of files in IXF is that they are self-defined. The header information completely defines the data that follows and therefore another application program can understand data when it is passed in IXF.

DataRefresher supports IXF Version 0. IXF Version 0 is a subset of the fully defined IXF. (See "Summary of IXF Version 0" on page 285 for a list of IXF Version 0 characteristics.)

This chapter presents a declaration in a PL/I-like language for records in IXF, a detailed description of each of these records, and examples. Throughout the chapter considerations for IXF Version 0 are noted. A section on DataRefresher-specific information for IXF is also included.

IXF record formats

IXF as defined does not have dependencies on the type of file used to contain it. The declarations are a mapping of the portion of the record in a file that contains data, and could be used with either fixed-length or variable-length records. However, variable-length records may save space, making them preferable to fixed-length records.

IXF is comprised of seven different record types, five of which are considered header or informational records and two that are considered data records.

Header records are in all-character format for compatibility with the PC and for readability; numeric values are right-justified and are represented in character format. Header records are:

- H-type
- T-type
- C-type
- F-type
- A-type

Data records can be in either all-character or machine (mixed binary and character) format, but DBCS data is not converted to character format. Data records are:

- D-type
- X-type

The convention for the order of the header records is:

- One H-type record is required. This is a header record that consists of general information about the file.
- One group of T-, C-, and F-type records.
  - One T-type record is required. This is a table record that consists of data and table information about the file.

  - One C-type record for each column in the data (if data is not columnar, there are no C-type records) in order of ascending IXFCPOSN in IXF Version 0.
  - One F-type record (if data resides in an external file). This record type is not supported in IXF Version 0.
- Any number of A-type records, according to the application, may appear wherever you choose to place them, inside or outside T-C-F groups, but not within the data records. This record type is not supported by IXF Version 0.
- D-type and X-type records (if data resides in the same file as the header, as it always will in IXF Version 0). D-type records are the data records. X-type records do not exist in IXF Version 0.

The following figure shows an example of IXF record formats.

```
DECLARE
/*-- HEADER RECORD --- GENERAL INFORMATION --- ONE PER IXF----------*/
1 IXFHREC,                          /* HEADER RECORD (MANDATORY)     */
  2 IXFHRECT CHAR(01),              /* RECORD TYPE = H               */
  2 IXFHID   CHAR(03),              /* IXF ID: 'IXF'                 */
  2 IXFHVERS CHAR(04),              /* VERSION OF IXF USED           */
  2 IXFHPROD CHAR(12),              /* PRODUCT ORIGINATING HEADER    */
    3 IXFHSPRD CHAR(06),            /* ORIGINATING PRODUCT NAME      */
    3 IXFHSREL CHAR(06),            /* ORIGINATING PRODUCT RELEASE   */
  2 IXFHDATE CHAR(08),              /* DATE WRITTEN (YYYYMMDD)       */
  2 IXFHTIME CHAR(06),              /* TIME WRITTEN (HHMMSS)         */
  2 IXFHHCNT CHAR(05),              /* NUMBER OF RECORDS BEFORE DATA */
  2 IXFHDBCS CHAR(01),              /* DBCS DATA POSSIBLE: Y/N       */
  2 *        CHAR(02),              /* RESERVED                      */

/*-- TABLE RECORD --- DATA/TABLE INFORMATION --- ONE PER IXF--------*/
1 IXFTREC,                          /* DATA/TABLE RECORD             */
  2 IXFTRECT CHAR(01),              /* RECORD TYPE = T               */
  2 IXFTNAML CHAR(02),              /* DATA NAME LENGTH              */
  2 IXFTNAME CHAR(18),              /* NAME OF DATA                  */
  2 IXFTQUAL CHAR(08),              /* QUALIFIER                     */
  2 IXFTSRC  CHAR(12),              /* DATA SOURCE DESCRIPTION       */
  2 IXFTDATA CHAR(01),              /* DATA CONVENTION: C = COLUMNAR */
                                    /*                  A = APL CDR  */
  2 IXFTFORM CHAR(01),              /* DATA FORMAT: M = MACHINE      */
                                    /*              C = CHARACTER    */
  2 IXFTLOC  CHAR(01),              /* DATA LOCATION: I = INTERNAL   */
                                    /*                E = EXTERNAL   */
  2 IXFTCCNT CHAR(05),              /* NUMBER OF COLUMN DESCRIPTORS  */
                                    /*   ('C' RECORDS) FOR TABLE     */
  2 *        CHAR(02),              /* RESERVED                      */
  2 IXFTDESC CHAR(30),              /* DESCRIPTION OF DATA           */

/*-- COLUMN DESCRIPTOR RECORD --- ONE PER COLUMN OF TABLE ----------*/
1 IXFCREC,                          /* COLUMN DEFINITION             */
  2 IXFCRECT CHAR(01),              /* RECORD TYPE = C               */
  2 IXFCNAML CHAR(02),              /* COLUMN NAME LENGTH            */
  2 IXFCNAME CHAR(18),              /* COLUMN NAME                   */
  2 IXFCNULL CHAR(01),              /* COLUMN ALLOWS NULLS:   Y/N    */
  2 IXFCSLCT CHAR(01),              /* SELECTED COLUMN FLAG:  Y/N    */
  2 IXFCKEY  CHAR(01),              /* KEY COLUMN:            Y/N    */
  2 IXFCCLAS CHAR(01),              /* DATA CLASS                    */
  2 IXFCTYPE CHAR(03),              /* DATATYPE                      */
  2 IXFCCODE CHAR(05),              /* CODE PAGE                     */
  2 *        CHAR(05),              /* RESERVED FOR CODE ID          */
  2 IXFCLENG CHAR(05),              /* COLUMN DATA WIDTH/LENGTH      */
    3 IXFCPRCS CHAR(03),            /* PRECISION                     */
    3 IXFCSCAL CHAR(02),            /* SCALE                         */
  2 IXFCPOSN CHAR(06),              /* STARTING POSITION             */
  2 IXFCDESC CHAR(30),              /* COLUMN DESCRIPTION / LABEL    */
  2 IXFCNDIM CHAR(02),              /* NUMBER OF DIMENSIONS          */
  2 IXFCDSIZ(*) CHAR(05),           /* SIZE OF EACH DIMENSION        */
```

Figure 9 (Part 1 of 2). Example IXP record formats

```
/*-- EXTERNAL FILE RECORD (NOT IN IXF V0)----------------------------*/
1 IXFFREC,                          /* EXTERNAL FILE RECORD          */
  2 IXFFRECT CHAR(01),              /* RECORD TYPE = F               */
  2 IXFFFILE CHAR(44),              /* IDENTIFIER OF EXTERNAL FILE:  */
                                    /*  MVS: CATALOGUED DSNAME       */
    3 IXFFUSER CHAR(08),            /*  VM: USERID                   */
    3 IXFFDISK CHAR(03),            /*  VM: MINIDISK                 */
    3 IXFFFN   CHAR(08),            /*  VM: FILENAME                 */
    3 IXFFFT   CHAR(08),            /*  VM: FILETYPE                 */
    3 IXFFFM   CHAR(02),            /*  VM: FILEMODE                 */
  2 IXFFDFPX CHAR(01),              /* DATA RECORDS HAVE PREFIX: Y/N */
  2 IXFFHCNT CHAR(05),              /* NUMBER OF RECORDS BEFORE DATA */


/*-- APPLICATION RECORD --(NOT IN IXF V0)----------------------------*/
1 IXFAREC,                          /* APPLICATION RECORD            */
  2 IXFARECT CHAR(01),              /* RECORD TYPE = A               */
  2 IXFAPPID CHAR(12),              /* APPLICATION IDENTIFIER        */
  2 IXFADATA CHAR(*);               /* APPLICATION DATA              */


/*-- DATA RECORD - NO CONTINUATION FOLLOWS --------------------------*/
1 IXFDDATA,                         /* DATA RECORD, WITHOUT A        */
                                    /*  CONTINUATION FOLLOWING.      */
  2 IXFDPRFX,                       /* DATA RECORD PREFIX            */
    3 IXFDRECT CHAR(01),            /* RECORD TYPE ID: 'D'           */
    3 *        CHAR(02),            /* RESERVED                      */
    3 *        CHAR(01),            /* RESERVED                      */
    3 IXFDSO   CHAR(01),            /* CONTAINS SHIFT-OUT CHAR IF    */
                                    /*   THIS IS A CONTINUATION, AND */
                                    /*   SET TO 'Y'. BLANK OTHERWISE.*/
  2 IXFDCOLS CHAR(*),               /* DATA PORTION OF DATA RECORD   */

/*-- DATA RECORD - CONTINUATION FOLLOWS   (NOT IN IXF V0)-----------*/
1 IXFXDATA,                         /* DATA RECORD WITH A            */
                                    /*  CONTINUATION FOLLOWING.      */
  2 IXFXPRFX,                       /* DATA RECORD PREFIX            */
    3 IXFXRECT CHAR(01),            /* RECORD TYPE ID: 'X'           */
    3 *        CHAR(02),            /* RESERVED                      */
    3 IXFXSI   CHAR(01),            /* 'Y' IF SHIFT-IN CHARACTER IS  */
                                    /*   PRESENT AT END OF RECORD TO */
                                    /*   BALANCE SHIFT-IN (MEANS     */
                                    /*   CONTINUATION HAS SHIFT-OUT  */
                                    /*   IN FIELD IXFDSO OR IXFXSO)  */
    3 IXFXSO   CHAR(01),            /* CONTAINS SHIFT-OUT CHAR IF
                                    /*   THIS IS A CONTINUATION, AND */
                                    /*   PREVIOUS RECORD HAS IXFXSI  */
                                    /*   SET TO 'Y'. BLANK OTHERWISE.*/
  2 IXFXCOLS CHAR(*);               /* DATA PORTION OF DATA RECORD   */


/*-- END OF IXF -----------------------------------------------------*/
```

Figure 9 (Part 2 of 2). Example IXP record formats

IXF record descriptions

IXF Version 0 includes a subset of the full IXF capabilities. In the following descriptions, if a field or record is *not* used in IXF Version 0, it is so noted. Special considerations for IXF Version 0 are also noted.

When creating a file in IXF, observe the following conventions:

- Use blanks in the initialization of:
  - Unnamed areas of the records (denoted by "3 *...".)

- Named areas which may not be used in particular circumstances (such as IXFCLENG, IXFCPRCS and IXFCSCAL, for data types where length information is implied by the data type)

- When placing values that represent numeric quantities in H, T, and C type records:
  - Right-justify the value if it does not fill the character space for field.
  - Use either leading blanks or leading zeros to fill the area.
  - If a numeric zero is intended for one of these fields, at least one right-justified character zero must be used. The area may not be left entirely blank. For example, a zero scale for a decimal column must be expressed as '0' or '00', not as ' ', in field IXFCSCAL.
  - The IXF Version 0 fields to which these rules apply are: IXFHHCNT, IXFTCCNT, IXFCNAML, IXFCLENG, IXFCPRCS, IXFCSCAL, IXFCPOSN.

Header record descriptions

IXFHREC (Header Record - General Information)
is used to specify general information about the file in IXF. Only one Header Record is present per IXF file, and record specification is required.

IXFHRECT
is the record type; it is set to `H` for Header Record.

IXFHID
identifies the current file as a file in IXF; it is set to the value `IXF`.

IXFHVERS
is the level of the IXF used when the file is created; it is set to the value `0000`.

IXFHPROD
may be used by the creator of the file to identify itself as a product or an application.

IXFHSPRD
identifies the product that creates the file.

IXFHSREL
identifies the release of the product; for example, R1 or V2R1.

IXFHDATE
holds the date the file was written, in the format *YYYYMMDD*.

IXFHTIME
holds the time the file was written, in the format *HHMMSS*. This field is optional, since all products may not have the same facilities for obtaining the time.

IXFHHCNT
contains a count of the records that precede the first data record in the file. (If the table record field IXFTLOC is equal to `E`, then this value is the total number of records in the file.)

IXFHDBCS
is set either to `Y`, to indicate that DBCS data is a possibility, or to `N`, if DBCS data will not occur.

Except in IXF Version 0, if DBCS data can exist there is a chance that a record has been split in the middle of DBCS data. See the discussion of the IXFXSI and IXFXSO fields in the IXFXDATA data record below.

Also, if IXFHDBCS is set to Y, it may be that a character field has embedded DBCS data. This may have system or device implications for the receiving system. For example, the receiving system may not support such embedded DBCS, and may have to take special action or declare the instances of data containing DBCS to be invalid.

IXFTREC (table record - data/table information)
is used to specify data and table information. Only one table record is present per table, and record specification is required.

IXFTRECT
is the record type; it is set to T for table record.

IXFTNAML
contains the actual length of the name specified in IXFTNAME, below. The full length (18) may be used if it is not required or desired to distinguish between a given name and the same name with trailing blanks.

IXFTNAME
specifies the name of the data. IXFTNAME can be the table name if the data comes from a relational database. This field should be provided so that the receiver has a name for the data.

IXFTQUAL
is the table name qualifier, which identifies the creator of the table in a relational system. This field is optional.

IXFTSRC
can be used optionally to indicate the original source of the data; for example, DB2V1R1 or DB2/STLMVS1.

IXFTDATA
is set to C for columnar data. Individual column attributes are described in the column records that follow. Data follows IXF convention.

For IXF Version 0, only the 'C' (columnar data) value is supported.

IXFTFORM
indicates the convention used to store the data. Currently defined conventions are:

M Machine or internal format. Numeric data is in IBM S/370* internal form. A complete description follows in "Data record descriptions" on page 277.

C Character or external format. All data is represented in character format. A complete description follows in "Data record descriptions" on page 277.

Both machine and character formats are supported in IXF VERSION 0.

* S/370 is a trademark of the International Business Machines Corporation.

IXFTLOC
indicates the location of the data:

I means that data is in this file.

E means that the data is in an external file whose location is described in a subsequent F-type record.

For IXF VERSION 0, only the 'I' value (data internal to the file) is supported.)

IXFTCCNT
indicates how many column descriptors (C-type records) are present for the table.

IXFTDESC
is used to provides any desired descriptive information about the data. This field is optional.

IXFCREC (Column Descriptor Record)
is used to describe a given table column, with one occurrence per column, if the table record field IXFTDATA is equal to C. Because only columnar data is supported in IXF VERSION 0, the IXFCREC record is required for IXF VERSION 0.

IXFCRECT
is the record type; it is set to C for column record.

IXFCNAML
indicates the actual length of the name specified in IXFCNAME, below. The full length (18) may be used if it is not required or desired to distinguish between a given name and the same name with trailing blanks.

IXFCNAME
indicates the name of the column.

IXFCNULL
indicates whether nulls are allowed in this column. Valid settings are Y (yes) or N (no).

IXFCSLCT
allows selection of a subset of the columns in the data:

Y means that the column *is* selected.

N means that the column is *not* selected.

By manipulating the values in this column, it is possible to pass different data to one or more applications, using the same IXF file.

For IXF VERSION 0, only the 'Y' value is supported.

IXFCKEY
indicates whether this column is a key column:

Y means that the column *is* a key column.

N means that the column is *not* a key column.

A key for a table is a set of columns that identify a unique row of the table. Each column of that set is a key column.

IXFCCLAS
defines the class of data types to be used in IXFCTYPE, below. Classes currently defined are:

R Relational. Data types used are within the set of data types defined by the relational products SQL/DS and DB2.

M Machine. In addition to the relational data types, other common data types have been defined.

For IXF VERSION 0, only the R (relational) value is supported.

IXFCTYPE
indicates the data type. See "Data record descriptions" on page 277 for a list of the values currently defined for this field, along with a detailed discussion of the data format for each data type.

IXFCCODE
indicates the code page or character set to be used. This field is added for future support of national languages and should be set to 00000 to use the default character set. If a column contains bit data, this field should be set to FFFFF.

IXFCLENG
indicates the length or size of this column. The meaning of this field varies according to data type:

- For some data types, IXFCLENG has meaning.
- For other data types, the subfields have meaning.
- For still other data types, the length is inherent in the data type.

For more information, see "Data record descriptions" on page 277.

IXFCPRCS
a subfield of IXFCLENG, above, that indicates the precision or length of numeric data types that permit scale. See "Data record descriptions" on page 277 for more details.

IXFCSCAL
a subfield of IXFCLENG, above, that indicates the scale (number of digits after the decimal point) of numeric data types that permit scale. See "Data record descriptions" on page 277 for more details.

IXFCPOSN
is the starting position of this column:

- If the column *does* allow nulls, this field will point to the null indicator.
- If the column does *not* allow nulls, it will point to the data itself.

Thus, columns that do not allow nulls may or may not have space physically left for the null indicator, as desired. The starting position is based from the first byte that contains data. The Data Record prefixes (IXFDPRFX and IXFXPRFX) are simply not included in any consideration of starting position. The first data position is position 1 (not position 0).

IXFCDESC
is for descriptive information about the column (optional).

IXFCNDIM
indicates the number of dimensions in the column.

- If this field *is* equal to 0, the data is scalar (each row contains only one item of data).
- If this field is *not* equal to 0, then the data is an array with the number of dimensions given; the size or range of each dimension is given in IXFCDSIZ.

For an array, the data values are stored adjacent to one another. If the column can contain nulls (IXFCNULL=Y), then each data value has a null indicator immediately preceding it, and IXFCPOSN gives the position of the null indicator of the first data value. If the column cannot contain nulls, then IXFCPOSN gives the position of the first data value, and the leftmost position of the second data value is adjacent to the rightmost position of the first data value, etc.

For IXF VERSION 0, this field may either be blank or zero. Arrays are not supported in IXF VERSION 0.

IXFCDSIZ(*)
contains the size or range of each dimension. There are IXFCNDIM occurrences of this field. For example, if IXFCNDIM=001, and IXFCDSIZ(1)=00005, then this indicates a one-dimensional array with five elements. Or, if IXFCNDIM=002, IXFDSIZ(1)=00003, and IXFCDSIZ(2)=00005, then this indicates a two-dimensional array, with size 3 and 5 dimensions—in other words, a 3-by-5 matrix.

This field is not supported in IXF VERSION 0.

IXFFREC (External File Record)
has one occurrence per file in IXF if the IXFTLOC field of the table record is equal to E.

This record is not used in IXF Version 0.

IXFFRECT
indicates the record type; it is set to F for External File Record.

IXFFFILE
is the file identifier:

- In MVS, this field contains the fully qualified data set name. (The data set is presumed to be cataloged.)
- In CMS, this field is subdivided into the following fields:

  IXFFUSER
  is the userid. If this field is not blank, the file is identified by USERID/DISK/FN/FT. If this field *is* blank, the file is identified by FN/FT/FM. Users must be aware of password and linkage considerations in applications using external files for the data. Users should also be careful in choosing the appropriate way to identify the file.

  IXFFDISK
  identifies the minidisk.

IXFFFN
identifies the filename.

IXFFFT
identifies the filetype.

IXFFFM
identifies the filemode.

IXFFDPFX
indicates whether data records in the external file have prefixes:

Y means that data records have prefixes, as defined by IXFDPRFX/IXFXPRFX.

N means that no such prefix is present.

IXFFHCNT
is a count of leading records that precede the data in the external file. This allows applications to create IXF descriptions of existing files and skip the header information in those files.

IXFAREC (Application Record)
is not written in IXF VERSION 0 by any of the participating IBM products. Applications reading IXF files should be prepared to ignore 'A' records.

IXFARECT
is the record type; it is set to A for application record.

IXFAPPID
is an identifier for the application creating the file.

IXFADATA
can consist of any supplemental information the application wishes to include.

IXFDDATA (Data Record - No Continuation Follows)
is the data record in which record continuation *does not* follow.

IXFDPRFX
is the data record prefix:

IXFDRECT
is the record type; it is set to D, indicating that this is the last physical record containing data for the row of tabular data (it may also be the first, and therefore the only physical record for the row. This is always the case in IXF VERSION 0.)

IXFDSO
contains a shift-out character (X'0E') if this record is a continuation record, and if the IXFXSI field in the previous (continued) data record (IXFXDATA) contains a Y. Otherwise, it contains a blank. (This field will always be blank in IXF Version 0.) For more information, refer to the discussion for the field IXFXSI in the IXFXDATA data record described below.

IXFDCOLS
is the area for columnar data, as defined in the "Data record descriptions" on page 277.

IXFXDATA (Data Record - Continuation Follows)
is a data record in which record continuation *does* follow.

This record is not used in IXF VERSION 0.

IXFXPRFX
is the data record prefix:

IXFXRECT
record type is set to X, indicating that there is data for this row in the next physical record.

IXFXSI
is the DBCS shift indicator:

- is set to Y if, for the sake of data integrity in the event that a DBCS column was split across the two records, a shift-in character (X'0E') was added to the end of this row and a shift-out character (X'0F') was added to the IXFDSO or IXFXSO field in the beginning of the next (continuation) record.
- N means that no shift indicators were added.

If there is a possibility that the file will be sent to a DBCS-supporting device, this action should be taken if DBCS columns are split across records. Otherwise, the data stream may bring the device down.

IXFXSO
contains a shift-out character (X'0E') if it is a continuation record, and if the IXFXSI field in the previous continued data record contains a Y. Otherwise, it contains a blank.

IXFXCOLS
is the area for columnar data, as defined in "Data record descriptions."

Data record descriptions

For each column, the data consists of an optional null indicator (required if IXFCNULL=Y for the Column Descriptor Record for the column), followed by the data itself.

For data records, including external data as well as internal type D- and X-type records, the data for the *n* columns is placed side by side:

```
column-1 | column-2 | ... | column-n
```

For each column, the data consists of a null indicator followed by the data itself:

```
column-null-indicator | column-data
```

OR

```
column-data
```

Note: If IXFCNULL=Y, then the IXFCPOSN field of the Column Descriptor Record gives the position of the null indicator in the record. If IXFCNULL=N, then IXFCPOSN gives the position of the data itself in the record. In the latter case, the space for the null indicator still might be present in the data records.

The value in IXFCPOSN represents the position of the column null/data value within the concatenated IXFDCOLS and IXFXCOLS fields (of the IXFDDATA and IXFXDATA records, respectively) for the data records making up a row of data. The first position is represented by an IXFCPOSN value of 1. The data record prefixes (IXFDPRFX and IXFXPRFX of the IXFDDATA and IXFXDATA records, respectively) are simply ignored in computing the position represented by an IXFCPOSN value.

In general, the format of the null indicator and the data depends on the value of the table record variable IXFTFORM:

C Character format

M Machine format

Format of column null indicator

When IXFTFORM=C (character format), one byte is used for the null indicator:

- Data is null.

blank Data is not null.

When IXFTFORM=M (machine format), two bytes are used for the null indicator:

X'FFFF' Data is null.

X'0000' Data is not null.

Format of column data by data type

For each data type, the significance of the length fields (IXFCLENG–length, IXFCPRCS–precision, IXFCSCAL–scale) is covered, and the format of the data for both character (IXFTFORM=C) and machine (IXFTFORM=M) formats is defined.

Note: For the character formats of numeric data, the first position is generally reserved for a sign (+, -, or blank), unless the data type is unsigned. Leading zeros are explicitly included, rather than leaving them blank. For more information, see the examples for the particular data types.

- Fixed-length character format (IXFCTYPE=452):

  IXFCLENG is the length of the character string in bytes. The valid range of values for IXFCLENG is 1 to 254 inclusive. IXFCPRCS and IXFCSCAL are not used.

  – For IXFTFORM=C (character format):

    The *n* bytes indicated by IXFCLENG are included. Example:

    ```
    If IXFCLENG='00005':  'SMITH'
    ```

  – For IXFTFORM=M (machine format):

    The *n* bytes indicated by IXFCLENG are included. Example:

    ```
    If IXFCLENG='00005':  'SMITH'
    ```

- Variable-length character format (IXFCCLAS=R, IXFCTYPE=448):

  IXFCLENG is the maximum length of the character string in bytes. The valid range of values for IXFCLENG is 1 to 254 inclusive. IXFCPRCS and IXFCSCAL are not used.

  – For IXFTFORM=C (character format):

The *n* bytes indicated by IXFCLENG are included, preceded by a 5-byte character count field. The number of characters indicated by the count field are valid; the rest might contain garbage. Example:

```
If IXFCLENG='00010':  '00005SMITHggggg'

where each g may be garbage
```

- For IXFTFORM=M (machine format):

  The *n* bytes indicated by IXFCLENG are included, preceded by a 2-byte binary count field. The number of characters indicated by the count field are valid; the rest might contain garbage. Example:

```
If IXFCLENG='00010':  'xxSMITHggggg'

where xx = X'0005', and each g may be garbage
```

- Long variable-length character (IXFCCLAS=R, IXFCTYPE=456):

  Both the significance of the length fields and the format of the data are the same as for Variable-Length Character. However, the valid range of values for IXFCLENG is 1 to 32767 inclusive.

- Fixed-length graphic (IXFCCLAS=R, IXFCTYPE=468):

  IXFCLENG is the number of double-byte characters (taking 2*n bytes). The valid range of values for IXFCLENG is 1 to 127 inclusive. IXFCPRCS and IXFCSCAL are not used.

  - For IXFTFORM=C (character format):

    The 2*n bytes indicated by IXFCLENG are included, plus a shift-out (X'0E') immediately preceding the data, and a shift-in (X'0F') immediately following the data. These bytes enable DBCS-devices to recognize shifts in mode between single- and double-byte character sets. Note that in the data, no attempt is made to convert any characters to EBCDIC. Example:

```
If IXFCLENG='00005':  'oZZYYXXWWVVi'

where: o is the shift-out character
       i is the shift-in character
       ZZYYXXWWVV is graphic or DBCS data
```

  - For IXFTFORM=M (machine format):

    The 2*n bytes indicated by IXFCLENG are included. No shift-out or shift-in characters are included. Example:

```
If IXFCLENG = '00005':  'ZZYYXXWWVV'

where ZZYYXXWWVV is graphic or DBCS data
```

- Variable-length graphic (IXFCCLAS=R, IXFCTYPE=464):

  IXFCLENG is the maximum number of double-byte characters (taking 2*n bytes). The valid range of values for IXFCLENG is 1 to 127 inclusive. IXFCPRCS and IXFCSCAL are not used.

  - For IXFTFORM=C (character format):

    The 2*n bytes indicated by IXFCLENG are included, preceded by a 5-byte character count field. The number of 2-byte characters indicated by the count field is valid, plus a shift-out character (X'0E') immediately preceding the data, and a shift-in character (X'0F') immediately following the data.

These bytes enable DBCS-devices to recognize shifts in mode between single- and double-byte character sets. The rest of the data may contain garbage. Note that in the data, no attempt is made to convert any characters to EBCDIC. Example:

```
If IXFCLENG = '00010':  '00005oZZYYXXWWVVigggggggggg'

where: o is the shift-out character
       i is the shift-in character
       ZZYYXXWWVV is graphic or DBCS data
       each g may be garbage
```

- For IXFTFORM=M (machine format):

  The 2*n* bytes indicated by IXFCLENG are included, preceded by a 2-byte binary count field. The number of 2-byte characters indicated by the count field included. No shift-out or shift-in characters are included. The rest of the data may contain garbage. Example:

```
If IXFCLENG='00010':  'xxZZYYXXWWVVgggggggggg'

where: xx = X'0005'
       ZZYYXXWWVV is graphic or DBCS data
       each g may be garbage
```

- Long variable-length graphic (IXFCCLAS=R, IXFCTYPE=472):

  Both the significance of the length fields, and the format of the data are the same as for variable-length graphic. However, the valid range of values for IXFCLENG is 1 to 16383 inclusive.

- A 2-byte Signed Binary (IXFCCLAS=R, IXFCTYPE=500):

  None of the fields IXFCLENG, IXFCPRCS, or IXFCSCAL have significance; the length is inherent in the data type.

  - For IXFTFORM=C (character format):

    A 6-byte character value, right justified, with the first character reserved for a sign, is included. Example:

```
' 00023'
'-21311'
'+00762'
```

  - For IXFTFORM=M (machine format):

    The 2-byte binary value is included.

- A 4-byte signed binary (IXFCCLAS=R, IXFCTYPE=496):

  None of the fields IXFCLENG, IXFCPRCS, or IXFCSCAL have significance; the length is inherent in the data type.

  - For IXFTFORM=C (character format):

    An 11-byte character value, right justified, with the first character reserved for a sign, is included. Example:

```
' 0000000001'
'-1180000000'
'+0033599708'
```

  - For IXFTFORM=M (machine format):

    The 4-byte binary value is included.

- Packed decimal (IXFCCLAS=R, IXFCTYPE=484):

  IXFCLENG has no significance, but IXFCPRCS and IXFCSCAL give the total number of digits (precision *P*) and the number to the right of the decimal point (scale *S*), respectively. The valid range for of values for IXFCPRCS is 1 to 31 inclusive. IXFCSCAL may assume a value from 0 to the value of IXFCPRCS inclusive.

  - For IXFTFORM=C (character format):

    The data is formatted as a *P*+2 byte character value (or *P*+1 bytes if *S* is equal to zero), right justified, with the first byte reserved for a sign, and a decimal point (position implied by *S*) present if and only if *S* is not equal to zero. Examples:

    ```
    If IXFCPRCS = '005', IXFCSCAL = '00':  ' 12345'
    If IXFCPRCS = '006', IXFCSCAL = '02':  '+2345.10'
    If IXFCPRCS = '004', IXFCSCAL = '03':  '-8.515'
    ```

  - For IXFTFORM=M (machine format):

    A (*P*+2)/2 byte packed decimal value, in standard IBM/370 packed decimal format, with *S* of the *P* digits interpreted as following the implied decimal point, is included. Examples:

    ```
    If IXFCPRCS = '005', IXFCSCAL = '00':  X'012345C'
    If IXFCPRCS = '006', IXFCSCAL = '02':  X'0234510D'
    ```

- Floating point (IXFCCLAS=R, IXFCTYPE=480):

  IXFCLENG is used to distinguish long (IXFCLENG = '00008') from short (IXFCLENG = '00004') floating point. For compatibility with earlier IXF, a blank IXFCLENG should be interpreted as '00008'. No other IXFCLENG values are valid. IXFCPRCS and IXFCSCAL are not used.

  - For IXFTFORM=C (character format):

    If IXFCLENG = '00008', a 23-byte character value in PL/I style scientific notation is included: 1 character for sign of data, an 18-character mantissa (17 digits and a decimal point between the first and the second), the character E, and finally a 3-character signed exponent (a sign followed by two digits). Examples:

    ```
    '-1.2345678901234567E+14'
    '+6.2345678901234567E-01'
    ' 0.0000000000000000E+00'
    ```

    If IXFCLENG = '00004:', a 14-byte character value in PL/I style scientific notation is included: 1 character for sign of data, a 9-character mantissa (8 digits and a decimal point between the first and the second), the character 'E', and finally a 3-character signed exponent (a sign followed by two digits). Examples:

    ```
    '-1.2345678E+14'
    '+6.2345678E-01'
    ' 0.0000000E+00'
    ```

  - For IXFTFORM=M (machine format):

    A 4-byte or 8-byte floating point value (depending on IXFCLENG), in standard IBM/370 format for floating-point, is included.

- Date (IXFCCLAS=R, IXFCTYPE=384):

  The DB2 full ISO format is to be used, with a length of 10 characters (bytes) determined by the data type. IXFCLENG has no significance. IXFCPRCS and IXFCSCAL are not used.

  – For IXTFORM=C (character format):

    The format (yyyy-mm-dd) is used, where yyyy is the year, mm the month, and dd the day. All are numeric characters. Leading zeroes may not be omitted from the year, month, or day. (Some ISO formats, acceptable to DB2 as input, are not acceptable here, such as '1963-8-13'.) The year can be between 0000 and 9999 inclusive. The month and day should be valid dates for the year (observing leap year conventions, etc.). Examples:

    ```
    '1993-09-08'  represents September 08, 1993
    '2000-11-01'  represents November 01, 2000
    ```

  – For IXTFORM=M (machine format)

    Same as character format.

- Time (IXFCCLAS=R, IXFCTYPE=388):

  The DB2 full ISO format is to be used, with a length of 8 characters (bytes) determined by data type. IXFCLENG has no significance. IXFCPRCS and IXFCSCAL are not used.

  – For IXTFORM=C (character format):

    The format (hh.mm.ss) is used, where hh is the hour, mm the minutes, and ss the seconds. All are full numeric characters. Leading zeroes may not be omitted from the hour, minutes, or seconds. (Some ISO formats, acceptable to DB2 as input, are not acceptable here, such as '4.42.00'.) The hour can be any between 00 and 23 inclusive. (Note that the special value 24.00.00 is also valid.) Examples:

    ```
    '23.59.59'  represents 11:59:59 PM
    '08.45.00'  represents 8:45 AM
    ```

  – For IXTFORM=M (machine format):

    Same as character format.

- Timestamp (IXFCCLAS=R, IXFCTYPE=392):

  The full DB2 format is to be used, with a length 26 characters (bytes) determined by data type. IXFCLENG has no significance. IXFCPRCS and IXFCSCAL are not used.

  – For IXTFORM=C (character format):

    The format (yyyy-mm-dd-hh.mm.ss.nnnnnn) is used, where yyyy is the year, the first mm the month, dd the day, hh the hour, the second mm the minutes, ss the seconds, and nnnnnn the microseconds. Truncated formats, accepted by DB2 as input, are not valid here. Valid ranges for the component subfields are as defined above for Date and Time data types. Examples:

    ```
    '1994-12-31-23.59.59.999999' is the
        last microsecond in 1994
    '1995-01-01-00.00.00.000000' is the
        first microsecond in 1995
    ```

Like time type data, the special value 24.00.00.000000 is also valid for the time portion of a timestamp.

- For IXTFORM=M (machine format):

  Same as character format.

- Unsigned binary (IXFCCLAS=M, IXFCTYPE=020):

  IXFCLENG is the length of the field, from 1 to 4 inclusive. Neither IXFCPRCS nor IXFCSCAL is significant.

  - For IXFTFORM=C (character format):

    A character string of the length indicated in the examples is included:

    ```
    If IXFCLENG = '00001': 3 characters,
       example: '254'
    If IXFCLENG = '00002': 5 characters,
       example: '00318'
    If IXFCLENG = '00003': 8 characters,
       example: '16110865'
    If IXFCLENG = '00004':10 characters,
       example: '3118000000'
    ```

  - For IXFTFORM=M (machine format):

    The *n* byte binary string is included.

- Signed binary with scale (IXFCCLAS=M, IXFCTYPE=028):

  IXFCLENG has no significance. IXFCPRCS is the length of the field, either 2 or 4. IXFCSCAL is the number of characters to the right of the presumed decimal point. (Note that IXFCPRCS and IXFCSCAL deal with different quantities in this case—the former with binary bytes, and the latter with decimal digits.)

  - For IXFTFORM=C (character format):

    A character string of the length indicated in the examples is included, with the leftmost character reserved for a sign, and the decimal point positioned as indicated by IXFCSCAL. Note that when IXFCSCAL is 0, the formatted data is shorter. Examples:

    ```
    If IXFCPRSC = '002', IXFCSCAL = '02':
        7 chars, example: '-003.18'
    If IXFCPRSC = '002', IXFCSCAL = '00':
        6 chars, example: '-00318'
    If IXFCPRSC = '004', IXFCSCAL = '05':
       12 chars, example: ' 31180.00000'
    If IXFCPRSC = '004', IXFCSCAL = '00':
       11 chars, example: ' 3118000000'
    ```

  - For IXFTFORM=M (machine format):

    The *n*-byte binary string is included, with the decimal point implied by IXFCSCAL, and the sign indicated by the data. Examples:

    ```
    If IXFCPRSC = '002', IXFCSCAL = '00': X'001C'
    If IXFCPRSC = '004', IXFCSCAL = '05': X'001C39A0'
    ```

- Short floating point (IXFCCLAS=M, IXFCTYPE=036):

  None of the fields IXFCLENG, IXFCPRCS, or IXFCSCAL have significance; this information is inherent in the data type and data format.

  - For IXFTFORM=C (character format):

A 14-byte character value in PL/I style scientific notation is included: 1 character for the sign of the data, a 9-character mantissa (8 digits, with a decimal point between the first and the second), the character E, and finally a 3-character signed exponent (a sign followed by two digits). Examples:

```
'-1.2345678E+14'
'+6.2345678E-01'
' 0.0000000E+00'
```

- For IXFTFORM=M (machine format):

  A 4-byte floating point value is included, in standard IBM/370 format for short floating-point values.

- Zoned decimal with scale (IXFCCLAS=M, IXFCTYPE=044):

  IXFCLENG is not significant. IXFCPRCS is the number of digits in the data (precision *P*); IXFCSCAL is the number of digits to the right of the decimal point (scale *S*).

  - For IXFTFORM=C (character format):

    Included is a *P*+2-byte character value (or *P*+1 bytes if *S* is equal to 0), right justified, with the first byte reserved for a sign, and a decimal point (position implied by *S*) present if, and only if, *S* is not equal to 0. Examples:

    ```
    If IXFCPRCS = '005', IXFCSCAL = '00': ' 12345'
    If IXFCPRCS = '006', IXFCSCAL = '02': '+2345.10'
    If IXFCPRCS = '004', IXFCSCAL = '03': '-8.515'
    ```

  - For IXFTFORM=M (machine format):

    The data is formatted in standard IBM/370 zoned decimal format, *n* bytes (IXFCPRSC) in length, with decimal point position implied by IXFCSCAL, and the sign included as the high-order 4 bits of the low-order byte. Examples:

    ```
    If IXFCPRCS = '005',IXFCSCAL = '00': X'F1F2F3F4C5'
    If IXFCPRCS = '006',IXFCSCAL = '02': X'F2F3F4F5F1D0'
    ```

- Bit string (IXFCCLAS=M, IXFCTYPE=052):

  IXFCLENG is the number of consecutive bits in the string, and neither IXFCPRCS nor IXFCSCAL has significance.

  - For IXFTFORM=C (character format):

    A string of *n* characters is included (*n* given by IXFCLENG), each of which is a `0` or `1`. Example:

    ```
    IXFCLENG = 00005': '10010'
    ```

  - For IXFTFORM=M (machine format):

    Included are *n*+7/8 bytes of bit string data (*n* given by IXFCLENG). Only the first *n* bits are valid, the remaining bits may contain arbitrary values. The bits are consecutive, beginning with the first bit (bit 0) of the first byte.

IXF example

The following example has "xxxx" at the beginning of the data records. These fields are unprintable length fields which are represented by hexadecimal values in the actual output. Note that the "xxxx" will not be present in CMS files in IXF; CMS variable-length files do not record the length of their records. The following figure shows partial output in IXF when data is extracted from a DEM data source.

```
xxxxHIXF0000DVR   V1R0  19940218104810000008N
xxxxT18DEPTSUMM                              CCI00006
xxxxC18DEPTNBR            YYNR45200000     00002000001                      00
xxxxC18DEPTNAME           YYNR45200000     00010000004                      00
xxxxC18SUMMDATE           YYNR38400000     00010000015                      00
xxxxC18DEPT#PERS          YYNR50000000          000026                      00
xxxxC18DEPT#SALES         YYNR49600000          000033                      00
xxxxC18DEPT$SALES         YYNR48400000     01300000045                      00
xxxxD     11 MILLINERY  1993-09-30  00009  0000000291  0000000111111
xxxxD     23 LEATHERS   1993-09-15  00007  0000000072  0000000012345
xxxxD     25 JEWELRY    ----------  00010  0000000901  0000002056000
xxxxD     50 AUTOMOTIVE 1993-10-06  00030  0000001025---------------
xxxxD     84 SPORTS EQ. 1993-09-22  00003  0000000170  0000000450980
xxxxD     94 STATIONERY 1993-09-16  00020  0000001871  0000001250000
xxxxD     99 APPLIANCES 1993-09-17 -00025 -0000000820 -0000000044400
```

Summary of IXF Version 0

Version 0 of IXF contains a subset of the full defined IXF. This subset consists of:

- A variable-length record implementation
- Both machine and character column data formats
- Columnar data only, with data types being those supported by the relational systems DB2 and SQL/DS
- No external data (which means no data files split from the file containing header records and no F-type records)
- No support for application records (A-type records)
- No arrays
- No support for continued data records (X-type); only D-type records are supported

DataRefresher support for IXF

DataRefresher supports IXF VERSION 0 as described in this chapter.

DataRefresher-specific considerations for IXF, as well as the values that are assigned to certain fields, are presented in this section.

Header record (IXFHREC) information

DataRefresher assigns values to the following fields:

- IXFHPROD -- Set to DVR V1R0.
- IXFHSPRD -- Set to DVR.
- IXFHSREL -- Set to V1R0.

Table record (IXFTREC) information

The DataRefresher considerations for the table record are:

IXFTNAML
DataRefresher always indicates that the length of the object name is 18.

IXFTNAME
If the INTO keyword of the EXTRACT statement is provided, DataRefresher writes the name of the table identified by the INTO keyword. If the INTO keyword is not provided, DataRefresher writes the name of the extract ID.

IXFTQUAL
If the INTO keyword of the EXTRACT statement is provided, and the table name in the INTO keyword is qualified, DataRefresher writes the qualifier. If the INTO keyword is not provided, or the table name in the INTO keyword is not qualified, DataRefresher leaves the field blank.

IXFTSRC
DataRefresher leaves this field blank.

IXFTFORM
Both machine and character formats as described above are supported by DataRefresher.

- If FORMAT=SOURCE, DataRefresher writes numeric data in machine format and this field contains an M.
- If FORMAT=EBCDIC, DataRefresher writes numeric data in character format and this field contains a C.

Column descriptor record (IXFCREC) information

The column descriptor record is required for DataRefresher, since only columnar data is currently supported. In addition, the following is true:

IXFCNAML
DataRefresher always indicates that the length of the column name is 18.

IXFCNAME
If the INTO keyword of the EXTRACT statement is provided, DataRefresher writes the name of a column identified by the INTO keyword. If the INTO keyword is not provided, DataRefresher writes the name of the FDTLIB source field name or the relational source column name.

IXFCNULL
DataRefresher sets the null flag based on the presence or absence of the NOT NULL value of the INTO keyword for each column:

- If the NOT NULL value is absent, DataRefresher sets this flag to Y for allowing nulls.
- If the NOT NULL value is present, DataRefresher sets this flag to N for NOT NULLS.
- If the INTO keyword itself is not present, DataRefresher sets this flag to Y for allowing nulls, on the assumption that other programs with access to IXF data will be capable of handling nulls in some manner.

IXFCKEY
DataRefresher always sets this flag to N, since it will not know if the column is a key column.

IXCFCPOSN
DataRefresher leaves space for a null indicator column before each data column. If the IXFCNULL field has an N (for "nulls not allowed" in the column, this value points to the start of the data. However, if the IXFCNULL field has a Y for "nulls permitted", this value points to the null indicator column.

IXFCDESC
DataRefresher leaves this field blank.

IXFCNDIM
DataRefresher sets this field to '00'.

IXFCDSIZ(*)
DataRefresher does not support this field.

Terms and abbreviations

This glossary defines terms and abbreviations as they are used in the DataRefresher library. Entries often include further information about how the term applies specifically to DataRefresher.

abend. Abnormal end of a task.

access control list (ACL). Defines the access rights to the associated data source in DataRefresher.

access level. The level of authority a user has when using a protected DataRefresher resource or command set.

Accounting exit routine. A user-written routine used for charging resources to individual users. This routine is started at the beginning and the end of the extract request execution cycle. It can be used to change output limits, change priorities of the DEM, and account for the resources used by the DEM on behalf of individual extract requests. This routine can be coded in Assembler, PL/I, or COBOL.

ACL. access control list.

Administrative Dialogs. A series of menus and displays that help a user create and submit data descriptions and extract requests, maintain the profiles for DataRefresher Dialogs and JCL for submitting data descriptions and extract requests, and administer End User Dialogs.

Advanced Program-to-Program Communications (APPC). The communication protocol (LU 6.2) that is used by DataRefresher.

alias. An alternate name for a member of a partitioned data set or for a name of a field described in a data description.

APAR. authorized program analysis report.

APPC. Advanced Program-to-Program Communications.

Application System (AS). An IBM* integrated decision support program that helps provide business planning, graphics, project control and management, statistical data analysis, and other functions.

AS. Application System.

asynchronous. Occurring without a regular or predictable time relationship.

ASCII. ANSI Standard Code for Information Interchange.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current, unaltered release of a program.

batch message processing (BMP). An IMS/VS region where batch message processing occurs.

BMP. batch message processing.

Boolean expression. In DataRefresher, a conditional expression that evaluates to true or false to determine whether a particular unit of data is extracted. The expression may contain multiple conditions connected by the logical operators AND, OR, and NOT.

card-image input. Input that simulates punched card input (80 columns per record).

CCU. Consistency Check Utility - a DataPropagator* NonRelational feature used with the DataRefresher UIM when using a DataPropagator NonRelational map capture exit in the DataRefresher SUBMIT command.

CEEPIPI. Common Execution Environment Pre-Initialized Program Interface.

child segment. In a database, a segment that lies immediately below its parent segment. A child segment has only one parent segment.

CLIST. command list.

CMS. conversational monitor system.

code page. An assignment of graphic characters and control function meanings to all code points.

code point. A 1-byte code representing one of 256 potential characters.

command list (CLIST). A data set or a member of a partitioned data set containing TSO commands that run sequentially in response to the EXEC command.

---

* DB2, DataPropagator, CICS, Language Environment, SAA, AD/Cycle, VTAM, Systems Application Architecture, and SQL/DS are trademarks of the International Business Machines Corporation.

command string. A language construct that represents one step in a sequence of steps that produce a DataRefresher command.

containing segment. A parent segment that contains one or more internal segments.

control blocks. Storage areas used to hold control information.

conversation. An exclusive use of an LU-to-LU session by two transaction programs using the APPC. This is a short logical connection that lasts only for the duration of one complete transaction. (Contrast with *session*).

conversational monitor system (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities.

DAP. Dictionary Access Program.

database format. The format of the data prior to any segment preprocessing or data exit manipulation. (Contrast with *FDTLIB format*).

database manager. A program that controls the user's data, ensuring security and data integrity in a multiple user environment. Examples include Information Management System/Virtual Storage (IMS/VS), IBM DATABASE 2 (DB2*), and Structured Query Language/Data System (SQL/DS*).

Database Management System (DBMS). A software system that controls the creation, organization, and modification of a database and access to the data stored within it.

data definition (DD) statement. A job control statement that describes the data sets associated with a specific job step.

data description. The description of a file, IMS/VS DL/I database, or any data source accessed by a user-written generic data interface (GDI) exit.

Data Dictionary. The IBM OS/VS DB/DC Data Dictionary is a central repository of information about data such as names, meaning, relationships to other data, origin, usage, and format.

data entry database (DEDB). An IMS/VS Fast Path database used to provide efficient access to large volumes of detailed data. Each DEDB can be partitioned, or divided into multiple "areas" for ease of access.

Data exit routine. A user-written routine to provide data verification. The routine can be used to process each record in source files and each segment in source databases. This routine can be coded in assembler, PL/I, or COBOL.

DataRefresher. An IBM program that extracts data from a source database or file and formats it for a target database or file.

Data Extract Manager (DEM). The DataRefresher program that extracts data from a VSAM file, a physical sequential file, an IMS/VS DL/I database, or any data source accessed by a generic data interface (GDI) exit.

data facility sort (DFSORT). An IBM product that works in conjunction with DataRefresher or other database products to sort data as it is being processed.

Data Language 1 (DL/I). The database management language for IMS/VS.

data propagation. The process of applying the changes to one set of data to the copy of that data in another database system.

Data Reformat Utility (DRU). This utility recombines 80-character record segments (used to transmit data from one system to another) into logical records.

Date/Time Conversion exit routine. A user-written routine to convert date/time data to ISO format. The routine can be used to process each field containing date/time data. It can be coded in assembler, PL/I, or COBOL.

DBCS. double-byte character set.

DBMS. Database Management System.

DB2. IBM DATABASE 2.

DD. data definition.

DEDB. data entry database.

DEM. Data Extract Manager.

DEM data source. The source data from which the DEM extracts data. (For example, an IMS/VS DL/I database, a VSAM or physical sequential data set, or data accessed by a user-written generic data interface (GDI) exit.)

DFSORT. data facility sort.

Dictionary Access Program (DAP). The DataRefresher program that generates descriptions of the files and nonrelational databases from which users extract data. The descriptions are taken from existing definitions in the IBM Data Dictionary.

DL/I. Data Language I.

double-byte character set (DBCS). A set of characters in which each character occupies 2 bytes. Languages such as Japanese, Chinese, and Korean that contain more symbols thatn can be represented by 256 code points require double-byte character sets. Entering, displaying, and printing DBCS characters requires special hardware and software support.

DRU. Data Reformat Utility.

DataRefresher dialogs. The DataRefresher programs that help users build and send data descriptions (Administrative Dialogs) and extract requests (End User Dialogs and Administrative Dialogs).

DXTFILE. A DataRefresher object stored in the FDTLIB that describes a SAM, VSAM, physical sequential, or other type of data set available by using the generic data interface.

DXTFILE description. Describes a physical sequential file, VSAM file, or any other data source accessed by a generic data interface (GDI) exit. (Describes the file organization and selected fields.)

DXTPCB. DataRefresher Program Communication Block.

DataRefresher Program Communication Block (DXTPCB). A DataRefresher object stored in the FDTLIB that describes an IMS/VS PCB to DataRefresher.

DataRefresher Program Specification Block (DXT PSB). A DataRefresher object stored in the FDTLIB that describes an IMS/VS PSB to DataRefresher.

DXTPSB. DataRefresher Program Specification Block.

DXTPSB description. For a given PCB, a description of the fields of interest to the user and the segments where they exist. Also included in the description of a segment are its length, its format (variable or fixed length), and the name of its parent. Included in the description for a field are its origin in its segment, its length, and its data characteristics.

DataRefresher user data type. Data in a format that DataRefresher does not directly support. A user-written conversion exit is required to convert data from the unsupported format into a format that DataRefresher supports.

DataRefresher user data type description. A description of the user-defined format to DataRefresher.

DXTVIEW. A DataRefresher object stored in the FDTLIB that defines which fields or segments the user may access. A DXTVIEW describes only those segments and fields in a single path of the hierarchy that the user can retrieve.

DXTVIEW description. Defines a DXTVIEW for a physical sequential or VSAM file, an IMS/VS DL/I database, or a data source accessed by a user-written generic data interface (GDI) exit.

EAR. Exit Address Routine.

EBCDIC. extended binary-coded decimal interchange code.

ECF. Enhanced Connectivity Facilities.

End User Dialogs. The DataRefresher program that lets a user build and submit extract requests through a series of panels.

end user table. A DataRefresher table generated to keep track of user IDs between systems.

Enhanced Connectivity Facilities (ECF). A set of programs designed to connect personal computers with host computers so that many host services and resources become available to personal computer users and application programmers.

entry-sequenced data set (ESDS). In VSAM, a file whose records are ordered by time of entry into the data set, and whose relative byte addresses cannot change. Records are retrieved and stored by sequential access, and new records are added at the end of the data set.

ESDS. entry-sequenced data set.

EXEC. A program consisting of a set of CP and CMS commands.

EXEC statement. An instruction within JCL/JCS that identifies the program to be run.

exit address routine. A user-written routine that, given the name of a GDI exit routine by DataRefresher, returns the current address of that GDI exit routine to DataRefresher.

exit routine. A user-written DataRefresher program. DataRefresher passes control to the exit routine for specialized processing.

EXTLIB. extract request library.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

extract request. A request to extract data from a source accessible by Data Refresher. Requires use of the SUBMIT and EXTRACT commands.

extract request library (EXTLIB). A VSAM key-sequenced data set (KSDS) that holds extract requests.

Fast Path. The IMS/VS function that supports applications requiring data availability and fast processing of simple data structures. Although Fast Path has its own databases and message processing, it is an integral part of IMS/VS.

FDTLIB. file description table library.

FDTLIB format. The format of a segment after a data exit transforms it, but prior to manipulation by any user data type exit during the extraction process. (Contrast with *database format*).

FDTLIB Migration Utility. A program used to migrate data descriptions from all prior releases of FDTLIB to the current release of FDTLIB.

field definition time. The time at which a user data type is specified in a FIELD statement of a CREATE DXTFILE or CREATE DXTPSB command.

file description table library (FDTLIB). A VSAM key-sequenced data set (KSDS) that holds the descriptions of all databases, files, and DataRefresher views available to the DEM and UIM.

file space. Used by shared file system (SFS), a logical space where a user's files are kept.

file pool. A set of minidisks managed by SFS.

fixed-length record. A record having the same length as all other records with which it is logically or physically associated. (Contrast with *variable-length record*)

full-screen editing. Editing at a display terminal which displays an entire screen of data at once and in which the user can access data through commands or by positioning a cursor.

GDI. generic data interface.

GDI Record exit routine. A user-written program that can access a non-IBM DBMS or data source that does not have an SQL interface, extract from a self-defining file, such as an IXF file, or join data from diverse data sources. DataRefresher does not pass SELECT statements to a GDI Record exit. However, DataRefresher can pass key values to a GDI Record exit. (Contrast with *GDI Select exit routine*).

GDI Select exit routine. A program that a user defines to access a non-IBM DBMS that has an SQL interface, or for help performing a two-stage extraction from a DB2 database. GDI select exits let a user submit an SQL-like SELECT statement to DataRefresher without previously storing field descriptions in the FDTLIB. (Contrast with *GDI record exit*).

General Data Extract feature. This DataRefresher feature (which includes the UIM and the DEM) lets you extract data from an IMS/VS database, or a physical sequential or VSAM file. If you use a generic data interface exit, you can extract data from IBM relational databases without using the REM or from other data sources not directly supported by DataRefresher.

generic data interface (GDI). An interface that accesses MVS databases and files not directly supported by DataRefresher. This source data is accessed via a GDI exit.

generic data interface (GDI) exit routine. A user-written routine that accesses MVS databases, VMS databases, and any files not directly supported by DataRefresher. There are two types of GDI exits: GDI Select exits and GDI Record exits.

generic output interface (GOI) exit routine. A user-written routine that receives extracted data and can be used to convert the data to a user-defined format or written to files not directly supported by DataRefresher.

GOI. generic output interface.

help panel. Information displayed when the user presses the HELP function key while using DataRefresher Dialogs.

hierarchical database. A tree-like, top-down arrangement of segments in a database, beginning at the top of the hierarchy with a root segment and proceeding downward to dependent segments, as in an IMS/VS DL/I database.

high-level language (HLL). A programming language that does not reflect the structure of any particular computer operating system.

High Speed Sequential Retrieval (HSSR). An IMS database tool that delivers efficient access performance to IMS data.

HLL. high-level language.

host. The primary or controlling computer in a multiple computer installation; in this case, the computer running DataRefresher. (Contrast with *remote*).

HSSR. High Speed Sequential Retrieval.

HUP. Hierarchical Update Program - a DataPropagator NonRelational feature used for propagating from a relational source, such as DB2, to a hierarchical target, such as IMS.

IBM Database 2* (DB2*). A program that provides a full-function relational database management system on MVS and supports access from MVS applications under IMS, CICS*, TSO, or batch environments.

IBM software distribution (ISD). The IBM division that distributes IBM programs.

IMS/VS. Information Management System/Virtual Storage.

Information Management System/Virtual Storage (IMS/VS). A database/data communication system capable of managing complex databases and networks.

INGRES* *. A relational database management system that is a product of Relational Technology, Inc.

installation verification procedure (IVP). An IBM program shipped with a product which verifies whether major segments of the product operate correctly following installation of the product.

Integration Exchange Format (IXF). IXF is a self-defining sequential file format providing character and source representation of data that helps applications exchange data.

Interactive System Productivity Facility (ISPF). A program that controls the execution of DataRefresher Dialogs.

internal segment. A repeating group of data within a parent segment; the data can be fixed or variable in length.

ISD. IBM software distribution.

ISPF. Interactive System Productivity Facility.

IVP. installation verification procedure.

IXF. Integration Exchange Format.

JCL. job control language.

JCS. job control statement(s).

JES2. Job Entry Subsystem 2.

JES3. Job Entry Subsystem 3.

job. A set of computer programs, files, and control statements that are sent to the operating system for processing.

job control language (JCL). A control language used to identify a job to an operating system and to describe the job's requirements. In DataRefresher, JCL is the control language used to describe required DataRefresher data resources and to run a Data Refresher job. A DataRefresher job stream includes JCL and Data Refresher commands.

job control statement(s) (JCS). JCS is a set of control statements that identify and describe a job to the operating system for routing and final processing of extracted data. JCS can include both JCL and extracted data. The JCS controls the DRU, any load utility, or any application that processes the extracted data.

Job Entry Subsystem 2 (JES2). An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with multiple processors, each JES2 processor runs independently. See also *Job Entry Subsystem 3*.

Job Entry Subsystem 3 (JES3). An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with several loosely coupled processors, JES3 lets the global processor exercise centralized control. See also *Job Entry Subsystem 2*.

job statement. The job control statement that identifies the beginning of a job. It contains such information as the name of the job, account number, and class and priority assigned to the job.

job step. A unit of work represented by running a single program that resides in the load library. A job can consist of one or more steps.

Julian date. A date format that contains the year in positions 1 and 2, and the day in positions 3 through 5. The day is represented as 1 through 366, right-adjusted, with zeros in any unused high-order positions.

Kanji feature. The DataRefresher program that establishes a Kanji-language environment for DataRefresher Dialogs users.

katakana. A character set of symbols used in one of the Japanese phonetic alphabets. The DataRefresher uppercase feature allows the dialogs to be viewed on terminals which support katakana.

key. (1) One or more characters used to identify the record and establish the order of the record within an indexed file. (2) In VSAM, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records.

key-sequenced data set (KSDS). A VSAM file whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed

access or by sequential access, and new records are inserted in key sequence by means of distributed free space.

keyword. (1) A part of a DataRefresher command parameter that has a specific meaning to that command and is shown in uppercase letters in the syntax diagram. See also *parameter*.

KSDS. key-sequenced data set.

LE/370 (Language Environment/370). SAA* AD/Cycle* Language Environment*/370.

Language Environment/370. SAA AD/Cycle Lanaguage Environment/370. A program that allows one high level language to have imbedded calls to routines written in some other high level language.

load utility. A program that puts data into one or more tables in a table space or partition.

Logical Unit (LU). An interface through which a DataRefresher user accesses the SNA network.

logical unit name. The Virtual Telecommunications Access Method (VTAM*) logical unit resource name for initiating communications with the remote node.

LU. Logical Unit.

main storage databases (MSDB). An IMS/VS Fast Path database used to store and provide access to an installation's most frequently used data. The data in an MSDB is stored in segments. Each segment can be available to all terminals, or assigned to a specific terminal. To provide fast access and allow frequent update to this data, MSDBs reside in virtual storage during execution. MSDBs cannot be shared.

Map Capture Communication Area (MCCA). The control block used for communication between DataRefresher (either the UIM or the DEM) and a user-written map capture exit routine.

Map Capture exit routine. A user-written exit routine that can retrieve DataRefresher mapping information for all files and PSBs used during an extract request. This mapping information can be saved for later use, for example, for data propagation.

master index table (MIT). A table that contains administrative information for End User Dialogs. For example, the nickname table and the end user table.

MCCA. Map Capture Communication Area.

minimum segment (MINSEGM). The lowest segment necessary in a hierarchical path to qualify for extraction.

MINSEGM. minimum segment.

MIT. master index table.

MSDB. main storage databases.

MVG. Map Verification and Generation - a DataPropagator NonRelational feature used with the DataRefresher UIM when using a DataPropagator NonRelational map capture exit in the DataRefresher SUBMIT command.

Multiple Virtual Storage (MVS). An IBM operating system that is in an SAA environment.

MVS. Multiple Virtual Storage.

network job entry (NJE). Used in JES2. Allows selected jobs, in-stream (SYSIN) data sets, system output (SYSOUT) data sets, operator commands and messages, and job accounting information to be transmitted from one computer system to another.

network job interface (NJI). Used in JES3. Allows selected jobs, in-stream (SYSIN) data sets, system output (SYSOUT) data sets, operator commands and messages, and job accounting information to be transmitted from one computer system to another.

nickname. In DataRefresher End User Dialogs, a short, convenient name assigned to a specific node or subsystem and the JCL used to route requests to that system.

NJE. network job entry.

NJI. network job interface.

node entry. A name that defines the source and target systems; for example, the node entry in a JCL job control statement or CMSBATCH link.

null separator field. The 1-byte field (2 bytes for IXF output) that begins each data field in an extract output row. DataRefresher puts a hyphen in this field if the data field is null. It is called a separator because it visually separates the data columns in the extract output.

Online DataRefresher commands. A set of TSO REXX EXECs that let a user run DataRefresher commands in the TSO foreground; process UIM, DEM, and REM requests.

ORACLE**. A database management system that is a product of Oracle Corporation.

packed decimal data type. A data type in which each byte in the field except the right-most byte represents two numeric digits. The rightmost digit contains one digit and the sign. For example, the decimal value +123 is represented as 0001 0010 0011 1111.

panel. A predefined display image. It may be a menu, a data entry panel, or for information only.

parameter. A keyword, or variable, or a combination of keywords and variables used with a command to affect its result. In DataRefresher command syntax, required parameters are displayed on the main path of the syntax and optional parameters are displayed below the main path. Default parameters are displayed above the main path of the syntax. See also *keyword*.

parent segment. A segment in a database that has one or more dependent segments below it in a hierarchy.

partial path. A hierarchical path without an occurrence for every segment.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PCB. Program Communication Block.

PDS. Partitioned Data Set.

persistent extract. An extract request that is retained in the EXTLIB after an extract runs.

physical segment. The smallest unit of accessible data in a database.

physical sequential file. A file in which records are processed in the order in which they occur in the file.

polling interval. The elapsed time between DEM searches of the EXTLIB. The DEM periodically searches for qualifying extract requests to process. This is used with long-running DEMs.

Program Communication Block (PCB). An object that describes a communication block to a program.

Program Specification Block (PSB). A set of statements naming the required databases, segments to access, and database modification options for a program. The PSB contains a given program communication block (PCB) for each database named, in the sequence used by the program.

Program Support Representative (PSR). An IBM appointed program support technician.

program temporary fix (PTF). A temporary solution or bypass of a problem diagnosed by IBM.

proxy. An account identifier on a VMS operating system (a product of the Digital Equipment Corporation). The proxy provides IBM users with file access and default privileges on the VAX** computer system.

PSB. Program Specification Block.

PSR. Program Support Representative.

PTF. program temporary fix.

QMF. Query Management Facility.

QSAM. queued sequential access method.

Query Management Facility (QMF*). An interactive query product that lets you create reports and charts from relational data.

queued sequential access method (QSAM). A queue containing input data blocks that are awaiting processing or output data blocks that have been processed and are awaiting transfer to either auxiliary storage or to an output device.

RACF. Resource Access Control Facility.

relational database. A database that is organized and accessed according to relationships between data items. Contrast with *hierarchical database*.

relational database view. A relational database view is created by DB2 or SQL/DS and controls access to data on these relational databases.

Relational Data Extract Feature. A DataRefresher feature. The REM is used for extracting data from a DB2 or SQL/DS database.

Relational Extract Manager (REM). The DataRefresher program that extracts data from a DB2 or SQL/DS database.

REM. Relational Extract Manager.

REM data sources. Any DB2 or SQL/DS data accessed by the Relational Extract Manager (REM).

remote. Pertaining to a system, program, or device that is accessed through a telecommunication line. (Contrast with *host*)

remote file description table library (RFDTLIB). A library on a non-IBM computer containing all the DXTFILE and DXTVIEW definitions of data accessible to the non-IBM operating system that are candidates for extracts using DataRefresher.

This feature was used by DXT/D1 but is no longer supported.

Remote Spooling Communications Subsystem (RSCS). The licensed program that allows the VM system to fully participate in a network of SNA/non-SNA

Network Job Entry (NJE) System nodes, SNA/non-SNA 3270* Information Display System printer nodes, and Bisync Remote Job Entry (RJE) nodes. This capability permits CMS users to transmit and receive spool files or messages to or from any defined node in the network.

repository. An organized group of information that supports business and data processing activities and provides a single point of control for the management and sharing of that information.

Resource Access Control Facility (RACF). An IBM program that provides for controlled access to system resources by identifying and verifying users to the system, authorizing access to DASD data sets, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected data sets.

RFDTLIB. remote file description table library.

root segment. In IMS/VS, the main segment of a database to which all other segments are related. This is the top of the hierarchy tree.

RSCS. Remote Spooling Communications Subsystem.

run mode. Identifies a DEM as either long-running or terminating. The run mode of a long-running DEM determines how long the DEM should run. When in long-running mode the DEM uses a polling interval.

RUP. Relational Utility Program - a DataPropagator NonRelational feature used with the DataRefresher UIM when using a DataPropagator NonRelational map capture exit in the DataRefresher SUBMIT command.

SAA. Systems Application Architecture*.

SAP. Structures Access Program.

session. A long logical connection that allows communication between two logical units using the APPC. (Contrast with *conversation*)

SFS. shared file system.

SFS Directory. The place where shared file system files are grouped — analogous to a minidisk.

shared file system (SFS). An extension of the CMS file system, that allows simultaneous sharing of CMS programs and data by multiple users and applications.

simple file. A file that contains only one record type. Contrast with *structured file*.

SMP. System Modification Program.

SQL. Structured Query Language.

SQL/DS. Structured Query Language/Data System.

structured file. A file that contains multiple record types or internal segments or both. (Contrast with *simple file*).

Structures Access Program (SAP). A DataRefresher program that employs user-specified data structures to generate DataRefresher data description statements, extract request statements, and a statement specifying the creation of a DB2 table to contain the extracted data.

Structured Query Language (SQL*). A language used to communicate with DB2 and SQL/DS.

Structured Query Language/Data System (SQL/DS*). The relational database management system that runs under VM.

synchronous. Occurring with a regular or predictable time relationship.

System Modification Program (SMP). The program used to install DataRefresher under MVS.

Systems Application Architecture* (SAA*). A set of IBM software interfaces, conventions, and protocols that provide a framework for designing and developing applications that are consistent across systems.

table. A named collection of data consisting of a number of named vertical rows and a number of unordered horizontal rows which is under the control of a relational database manager.

time sharing option (TSO). An option on the operating system that provides interactive time sharing from a display station.

transaction. (1) A job or a job step. (2) In IMS, a specific set of input data that starts a specific processor job.

translation table. A table used by DataRefresher that provides replacement characters of one code page for characters of a different code page.

TSO. time sharing option.

TSO/E REXX. The implementation of the Systems Application Architecture (SAA) Procedures Language on the MVS system.

TSO foreground. The environment in which programs are swapped in and out of main storage to let terminal users share processing time.

2-stage extraction. The DataRefresher process for extracting data from a non-relational data source. First, the UIM validates and queues the request in EXTLIB; the DEM then executes the request, depending on the schedule that you have established for the DEM. By

contrast, extract requests from relational data sources are immediately executed by the REM (one-stage).

UCF. Uppercase Feature.

UIC. user identification code.

UIM. User Input Manager.

Uppercase Feature (UCF). The DataRefresher program that lets users use the DataRefresher Dialogs on Japanese Katakana terminals (555x, 556x).

User Data Type exit routine. A user-written routine that transforms fields in a user-defined format into a data type supported by DataRefresher.

user identification code (UIC). A unique identifier for each user on the VMS system.

User Input Manager (UIM). The DataRefresher program that validates and enters file and database descriptions into the FDTLIB and validates and enters DEM extract requests into the EXTLIB.

value. Information assigned to a parameter associated with a command or keyword.

variable. A part of a DataRefresher command parameter that you supply and is displayed in lowercase letters in the syntax diagram.

variable-length record. A record having a length independent of the length of other records with which it is logically or physically associated. (Contrast with *fixed-length record*).

view. See *DXTVIEW, DXTVIEW description*, or *relational database view*.

Virtual Machine (VM). An IBM system that is part of an SAA environment.

Virtual Memory System (VMS)**. An operating system produced by Digital Equipment Corporation that is used on the VAX computer.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed- and variable-length records on direct access devices. A key field (key sequence) can organize the records in a data set or file logically, in the physical sequence in which they are written (entry sequence), or by relative-record number. DataRefresher does not support extraction from a VSAM relative record data set (RRDS).

VM. Virtual Machine.

VMS. Virtual Memory System.

VSAM. Virtual Storage Access Method.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Communicating Your Comments to IBM

DataRefresher
Version 1
Command Reference

Publication No. SH19-6999-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  +353 - 1 - 6614246

- If you prefer to send comments electronically, use this network ID:
  - IBM Mail Exchange: IEIBM3FL at IBMMAIL
  - Internet: IEIBM3FL@IBMMAIL.COM

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

DataRefresher
Version 1
Command Reference
Publication No. SH19-6999-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____ _____

Phone No. _____ _____

Readers' Comments — We'd Like to Hear from You
SH19-6999-00

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Software Solutions,
Information Development (IISL),
2 Burlington Road,
Dublin 4,
Ireland.

Fold and Tape **Please do not staple** Fold and Tape

Readers' Comments — We'd Like to Hear from You

DataRefresher
Version 1
Command Reference
Publication No. SH19-6999-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____ _____

Phone No. _____ _____

Readers' Comments — We'd Like to Hear from You
SH19-6999-00

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Software Solutions,
Information Development (IISL),
2 Burlington Road,
Dublin 4,
Ireland.

Fold and Tape **Please do not staple** Fold and Tape

Program Number: 5696-703

DataRefresher Version 1 Library

SH19-5000	Messages and Codes
LY19-6386	Diagnosis Guide
GH19-6993	An Introduction
SH19-6995	Administration Guide
SH19-6996	MVS and VM User's Guide
SH19-6997	OS/2 User's Guide
SH19-6998	Exit Routines
SH19-6999	Command Reference
GH19-9994	Licensed Program Specifications

SH19-6999-00