UNLOAD PLUS for DB2
Reference Manual

Supporting

Version 11.2 of UNLOAD PLUS for DB2
Version 11.2 of BMC High Speed Utilities for DB2
Version 11.2 of BMC Object Administration for DB2
Version 11.2 of BMC Database Administration for DB2

May 2015
Contacting BMC Software

Several methods are available for contacting BMC Software.

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC</td>
<td>1 713 918 8800</td>
<td>1 713 918 8000</td>
</tr>
<tr>
<td>2101 CITYWEST BLVD HOUSTON TX 77042-2827 USA</td>
<td>or</td>
<td>1 800 841 2031</td>
</tr>
</tbody>
</table>

Outside United States and Canada

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+01 713 918 8800</td>
<td>+01 713 918 8000</td>
</tr>
</tbody>
</table>


BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

BatchPipes, DB2, DB2 Universal Database, FlashCopy, IBM, IBM z, Language Environment, MVS, RACF, System z10, zArchitecture, z/OS, and z10 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

Restricted rights legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2101 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

Support website
You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can:

- Read overviews about support services and programs that BMC offers
- Find the most current information about BMC products
- Search a database for problems similar to yours and possible solutions
- Order or download product documentation
- Download products and maintenance
- Report a problem or ask a question
- Subscribe to receive proactive e-mail alerts
- Find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail
In the United States and Canada, if you need technical support and do not have access to the web, call 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:yourSupportContractID, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC
Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - Product name
  - Product version (release number)
  - License number and password (trial or permanent)

- Operating system and environment information
  - Machine type
  - Operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - System hardware configuration
  - Serial numbers
  - Related software (database, application, and communication) including type, version, and service pack or maintenance level

- Sequence of events leading to the problem
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - Product error messages
  - Messages from the operating system
  - Messages from related software
License key and password information

If you have questions about your license key or password, contact Customer Support through one of the following methods:

- Send an e-mail message to customer_support@bmc.com. (In the Subject line, enter `SupID:yourSupportContractID`, such as SupID:12345.)
- In the United States and Canada, call 1 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
# Contents

## About this book
- Related publications ................................................................. 11
- Conventions ............................................................................. 12
- Syntax diagrams ........................................................................ 12
- UNLOAD PLUS summary of changes ........................................ 14

## Chapter 1  Introduction to UNLOAD PLUS
- Overview of UNLOAD PLUS ...................................................... 21
- BMC Software solution integration .......................................... 23
- Tasks that UNLOAD PLUS performs ....................................... 24
- Understanding UNLOAD PLUS execution modes ..................... 28
- How UNLOAD PLUS works .................................................... 30
  - DIRECT YES processing phases ........................................ 31
  - DIRECT NO processing phases ....................................... 33
  - UNLOAD PLUS data sets ................................................ 35
  - Associated products and common components that UNLOAD PLUS uses ............................................. 38

## Chapter 2  Operational considerations in UNLOAD PLUS
- System setup for UNLOAD PLUS .............................................. 41
  - DB2 support in UNLOAD PLUS ........................................... 41
  - UNLOAD PLUS for DB2 system and software requirements .......................................................... 41
  - Required authorization for UNLOAD PLUS ......................... 42
  - Overriding the MEMLIMIT system parameter .................... 45
- Number of DB2 threads that UNLOAD PLUS uses ................. 45
- UNLOAD PLUS serialization and concurrency ......................... 46
  - Executing BMC utilities concurrently ................................ 46
  - Object status with DIRECT YES ........................................ 48
  - Object status with DIRECT NO ........................................ 51
  - Drain and lock table processing in UNLOAD PLUS ............. 51
- Dynamic data set allocation in UNLOAD PLUS ......................... 52
- Running SHRLEVEL CHANGE CONSISTENT YES ................. 53
- Unloading LOB and XML data .............................................. 54
  - Unloading LOB and XML data to unload data sets .............. 54
  - Unloading LOB and XML data to referenced files ............... 55
  - General considerations when unloading LOB and XML data .... 57
### Considerations when unloading LOB or XML data to unload data sets

- Considerations when unloading LOB or XML data to referenced files

### Considerations for unloading DB2 catalog data

- Considerations for unloading to a BatchPipes file

### UNLOAD PLUS output format

- DB2 output with external date and time (FORMAT STANDARD)
- External-formatted output (FORMAT EXTERNAL)
- DB2 output with internal date and time (FORMAT INTERNAL)
- Data migration using LOADPLUS (FORMAT BMLOAD)
- DSNTIAUL-formatted output (FORMAT DSNTIAUL)
- CSV output (FORMAT CSV)
- XML output (FORMAT XML)
- Examples of FORMAT option output

### DB2 user exits with UNLOAD PLUS

### DB2 features that UNLOAD PLUS does not support

### Chapter 3 Syntax of the UNLOAD command

- Command syntax rules for UNLOAD PLUS
- Specifying object names in your UNLOAD PLUS syntax
- Alphabetical listing of UNLOAD PLUS options
- UNLOAD PLUS command syntax diagrams
- UNLOAD PLUS options
  - Basic processing options
  - OUTPUT command options
  - SELECT options
  - Data type keywords in UNLOAD PLUS
- Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS
- Supported data type conversions in UNLOAD PLUS
- Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS
- Additional numeric field considerations in UNLOAD PLUS
- Additional date, time, or timestamp field considerations in UNLOAD PLUS
- Additional variable field considerations in UNLOAD PLUS
- Additional ROWID field considerations in UNLOAD PLUS
- Data translation in UNLOAD PLUS

### Chapter 4 Building and executing UNLOAD PLUS jobs

- Building the UNLOAD PLUS job
- JOB statement
UNLOAD PLUS EXEC statement ................................................................. 272
REGION parameter ................................................................................. 272
Utility parameters on the UNLOAD PLUS EXEC statement .............. 272
UNLOAD PLUS STEPLIB DD statement ................................................... 276
UNLOAD PLUS DD statements ................................................................. 276
Running UNLOAD PLUS jobs ................................................................. 308
Invoking UNLOAD PLUS ........................................................................ 308
Restarting UNLOAD PLUS ................................................................. 308
Terminating or canceling an UNLOAD PLUS job ................................. 309

Chapter 5 Examples of UNLOAD PLUS jobs 311
Overview of UNLOAD PLUS examples ................................................. 311
Example 1 - Unloading partitions to separate data sets ....................... 314
Example 2 - Unloading columns in clustering order ......................... 321
Example 3 - Unloading multiple tables to a single data set ............... 325
Example 4 - Unloading while allowing read write access to the table space .. 329
Example 5 - Unloading data from a DSN1COPY ................................. 332
Example 6 - Unloading a full image copy ............................................. 335
Example 7 - Using FILTERPART ............................................................... 338
Example 8 - Using multiple SELECT statements .................................. 342
Example 9 - Unloading from multiple table spaces ......................... 347
Example 10 - Unloading from incremental image copies .................. 350
Example 11 - Using a DDLIN data set ..................................................... 353
Example 12 - Unloading to CSV format ............................................... 357
Example 13 - Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism .. 360
Example 14 - Handling an abnormal termination ............................. 363
Example 15 - Unloading LOB and XML data; using nonstandard null indicators ............................................................................. 366
Example 16 - Using dynamic allocation .............................................. 373
Example 17 - Changing overpunch values in zoned decimal data .......... 377
Example 18 - Unloading a table space and unloading data as Unicode ............. 381
Example 19 - Unloading data to LOADPLUS ..................................... 389

Chapter 6 Tuning UNLOAD PLUS jobs 399
Tuning for performance ........................................................................ 399
Setting installation options for optimal performance in UNLOAD PLUS ......................................................... 399
Specifying command options for optimal performance in UNLOAD PLUS ................................................................. 401
Additional performance information for UNLOAD PLUS installation and command options ......................................................... 403
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabling multitasking for performance in UNLOAD PLUS</td>
<td>407</td>
</tr>
<tr>
<td>Additional performance tuning recommendations for UNLOAD PLUS</td>
<td>410</td>
</tr>
<tr>
<td>Performance tuning for specific scenarios in UNLOAD PLUS</td>
<td>411</td>
</tr>
<tr>
<td>Interpreting performance-related messages from UNLOAD PLUS</td>
<td>412</td>
</tr>
<tr>
<td>Tuning to improve memory use in UNLOAD PLUS</td>
<td>414</td>
</tr>
<tr>
<td>Tuning sort processing</td>
<td>415</td>
</tr>
<tr>
<td>Tuning when unloading objects with a large number of partitions</td>
<td>415</td>
</tr>
<tr>
<td>Appendix A</td>
<td>UNLOAD PLUS installation options</td>
</tr>
<tr>
<td>Overview of UNLOAD PLUS installation options</td>
<td>417</td>
</tr>
<tr>
<td>Basic UNLOAD PLUS installation options</td>
<td>418</td>
</tr>
<tr>
<td>Descriptions of basic UNLOAD PLUS installation options</td>
<td>422</td>
</tr>
<tr>
<td>Dynamic allocation installation options</td>
<td>447</td>
</tr>
<tr>
<td>Default output descriptor options for dynamic allocation</td>
<td>447</td>
</tr>
<tr>
<td>Options that are common to disk and tape data sets</td>
<td>449</td>
</tr>
<tr>
<td>Options for disk data sets</td>
<td>457</td>
</tr>
<tr>
<td>Options for tape data sets</td>
<td>461</td>
</tr>
<tr>
<td>DYNALOC installation option</td>
<td>463</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Common utility tables</td>
</tr>
<tr>
<td>Overview of common utility tables</td>
<td>467</td>
</tr>
<tr>
<td>Warnings and considerations for common utility tables</td>
<td>469</td>
</tr>
<tr>
<td>Managing common utility tables</td>
<td>471</td>
</tr>
<tr>
<td>BMCDICT table</td>
<td>472</td>
</tr>
<tr>
<td>BMCDICT table considerations</td>
<td>473</td>
</tr>
<tr>
<td>Maintaining the BMCDICT table</td>
<td>473</td>
</tr>
<tr>
<td>BMCHIST table</td>
<td>474</td>
</tr>
<tr>
<td>BMCHIST table considerations for COPY PLUS</td>
<td>476</td>
</tr>
<tr>
<td>BMCHIST table considerations for RECOVER PLUS</td>
<td>476</td>
</tr>
<tr>
<td>Maintaining the BMCHIST table</td>
<td>477</td>
</tr>
<tr>
<td>BMCLGRNEX table</td>
<td>477</td>
</tr>
<tr>
<td>BMCSYNC table</td>
<td>478</td>
</tr>
<tr>
<td>BMCSYNC table considerations</td>
<td>481</td>
</tr>
<tr>
<td>Maintaining the BMCSYNC table</td>
<td>483</td>
</tr>
<tr>
<td>Cleaning up RECOVER UNLOADKEYS entries</td>
<td>483</td>
</tr>
<tr>
<td>Shared access levels of BMC utilities</td>
<td>484</td>
</tr>
<tr>
<td>BMCTRANS table</td>
<td>486</td>
</tr>
<tr>
<td>BMCUTIL table</td>
<td>488</td>
</tr>
<tr>
<td>Maintaining the BMCUTIL table</td>
<td>491</td>
</tr>
<tr>
<td>BMXCOPY table</td>
<td>491</td>
</tr>
<tr>
<td>Maintaining the BMXCOPY table</td>
<td>498</td>
</tr>
</tbody>
</table>
### Appendix C  Generating control statements for DB2 or other software products  499

Overview of generating control statements in UNLOAD PLUS  499

Inserting additional command options into utility control statements  500

Control statements to create and load DB2 tables  501
  - DB2LOAD option  501
  - DB2 option  503
  - BMCLOAD option for LOADPLUS for DB2  505
  - DB2DDL option  508

Control statements for FOCUS processing  509

Control statements for SAS processing  511

Control statements to create and load SQL/DS tables  514
  - SQL/DS-LOAD option  514
  - SQL/DS option  516
  - SQL/DS-DDL option  518

Control statements for Easytrieve processing  518

Control statements to load Teradata tables  520
  - TERADATA-BULK option  522
  - TERADATA-FAST option  525
  - TERADATA-MULT option  526
  - TERADATA option  528

Control statements for NOMAD processing  528

### Appendix D  UNLOAD PLUS user exits  531

Overview of UNLOAD PLUS user exits  531

Requirements and restrictions for UNLOAD PLUS user exits  531

Using sample UNLOAD PLUS user exits  532

Parameter data and pseudo-SQLDA  533

Function codes and return codes for UNLOAD PLUS user exits  533

UNLOAD PLUS pseudo-SQLDA  534

UNLOAD PLUS Assembler user exit  537
  - Exit parameter block DSECT  538
  - Pseudo-SQLDA DSECT  539
  - Sample Assembler user exit  541

UNLOAD PLUS C user exit  550
  - C exit parameter structure  550
  - C pseudo-SQLDA structure  553
  - Sample C user exit  555

UNLOAD PLUS LE C user exit  566
  - LE C exit parameter structure  566
  - Pseudo-SQLDA structure  569
Sample LE C user exit ................................................................. 571
UNLOAD PLUS COBOL II and LE COBOL user exits ................................. 582
Exit parameter record ........................................................................ 582
COBOL pseudo-SQLDA record .............................................................. 584
COBOL II and LE COBOL user exits: sample 1 ........................................ 586
COBOL II and LE COBOL user exits: sample 2 ........................................ 588
About this book

This book contains detailed information about the UNLOAD PLUS for DB2 product and is intended for IBM DB2 system administrators, DB2 database administrators, and DB2 application programmers.

To use this book, you should be familiar with the following items:

- IBM DB2 Universal Database for z/OS (DB2) DBMS
- IBM z/OS operating system
- Job control language (JCL)
- Interactive System Productivity Facility (ISPF)

Like most BMC documentation, this book is available in printed and online formats. To request printed books or to view online books and notices (such as release notes and technical bulletins), see the support website at http://www.bmc.com/support.

Note

Online books are formatted as PDF or HTML files. To view, print, or copy PDF books, use the free Adobe Reader from Adobe Systems. If your product installation does not install the reader, you can obtain the reader at http://www.adobe.com.

Related publications

From the BMC Support Central website, you can use the following methods to access related publications that support your product or solution:


- View Quick Course videos (short overviews of selected product concepts, tasks, or features), which are available from the following locations:
— Documentation Center

— Support Central (at http://www.bmc.com/support/mainframe-demonstrations)

— BMC Mainframe YouTube channel (https://www.youtube.com/user/BMCSoftwareMainframe)


You can order hardcopy documentation from your BMC sales representative or from the support site. You can also subscribe to proactive alerts to receive e-mail messages when notices are issued.

**Tip**
You can access the BMC Support Central site at http://www.bmc.com/support.

---

**Conventions**

This document uses the following special conventions:

■ All syntax, operating system terms, and literal examples are presented in this typeface.

■ Variable text in path names, system messages, or syntax is displayed in italic text: testsys/instance/fileName

■ Menu sequences use a symbol to convey the sequence. For example, Actions => Create Test instructs you to choose the Create Test command from the Actions menu.

**Syntax diagrams**

The following figure shows the standard format for syntax diagrams:
The following example illustrates the syntax for a hypothetical DELETE statement. Because the FROM keyword, alias variable, and WHERE clause are optional, they appear below the main command line. In contrast, the tableName variable appears on the command line because the table name is required. If the statement includes a WHERE clause, the clause must contain a search condition or a CURRENT OF clause. (The searchCondition variable appears on the main line for the WHERE clause, indicating that this choice is required.)

The following guidelines provide additional information about syntax diagrams:

- Read diagrams from left to right and from top to bottom.

- A recursive (left-pointing) arrow above a stack indicates that you may choose more than one item in the stack.

- An underlined item is a default option.

- If a diagram shows punctuation marks, parentheses, or similar symbols, you must enter them as part of the syntax.

- In general, IBM commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum required
portion might be shown in uppercase letters, with the remainder in lowercase (for example, CANcel).

- The following conventions apply to variables in syntax diagrams:
  - Variables are typically displayed in lowercase letters and are always italicized.
  - If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words (for example, databaseName).

## UNLOAD PLUS summary of changes

This topic summarizes product changes and enhancements by version number and release date.

**Version 11.2.00, May 2015**

This version fixes known problems in the product. In addition, it includes the following enhancements:

- UNLOAD PLUS now allows you to unload inline LOB data when specifying DIRECT YES or DIRECT NO. (Earlier versions required DIRECT NO.) For more information, see “Unloading LOB and XML data” on page 54.

  **Note**
  
  PTF BPU6778 provided this functionality for version 11.1.00. For more information, see the technical bulletin dated October 31, 2014.

- When DIRECT YES is in effect, you can now unload LOB and XML data, including inline LOB data, to a standard VBS data set. To enable this feature, you specify the SPANNED YES option. For more information, see “SPANNED” on page 129.

  **Note**
  
  This data set is a standard VBS data set whose format is similar to the IBM proprietary spanned record format. The FORMAT SPANNED YES option of the IBM DB2 LOAD utility requires that the input be in this proprietary spanned record format. Therefore, you cannot use the IBM DB2 LOAD utility to load the VBS data set that UNLOAD PLUS produces.

  **Note**
  
  PTF BPU6778 provided this functionality for version 11.1.00. For more information, see the technical bulletin dated October 31, 2014.
When DIRECT NO is in effect, UNLOAD PLUS now optionally offloads eligible queries to the IBM DB2 Analytics Accelerator for z/OS. You can enable this support with either the QUERY_ACCELERATION DSNZPARM, as documented in the IBM documentation, or by using the new UNLOAD PLUS option SET CURRENT QUERY ACCELERATION ALL. For more information, see “SET CURRENT QUERY ACCELERATION ALL” on page 182.

UNLOAD PLUS now supplies a default value of (1950,2049) for the CENTURY installation option. (Earlier versions were not shipped with a value for this option, requiring that you supply it during installation.)

If you currently use a default value other than (1950,2049), either change the value in your installation options module after installation, or use the Installation System to migrate your current option values.

UNLOAD PLUS now supports read-or-replication-only (RREPL) status.

Starting with this release, UNLOAD PLUS does not support DB2 Version 9. Earlier releases will continue to support Version 9.

In addition, UNLOAD PLUS supports DB2 Version 10 only in new-function mode.

This version of UNLOAD PLUS has the following changes to minimum requirements. For full requirement information, see “System setup for UNLOAD PLUS” on page 41.

— DB2 Solution Common Code (SCC) version 11.1.00 with PTF BPJ0944
— DB2 Utilities Common Code (D2U) version 11.2.00

Version 11.1.00, January 2014

This version fixes known problems in the product. In addition, it includes the following enhancements:

UNLOAD PLUS supports the following features of DB2 Version 11:

— Catalog and directory changes and restructuring
— Extended relative byte addresses (RBAs) and log record sequence numbers (LRSNs)

UNLOAD PLUS no longer requires that you specify an XBM subsystem for unload jobs that use snapshot technology. When the following conditions exist, UNLOAD PLUS searches for an XBM subsystem that is at the correct maintenance and enablement level:

— You are unloading using SHRLEVEL CHANGE CONSISTENT YES.
— A value is not in effect for the XBMID option.
This release enhanced the locking technique for sequential file access to reduce the possibility of user 3900 abends.

This version of UNLOAD PLUS has the following changes to minimum requirements.

— IBM System z10 processor

— DB2 Solution Common Code (SCC) version 11.1.00 with PTF BPJ0689

— DB2 Utilities Common Code (D2U) version 11.1.00

— XBM or SUF version 6.1.00

**Version 10.2.00, June 2013**

This version fixes known problems in the product. In addition, it includes the following enhancements:

*DIRECT YES support for features for which UNLOAD PLUS required DIRECT NO in earlier releases*

— UNLOAD PLUS now unloads clone tables and base tables that participate (or have participated) in a clone relationship. You can use this feature with either DIRECT YES or DIRECT NO.

The new CLONE option on the UNLOAD TABLESPACE command enables this feature.

— UNLOAD PLUS now supports universal table spaces that are defined with MEMBER CLUSTER. You can use this feature with either DIRECT YES or DIRECT NO.

— UNLOAD PLUS now supports temporal tables. You can unload temporal tables with either DIRECT YES or DIRECT NO.

The following enhancements are part of this support:

— You can use either DIRECT YES or DIRECT NO to unload history tables.

— When unloading temporal tables, UNLOAD PLUS generates control cards for the following CNTLCARDS values:

  — DB2LOAD

  — BMCLOAD

  — DB2
UNLOAD PLUS summary of changes

For specific information about which control cards UNLOAD PLUS generates for temporal tables, see the individual CNTLCARDS values in “CNTLCARDS” on page 130.

- UNLOAD PLUS now supports timestamp columns that are defined with a precision (number of microseconds) other than 6. You can use this feature with either DIRECT YES or DIRECT NO.

You can now specify a precision other than 6 when including a timestamp constant in a predicate.

- UNLOAD PLUS now supports timestamp columns that are defined as TIMESTAMP WITH TIME ZONE. You can use the following functions with either DIRECT YES or DIRECT NO:

  — Specify the TIMESTAMP WITH TIME ZONE data type on your field specification.

  — Use the new IMPLICIT_TZ command option to specify a time zone to use in any of the following situations:
    - You are unloading a TIMESTAMP column (without a time zone) and the field specification for the column is TIMESTAMP WITH TIME ZONE.
    - You include a SELECT statement on a TIMESTAMP WITH TIME ZONE field, and the SELECT statement specifies a timestamp constant that does not include a time zone.
    - You include a WHERE clause on a TIMESTAMP WITH TIME ZONE column, and the WHERE clause specifies a timestamp constant that does not include a time zone.

You cannot include WITH TIME ZONE on a CURRENT TIMESTAMP specification.

Enhancements and changes that require DIRECT YES

- When DIRECT YES is in effect, UNLOAD PLUS now unloads binary XML data to referenced files. You can specify the BINARYXML keyword with the CHAR and VARCHAR BLOBF data types.

- When DIRECT YES is in effect, you can now specify the LOGICAL keyword to tell UNLOAD PLUS which logical partitions you want to unload instead of having to specify the corresponding physical partitions.

Additional enhancements and changes
• UNLOAD PLUS now unloads from cabinet copies that have been created by the BMC Recovery Management for DB2 solution. To unload from a cabinet copy, specify INFILE IMAGECOPY.

• You can now tell UNLOAD PLUS to cancel threads in DB2 that might prevent a successful drain. A new command option, FORCE, enables this functionality; corresponding installation options FORCE, FORCE_AT, and FORCE_RPT provide default values for this option.

This option requires additional DB2 authorization.

You can also specify a new DD statement, BMCFORCE DD, to contain the thread cancelation report.

• You can now send UNLOAD PLUS output messages to a second output data set by specifying a SYSPRIN2 DD statement in your JCL. In a worklist environment, you can use SYSPRIN2 to view UNLOAD PLUS output in real time.

   Note

   PTF BPU3886 provided this functionality for version 10.1.00.

• The DSNTYPE installation and command options have been expanded to enable you to specify the type of output data set that you want UNLOAD PLUS to create during dynamic allocation.

• The ON FAILURE option has changed: UNLOAD PLUS now always terminates the unload job on failure. If your job includes the now-obsolete syntax STOP UTILITY, UNLOAD PLUS changes it to TERMINATE UTILITY.

• You can now specify a ddname prefix instead of the default VSAMDD when unloading from VSAM data sets. The new VSAMDDPREFIX keyword of the INFILE option enables this functionality, allowing you to unload from more than 99 data sets.

• This release removes the ACFORTSS option. Now, if the following conditions exist, you must ensure that the DB2 external security exit (DSNX@XAC) from Computer Technologies is implemented:

   — You use the Computer Technologies CA-ACF2 or CA-Top Secret security product for DB2.

   — You previously used ACFORTSS=Y.

• This version of UNLOAD PLUS has the following changes to minimum requirements.

   — BMCSORT version 2.4.01

   — DB2 Solution Common Code (SCC) version 11.1.00
— DB2 Utilities Common Code (D2U) version 10.2.00

■ Starting with this release, UNLOAD PLUS does not support DB2 Version 8. Earlier releases will continue to support Version 8. Future releases of UNLOAD PLUS will not support the following modes:

— DB2 Version 9 CM

— DB2 Version 10 CM8

■ This release includes the following documentation changes:

— All messages are now available in the BMC Documentation Center, which is accessible from the BMC Support Central site (http://www.bmc.com/support). A separate messages manual is no longer available.

— Installation and configuration information is located in the following books:

— Installation System User Guide

— BMC Products and Solutions for DB2 Configuration Guide
Introduction to UNLOAD PLUS

This chapter describes the features and advantages of UNLOAD PLUS and provides an architectural overview of the product.

Overview of UNLOAD PLUS

The BMC Software utility products for DB2 address the needs of database administrators (DBAs) and system administrators by providing high-performance database administration and utility products. The UNLOAD PLUS for DB2 product is a BMC Software product for unloading DB2 data.

DB2 is a powerful relational database management system whose wide success in the DBMS community rests on its ability to facilitate fast application development and provide easy access to data. As users depend more and more on the DB2 DBMS for critical business applications, the need for continuous operations becomes crucial.

The need to unload data

Data in a DB2 table often must be moved or copied. For example, you might want to

- Move data to a different DB2 subsystem, from a DB2 table to a sequential file for external processing, or to another relational database system or platform
- Extract a subset of rows from a table to use as test data, or to unload a DB2 table for changes to DB2 object definitions, such as those that the BMC ALTER for DB2 and CHANGE MANAGER for DB2 products perform
- Convert unloaded data to a different format during unload processing

The UNLOAD PLUS solution

BMC developed a product line to respond to the needs of DBAs, system administrators, and other DB2 users who require high-performance database administration and utility products. The UNLOAD PLUS product is an exceptionally
flexible, high-performance tool that combines all of the basic unload tasks with additional unload utility functions.

**UNLOAD PLUS benefits**

UNLOAD PLUS provides the following significant benefits:

- **Reduced costs** of unloading DB2 data
  UNLOAD PLUS dynamically eliminates processing of partitions that do not meet the selection criteria. It also handles all of its own buffering and performs I/O operations at the lowest level possible.

- **High availability of DB2 data**
  UNLOAD PLUS ensures application availability with online access during execution. Working with either EXTENDED BUFFER MANAGER (XBM) or its SNAPSHOT UPGRADE FEATURE (SUF), UNLOAD PLUS unloads a consistent image of your data while enabling read and write access to a table space.

  The following features also provide availability benefits:

  — **Exclusive technology** that eliminates SQL -904 outages
  — **Unique, high-performance option** that works with LOADPLUS to reduce total processing time for unloading and reloading data

- **Flexible processing options**
  In addition to unloading directly from DB2 table spaces, you can unload data from other sources, such as VSAM linear data sets and several types of copy data sets.

  UNLOAD PLUS also provides flexibility through the following options:

  — **Powerful, easy-to-use SELECT-like syntax** for specifying the data that UNLOAD PLUS unloads
  — **Powerful syntax that can extract all data from a table space with one command**
    You can unload an entire table space without writing and maintaining multiple SELECT statements.

- **Minimal interference** with DB2 performance and resources because UNLOAD PLUS runs outside of DB2

- **Expanded functionality** to eliminate the need to perform additional tasks before and after data is unloaded

  For example, UNLOAD PLUS provides the following functions:
— Comprehensive set of data type conversions to reduce the need for special application code

— Ability to unload data from image copies even if an object has been changed or no longer exists

— Ability to dynamically allocate primary and secondary unload data sets, reducing the need to include DD statements in your JCL

■ Integration with BMC solutions for DB2

BMC Software solution integration

UNLOAD PLUS is also a component of the following solutions. For more information, see the release notes for these solutions.

BMC Object Administration for DB2 solution

The Object Administration for DB2 solution allows you to manage your DB2 databases. Using the solution simplifies the administration of your DB2 database, improves availability, and ensures data integrity.

This solution includes the following products and primary technology components:

■ ALTER for DB2
■ CATALOG MANAGER for DB2
■ CHANGE MANAGER for DB2
■ BMC Common Statistics
■ COPY PLUS for DB2
■ LOADPLUS for DB2
■ SNAPSHOT UPGRADE FEATURE for DB2 (SUF)
■ UNLOAD PLUS for DB2
■ Workbench for DB2

BMC High Speed Utilities for DB2 solution

The High Speed Utilities for DB2 solution delivers high availability of objects during data and database maintenance. The solution also significantly reduces runtime costs by minimizing CPU consumption.

This solution includes the following products and primary technology components:

■ CHECK PLUS for DB2
BMC Database Administration for DB2 solution

You can use the Database Administration solution to manage your DB2 databases quickly, efficiently, and effectively.

This solution includes the following products and primary technology components:

- CATALOG MANAGER for DB2
- CHANGE MANAGER for DB2
- COPY PLUS for DB2
- LOADPLUS for DB2
- SNAPSHOT UPGRADE FEATURE for DB2 (SUF)
- UNLOAD PLUS for DB2

Other BMC solutions

UNLOAD PLUS is also used by the ALTER component of the BMC Administrative Assistant for DB2 solution.

Tasks that UNLOAD PLUS performs

UNLOAD PLUS accomplishes the standard unload and SELECT tasks and offers numerous functional enhancements such as the features that this section describes.
Performance

- Optionally produces multiple output data sets either by using multiple SELECT statements or by partition using a single SELECT statement
- Optionally allocates primary and secondary unload data sets dynamically, eliminating the need to include DD statements in the JCL
- Eliminates the processing of partitions that do not meet the conditions of the WHERE clause
- Optimizes the processing of tables in a multi-table segmented table space
  UNLOAD PLUS processes only those segments that pertain to the tables that it is unloading. UNLOAD PLUS does not read data in tables that the selection criteria do not specify.
- Optionally adjusts the number of tasks
- Optionally reduces overhead and considerably reduces processing time when moving data from one table to another table that has an identical layout (for example, when using FORMAT BMCLOAD with LOADPLUS)

Object availability

- Optionally unloads a consistent image of the data in a table space, (using XBM or SUF to produce a snapshot image) while enabling read/write access to the table space
- Runs concurrently with other BMC utilities on DB2 spaces

Resources

- Optionally offloads eligible processing to a zIIP
- Optionally offloads eligible queries to an IBM DB2 Data Analytics Accelerator for z/OS server

Unloading data

- Unloads rows from one or more DB2 tables with one utility invocation, including DB2 catalog tables or views
  These tables or views can be in multiple table spaces.
- Optionally unloads all data from a table space without requiring written SELECT statements
- Provides powerful selection criteria for unloading specific rows
- Optionally sorts the output records by specified columns, or by clustering or partitioning key
  For multi-table table spaces, UNLOAD PLUS groups sorted output records by table.
- Provides for unloading a sample of rows from a table or table space
- Provides the ability to limit the number of rows that it unloads
- Optionally provides a character string constant for output-record tagging that you specify or that the utility generates
- In addition to unloading from DB2 table space data sets, provides for unloading from the following types of copies and data sets:
  - Full or incremental image copies
  - Encrypted copies created by COPY PLUS
  - Cabinet copies created by the BMC Recovery Management for DB2 or BMC Recovery for DB2 solution
  - DSN1COPY data sets
  - Inline copies
  - Instant Snapshot copies created by COPY PLUS
  - Online consistent copies created by the Online Consistent Copy component of the Recovery Management or Recovery solution
  - VSAM linear data sets
  - IBM FlashCopy image copies
- Optionally allows UNLOAD PLUS to use data definition language (DDL) for DB2 object definitions instead of object definitions in the DB2 catalog when you use the INFILE ddname option to unload data
- Optionally produces load control statements for loading the data back into a DB2 table, or file definition statements to use with the following products:
  - LOADPLUS
  - Information Builders’ FOCUS
  - SAS
— CA Technologies’ CA Easytrieve

— Teradata

— IBM’s SQL/DS

— Select Business Solution’s NOMAD

- Optionally allocates primary and secondary unload data sets dynamically, removing the need to include DD statements in the JCL

**Operations on data**

- Provides a comprehensive set of data type conversions and a user-exit facility for conversions

- Provides options to perform data translation

- Allows replacement of null values with a user-specified constant or automatically generates a null indicator value with each output field for each selected nullable column

- Provides options and syntax to change the default null indicator character, or its position or length

- Supports a comma-separated-value (CSV) format for data that can be used by other DBMS platforms or spreadsheet applications

- Supports formatting output in XML that can be used by XML processing tools

- Allows you to put a constant value or CURRENT DATE, CURRENT TIME, CURRENT TIMESTAMP, or CURRENT RID in an output record

- Uses high-performance direct access to produce a fixed-length output file in which UNLOAD PLUS formats the unloaded data (when you specify FORMAT DSNTIAUL) to match the output of the IBM DSNTIAUL program

- Provides an option that activates a format modifier that fills numeric external data types with zeros when UNLOAD PLUS converts numeric data types to their external representations

- Provides an option to assign positive or negative overpunch values to decimal-zoned numeric values
Understanding UNLOAD PLUS execution modes

You can use the DIRECT command option to select one of two execution modes, or specify DIRECT AUTO to let UNLOAD PLUS choose between the two modes based on the options that you specify and the object that you are unloading.

The following table presents an overview of these modes and the benefits of each.

### Table 1: Execution modes

<table>
<thead>
<tr>
<th>Mode type</th>
<th>Description</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT YES</td>
<td>UNLOAD PLUS reads data directly from the table space data set or image copy data sets to unload the data, using a SELECT-like syntax for data selection. SELECT functionality is a limited subset of the functionality that is normally available in DB2 SQL.</td>
<td>The primary benefit of the DIRECT YES mode is maximum performance when unloading large volumes of data. Additionally, some features are available in DIRECT YES mode that are not available in DIRECT NO mode.</td>
</tr>
</tbody>
</table>
| DIRECT NO  | UNLOAD PLUS processes the SELECT statement and reads the data using DB2 dynamic SQL. This mode is not a high-performance solution for unloading large volumes of data. | The benefit of the DIRECT NO mode is a full range of DB2 SQL SELECT functionality, including joined tables, subqueries, and so on. This functionality includes many of the features that are available in UNLOAD PLUS including the following features:  
  - DB2 parallelism  
  - Data type conversions  
  - Output formatting |

**DIRECT YES**

When you use DIRECT YES, UNLOAD PLUS reads data directly from the table space data set or image copy data set and uses a SELECT-like syntax for data selection. In this mode, UNLOAD PLUS does not run as part of the DB2 subsystem and you must have system authorization similar to authorization that DB2 requires.

For the syntax options that are supported with DIRECT YES, see “Alphabetical listing of UNLOAD PLUS options” on page 80.
Using DIRECT YES provides the following capabilities that are not available when you use DIRECT NO:

- Unloading from full or incremental image copies, DSN1COPY data sets, encrypted copies created by COPY PLUS, cabinet copies, inline copies, Instant Snapshot copies, online consistent copies, VSAM linear data sets, or VSAM FlashCopy image copies
- Unloading all data from a table space without requiring written SELECT statements, thereby reducing maintenance
- Dynamically eliminating the processing of partitions that do not meet the conditions of the WHERE clause
- Sorting the data by clustering key or partitioning key
- Providing various levels of data availability by using the SHRLEVEL option
- Specifying logical partitions instead of physical partitions
- Unloading binary XML data to referenced files
- Unloading LOB and XML data to variable-block spanned (VBS) output files

**DIRECT NO**

When DIRECT NO is in effect, UNLOAD PLUS uses DB2 dynamic SQL to process the SELECT statement. In this mode, UNLOAD PLUS uses the DB2 subsystem to read the data, which means that UNLOAD PLUS competes with other DB2 processes for subsystem resources.

DIRECT NO mode identifies many of the characteristics of selected objects differently than DIRECT YES, which might result in some objects behaving differently under each mode. If you specify DIRECT AUTO, UNLOAD PLUS determines which mode to use, which might produce results that you do not expect.

Additionally, DIRECT NO does not support all options, including the following features:

- Unloading from image copies
- User-defined types
- UNLOAD TABLESPACE command
- SHRLEVEL option
  Instead, you can use isolation levels in the DB2 SQL WITH clause to control access to objects during the unload process.
For the syntax options that are supported with DIRECT NO, see “Alphabetical listing of UNLOAD PLUS options” on page 80.

When you use DIRECT NO, you obtain the following benefits that are not available when you use DIRECT YES:

■ The ability to use the full range of DB2 SQL SELECT functionality, including joined tables, subqueries, and so on

**Note**
You can use the INTO clause (available for both DIRECT YES and DIRECT NO) to use the UNLOAD PLUS data type conversions on the rows that were read using DB2 dynamic SQL.

■ The ability to use the SET CURRENT DEGREE command before executing any dynamically executed SELECT statements when you specify CURRENTDEGREE ANY or CURRENTDEGREE 1

■ The option to fetch multiple rows at one time (by using the ROWSETSZ option)

■ Timely enablement of DB2 features, such as

  — Enforcement of row- and column-level security
  — Tables that are defined as ORGANIZE BY HASH

**Note**
BMC plans to add DIRECT YES support for these features in a later version of UNLOAD PLUS.

---

**How UNLOAD PLUS works**

This section describes the following information about how UNLOAD PLUS works:

■ UNLOAD PLUS processing phases

■ The data sets that UNLOAD PLUS uses

■ The common components that UNLOAD PLUS uses
**DIRECT YES processing phases**

This topic describes the processing phases of UNLOAD PLUS and their primary functions when you use DIRECT YES.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
</table>
| UTILINIT   | During this phase, UNLOAD PLUS completes the following processes:  
  - Initializes the job  
  - Reads, parses, and verifies the UNLOAD command  
  - Processes object definition information from the DB2 catalog or DDLIN data set  
  - Dynamically allocates output data sets |
| UNLOAD     | During this phase, UNLOAD PLUS completes the following processes:  
  - Unloads table space data  
  - Sorts the data if you specify ordering  
  - Writes to the output data sets (SYSRECnn, SYSREDnn, and referenced files) |
| UTILTERM   | During this phase, UNLOAD PLUS completes the following processes:  
  - Performs the termination processing for the utility  
  - Creates the control statement data set (SYSCNTLn)  
  - Optionally updates the BMCHIST table in the UNLOAD PLUS database |
The following figure illustrates processing during each phase.

**Figure 1: UNLOAD PLUS processing phases when using DIRECT YES**

---

All phases use the SYSPRINT data set and update the BMCUTIL and BMCSYNC tables.

All sort processing uses the UTPRINT data set. The UTILTERM phase updates the BMCHIST table.
DIRECT NO processing phases

This topic describes the processing phases of UNLOAD PLUS and their primary functions when you use DIRECT NO.

Table 3: Processing phases of UNLOAD PLUS (DIRECT NO)

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>During this phase, UNLOAD PLUS completes the following processes:</td>
</tr>
<tr>
<td></td>
<td>■ Initializes the job</td>
</tr>
<tr>
<td></td>
<td>■ Reads, parses, and verifies the UNLOAD command</td>
</tr>
<tr>
<td></td>
<td>■ Extracts DB2 SQL SELECT statements</td>
</tr>
<tr>
<td></td>
<td>■ Prepares the SELECT statements</td>
</tr>
<tr>
<td></td>
<td>■ Processes object definition information related to the result table</td>
</tr>
<tr>
<td></td>
<td>■ Dynamically allocates output data sets</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>During this phase, UNLOAD PLUS completes the following processes:</td>
</tr>
<tr>
<td></td>
<td>■ Processes the SELECT statement</td>
</tr>
<tr>
<td></td>
<td>■ Uses DB2 dynamic SQL programming techniques to read the data</td>
</tr>
<tr>
<td></td>
<td>While fetching each row or rowset, UNLOAD PLUS converts the data to its</td>
</tr>
<tr>
<td></td>
<td>final output format and writes to the output data sets (SYSRECnn, SYSREDnn,</td>
</tr>
<tr>
<td></td>
<td>and referenced files).</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>During this phase, UNLOAD PLUS completes the following processes:</td>
</tr>
<tr>
<td></td>
<td>■ Performs the termination processing for the utility</td>
</tr>
<tr>
<td></td>
<td>■ Creates the control statement data set (SYSCNTLn)</td>
</tr>
<tr>
<td></td>
<td>■ Optionally updates the BMCHIST table in the UNLOAD PLUS database</td>
</tr>
</tbody>
</table>
The following figure illustrates processing that occurs during each phase.

**Figure 2: UNLOAD PLUS processing phases when using DIRECT NO**

All phases use the SYSPRINT data set and update the BMCLIB and BMCSYNC tables.

The UTILTERM phase updates the BMCHIST table.
UNLOAD PLUS data sets

This topic briefly describes the data sets that UNLOAD PLUS uses.

You can override some of the ddnames or ddname prefixes with UNLOAD command options. For more information about how to specify the DD statement names and data set allocation sizes, and when the data sets are required, see “UNLOAD PLUS DD statements” on page 276.

Data sets for DIRECT YES

The following table lists the data sets that UNLOAD PLUS uses when DIRECT YES is in effect.

Table 4: UNLOAD PLUS data sets for DIRECT YES

<table>
<thead>
<tr>
<th>Data set or ddname</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCFORCE</td>
<td>BMCFORCE is an optional data set that contains the thread cancelation report that is generated when you specify either FORCE REPORTONLY or FORCE_RPT YES.</td>
<td>“BMCFORCE data sets” on page 278</td>
</tr>
<tr>
<td>DDLIN</td>
<td>DDLIN is an optional input data set for the DDL that defines all of the objects that are related to the table from which UNLOAD PLUS unloads rows. UNLOAD PLUS uses the DDL for DB2 object definitions instead of object definitions in the DB2 catalog when you use the INFILE ddname option to unload data.</td>
<td>“DDLIN data sets in UNLOAD PLUS” on page 278</td>
</tr>
<tr>
<td>Input copy data sets</td>
<td>You can optionally specify a copy data set (full image copy or incremental image copy, DSN1COPY data set, or inline copy data set), instead of a DB2 table space, from which you want UNLOAD PLUS to unload rows.</td>
<td>“UNLOAD PLUS input copy data set” on page 279</td>
</tr>
<tr>
<td>Referenced files</td>
<td>These files are output files that you reference in your unload data sets. You unload LOB or XML data to these files instead of to your unload data sets.</td>
<td>“SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286</td>
</tr>
<tr>
<td>SORTWKnn</td>
<td>SORTWK is the data set that BMCSORT uses. The data set is used in the UNLOAD phase to sort unloaded rows. You cannot allocate the sort work files as VIO data sets or tape data sets. You must allocate each individual work file on a single DASD unit.</td>
<td>“SORTWK data sets in UNLOAD PLUS” on page 281</td>
</tr>
<tr>
<td>Data set or ddname</td>
<td>Description</td>
<td>Reference</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>SYSCNTLn</td>
<td>SYSCNTL is the output data set that contains the control statements that UNLOAD PLUS generated for the unloaded data. UNLOAD PLUS requires the n value only if you specify multiple data sets.</td>
<td>“SYSCNTL data sets in UNLOAD PLUS” on page 284</td>
</tr>
<tr>
<td>SYSIN</td>
<td>SYSIN is the input data set that contains the UNLOAD command.</td>
<td>“SYSIN data sets” on page 285</td>
</tr>
<tr>
<td>SYSPRIN2</td>
<td>SYSPRIN2 is an optional output data set that contains the same UNLOAD PLUS messages that are output to SYSPRINT. In a worklist environment, SYSPRIN2 enables you to view UNLOAD PLUS output in real time.</td>
<td>“SYSPRIN2 data sets” on page 285</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>SYSPRINT is the output data set that contains UNLOAD PLUS messages.</td>
<td>“SYSPRINT data sets” on page 285</td>
</tr>
<tr>
<td>SYSRECnn</td>
<td>SYSREC is the output data set that contains the unloaded rows. UNLOAD PLUS requires the nn value only if you specify multiple data sets. Throughout this book, the primary output unload data set is referred to as a SYSREC data set. The name that you use for SYSREC is different if you change it in the installation options module or override it by using the UNLOADDN option.</td>
<td>“SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286</td>
</tr>
<tr>
<td>SYSREDnn</td>
<td>SYSRED is an optional secondary output data set that contains the unloaded rows. UNLOAD PLUS requires the nn value only if you specify multiple data sets. Throughout this book, the secondary output unload data set is referred to as a SYSRED data set. The name that you use for SYSRED is different if you change it in the installation options module or override it by using the UNLOADDN option.</td>
<td>“SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286</td>
</tr>
</tbody>
</table>
### Data set or ddname

<table>
<thead>
<tr>
<th>Data set or ddname</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTPRINT</td>
<td>UTPRINT is the data set that indicates that sort messages should be reported. The actual messages for each sort process appear in separate SYSnnnnn data sets, where nnnnn is a system-assigned sequential number. This data set is required by any phase that performs a sort. <strong>WARNING:</strong> JES3 users should be aware of a limitation within JES3 that does not allow concurrent tasks to share SYSOUT data sets. See the description of the UTPRINT data set for more information about this limitation.</td>
<td>“UTPRINT data sets” on page 307</td>
</tr>
<tr>
<td>VSAMDDnn</td>
<td>VSAMDD is an optional VSAM linear data set from which UNLOAD PLUS unloads rows instead of the DB2 table space. UNLOAD PLUS requires the nn value only if you specify multiple data sets. The default ddname, VSAMDD, supports up to 99 partitions. For more than 99 partitions, or to use a ddname other than VSAMDD, you can use the VSAMDDPREFIX option.</td>
<td>“VSAMDD data sets in UNLOAD PLUS” on page 281</td>
</tr>
</tbody>
</table>

### Data sets for DIRECT NO

The following table shows the data sets that UNLOAD PLUS uses when DIRECT NO is in effect.

#### Table 5: UNLOAD PLUS data sets for DIRECT NO

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced files</td>
<td>These files are output files that you reference in your unload data sets. You can unload LOB or XML data to these files instead of to your unload data sets.</td>
<td>“SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286</td>
</tr>
<tr>
<td>SYSCNTLn</td>
<td>SYSCNTL is the output data set that contains the control statements that UNLOAD PLUS generates for the unloaded data. UNLOAD PLUS requires the n value only if you specify multiple data sets.</td>
<td>“SYSCNTL data sets in UNLOAD PLUS” on page 284</td>
</tr>
<tr>
<td>SYSIN</td>
<td>SYSIN is the input data set that contains the UNLOAD command.</td>
<td>“SYSIN data sets” on page 285</td>
</tr>
</tbody>
</table>
Data set | Description | See page
---|---|---
SYSPRIN2 | SYSPRIN2 is an optional output data set that contains the same UNLOAD PLUS messages that are output to SYSPRINT. In a worklist environment, this data set enables you to view UNLOAD PLUS output in real time. | “SYSPRIN2 data sets” on page 285
SYSPRINT | SYSPRINT is the output data set that contains UNLOAD PLUS messages. | “SYSPRINT data sets” on page 285
SYSRECnn | SYSREC is the output data set that contains the unloaded rows. UNLOAD PLUS requires the nn value only if you specify multiple data sets. | “SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286
SYSREDnn | SYSRED is an optional secondary output data set that contains the unloaded rows. UNLOAD PLUS requires the nn value only if you specify multiple data sets. | “SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286

**Associated products and common components that UNLOAD PLUS uses**

In addition to its own processing components, UNLOAD PLUS uses the BMC products and common components that this topic describes.

**Table 6: Common components that UNLOAD PLUS uses**

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCSORT</td>
<td>The BMCSORT technology is a common BMC technology. UNLOAD PLUS uses BMCSORT to allocate sort work files and to perform sort processing. This component is installed during UNLOAD PLUS installation, but is maintained separately from UNLOAD PLUS.</td>
</tr>
<tr>
<td>DB2 Solution Common Code</td>
<td>BMC DB2 Solution Common Code (SCC) is a set of technologies that provide common processes for several BMC products for DB2. UNLOAD PLUS uses SCC technologies for such processes as setting object statuses. This component is installed during UNLOAD PLUS installation, but is maintained separately from UNLOAD PLUS.</td>
</tr>
<tr>
<td>DB2 Utilities Common Code (D2U)</td>
<td>D2U is a set of technologies that provides common processes for the BMC Utility products for DB2 and the DASD MANAGER PLUS product. This component is installed during UNLOAD PLUS installation but is maintained separately from UNLOAD PLUS.</td>
</tr>
</tbody>
</table>
### Component Description

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTENDED BUFFER MANAGER (XBM) or SNAPSHOT UPGRADE FEATURE (SUF)</td>
<td>XBM or SUF provides the following capabilities:</td>
</tr>
<tr>
<td></td>
<td>■ A snapshot image of data in a table space</td>
</tr>
<tr>
<td></td>
<td>■ zIIP processing</td>
</tr>
<tr>
<td></td>
<td>XBM and SUF are licensed, installed, and maintained separately from UNLOAD PLUS.</td>
</tr>
</tbody>
</table>
Operational considerations in UNLOAD PLUS

This chapter provides procedures and information to consider when using UNLOAD PLUS.

System setup for UNLOAD PLUS

Review this section for recommendations and requirements before you use the UNLOAD PLUS for DB2 product.

DB2 support in UNLOAD PLUS

This version of UNLOAD PLUS supports IBM DB2 Versions 10 (new-function mode only) and 11.

UNLOAD PLUS for DB2 system and software requirements

The UNLOAD PLUS product has the following requirements:

- This version of UNLOAD PLUS has the following system requirements:
  - IBM System z10 processor or a later 64-bit z/Architecture processor
  - z/OS Version 1.10 or later

- For all types of unload jobs, you must have a minimum of the following versions of BMC common components:
  - Version 2.4.01 of BMCSORT
  - Version 11.2.00 of the DB2 Utilities Common Code (D2U)
  - Version 11.1.00 with PTF BPJ0944 of the DB2 Solution Common Code (SCC)
To use SHRLEVEL CHANGE CONSISTENT YES when unloading data, UNLOAD PLUS requires version 6.1.00 or later of either the BMC EXTENDED BUFFER MANAGER (XBM) product or its SNAPSHOT UPGRADE FEATURE (SUF).

If you use the XBMID option to specify a particular XBM subsystem, that subsystem must be at this maintenance level. If you do not specify a particular XBM subsystem and SHRLEVEL CHANGE CONSISTENT YES is in effect, UNLOAD PLUS searches for an XBM subsystem at this level.

To offload eligible processing to a zIIP, UNLOAD PLUS requires version 6.1.00 or later of either XBM or SUF.

If you use the XBMID option to specify a particular XBM subsystem, that subsystem must be at this maintenance level. If you do not specify a particular XBM subsystem and ZIIP ENABLED is in effect, UNLOAD PLUS searches for an XBM subsystem at this level.

**Required authorization for UNLOAD PLUS**

Using UNLOAD PLUS requires that you have the appropriate authorization within DB2 and through your system security package, such as the IBM Resource Access Control Facility (RACF) component of the z/OS Security Server.

You need sufficient authorization to access resources and perform the tasks accomplished during UNLOAD PLUS processing.

**Authorization verification mechanisms for the Utility products**

The products use one of the following mechanisms to verify authorization:

- If the DB2 DSNX@XAC authorization exit is available for your system, the product uses this exit to verify authorization for external access. The exit is available from the following sources:
  
  — IBM provides a sample exit with DB2 for the IBM Resource Access Control Facility (RACF) component.

  — CA Technologies provides the DSNX@XAC exit with the CA-ACF2 and CA-Top Secret products.

  BMC recommends this mechanism for implementing external security. The access control authorization exit must be available in the STEPLIB, JOBLIB, linklist, or in the SYS3.DSN exit.
If the DSNX@XAC exit is not available, the product uses the standard DB2 method to check security.

### Setting UNLOAD PLUS authorizations

UNLOAD PLUS does not run as part of the DB2 subsystem. Therefore, users must have system authorizations and, for DIRECT YES, data set authorizations that are equivalent to the authorizations that DB2 requires. Use the following procedures to set the necessary authorizations.

**Note**

If you are using UNLOAD PLUS with ALTER for DB2 or CHANGE MANAGER for DB2, UNLOAD PLUS functions in DIRECT YES mode only.

### To set DB2 authorizations

1. For all load jobs, set the following authorizations:

   - Sufficient DB2 authority to execute the UNLOAD PLUS plan and all packages that the UNLOAD PLUS plan uses
   - Authorization equivalent to the authorization that the IBM DB2 UNLOAD utility requires

   **Note**

   UNLOAD PLUS enforces row- and column-level security only when DIRECT NO is in effect.

2. To enable the use of the FORCE option to cancel DB2 threads that might prevent a successful drain during an unload job, grant the following authorizations:

   - DISPLAY privileges
   - One of the following authorities:
     - SYSADM
     - SYSOPR
     - SYSCTRL

   **Note**

   These authorizations might be implicit in the authority that the users have.

3. To enable zIIP processing and SHRLEVEL CHANGE CONSISTENT YES, ensure that you have the appropriate authorizations for XBM or SUF.
For information about security levels and authorizations for XBM, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

**To enable data set access using the DB2 RACF ID**

1. Specify OPNDB2ID=YES in your installation options.
   
   This option tells UNLOAD PLUS to use the DB2 RACF ID for data set access.

**To enable data set access when not using the DB2 RACF ID**

When using DIRECT NO, UNLOAD PLUS uses DB2 to access data sets. In this case, users do not need the authorization described in this procedure.

1. Specify OPNDB2ID=NO in your installation options.
   
   This option tells UNLOAD PLUS not to use the DB2 RACF ID for data set access.

2. If using RACF or a similar system security package to protect underlying data sets and the Integrated Catalog Facility (ICF) catalog of a table or index space, grant READ privileges for the following sources:
   - DB2 VSAM data sets
   - DB2 image copy data sets
   - DSN1COPY data sets
   - Inline copy data sets
   - Instant Snapshot copy data sets
   - Online consistent copy data sets
   - Cabinet copy data sets
   - VSAM FlashCopy data sets
   - VSAM linear data sets
   - Encrypted copy data sets that are created by COPY PLUS
   - Key data sets for encrypted copies

The following steps illustrate one method for granting these data set authorizations when your site uses a system security package other than RACF:

1. Associate users with a security group.

2. Grant EXECUTE privileges on the UNLOAD PLUS product program (ADUUMAIN) to the security group.

3. Grant the data set authorizations to ADUUMAIN.
Overriding the MEMLIMIT system parameter

The utility requires above-the-bar memory and might abend if sufficient memory is not available.

The default value for the System Management Facility (SMF) MEMLIMIT parameter is 2 GB. This value is set in member SMFPRMxx in SYS1.PARMLIB.

To override the default MEMLIMIT value

1. Use one of the following methods to override the default MEMLIMIT value:
   - (recommended) Specify REGION=0M in the JCL.
   - Specify the one of the following values for the MEMLIMIT parameter in the JCL, or in an SMF IEFUSI exit:
     - Specify NOLIMIT to allow unlimited above-the-bar memory.
     - If you are unable to specify NOLIMIT, specify at least 4 GB. If you are operating on LOB or XML data, specify at least 32 GB.

Number of DB2 threads that UNLOAD PLUS uses

The number of DB2 threads that UNLOAD PLUS uses depends on the value of the DIRECT option that is in effect.

DIRECT YES

The maximum number of batch threads that UNLOAD PLUS uses concurrently is six per job. UNLOAD PLUS uses up to six threads during the UTILINIT phase and two threads for the duration of the job.

DIRECT NO

When you specify DIRECT NO, UNLOAD PLUS uses two batch threads during the UTILINIT phase, and uses one of these threads during the entire execution.

In the UNLOAD phase, each task uses one batch thread to process a SELECT statement. Therefore, the maximum number of batch threads that UNLOAD PLUS uses concurrently during the UNLOAD phase is equal to the number of tasks that can execute concurrently. When you specify DIRECT NO, UNLOAD PLUS uses the
MAXCONNECT command option to control the maximum number of tasks that can execute concurrently.

**UNLOAD PLUS serialization and concurrency**

This section discusses concurrency issues and object status requirements, which can vary with the command statement specifications.

**Executing BMC utilities concurrently**

All BMC Software utility products use the BMCSYNC and BMCUTIL tables to control concurrent access to DB2 objects.

The utility products use the BMCUTIL table to control the use of utility IDs, which identify executions of BMC utilities. Each BMC utility product must have a unique ID for restart purposes. This unique ID is stored in the BMCUTIL table. For more information about this table, see “BMCUTIL table” on page 488.

**Note**
The setting of the LOCKROW installation option determines whether the utility uses MVS enqueues or SQL LOCK TABLE statements to serialize the BMCSYNC and BMCUTIL tables. For information about the LOCKROW option, see the installation options descriptions for the utility.

**Shared access levels of BMC utilities**

BMC utility jobs register DB2 objects in the BMCSYNC table.

The registering utility assigns a sharing level to each registered object. The sharing level controls access to that object from other BMC utilities. For partitioned DB2 spaces, registration is performed at the partition level.

**Note**
All BMC utility products use the BMCUTIL table to control the use of utility IDs, which identify executions of BMC utilities. Each BMC utility product must have a unique ID for restart purposes. This unique ID is stored in the BMCUTIL table. For more information about this table, see “BMCUTIL table” on page 488.

The BMCSYNC table allows multiple BMC utilities (or multiple instances of a utility) to operate concurrently on different partitions of a DB2 space if no nonpartioning indexes are involved. In addition, some BMC utilities can operate concurrently on
the same object or partition. For information about which products can operate
concurrently, see the following table. For additional serialization and concurrency
issues for each utility, see that utility's reference manual.

The "Access level" column in the following table refers to the value of the
\texttt{SHRLEVEL} column name in the BMCSYNC table ("BMCSYNC table" on page 478).
The level can be one of the following values:

- S indicates shared access. Any other utility that registers with shared access (S)
can run against the object.
- X indicates exclusive access. No other utility can run against the object.
- A blank value indicates that no status is requested and any other utility can run
against the object.

\textbf{Table 7: Shared access levels of BMC utilities}

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK PLUS</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>COPY PLUS</td>
<td>S or blank</td>
<td>If you specify COPY IMAGECOPY, COPY PLUS registers the object with no access status (blank). Otherwise, COPY PLUS registers the object with shared access (S).</td>
</tr>
<tr>
<td>DASD MANAGER PLUS</td>
<td>S</td>
<td>If BMCSTATS is processing multiple objects and encounters an object that is held by another utility, the BMCSTATS job issues a warning. The warning identifies the object and the utility that is using it. BMCSTATS continues processing the next object. If BMCSTATS is processing an object and another utility requires exclusive control of that object, the other utility stops execution at initialization time.</td>
</tr>
<tr>
<td>(BMCSTATS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOADPLUS</td>
<td>X</td>
<td>If you specify PART, LOADPLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions while running this job.</td>
</tr>
</tbody>
</table>
**UNLOAD PLUS serialization and concurrency**

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER PLUS</td>
<td>X, S, or blank</td>
<td>RECOVER PLUS registers an object with shared access (S) under the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The table space for an index is registered with shared access if the index is being rebuilt and its table space is not recovered in the same job.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ A table space partition is registered with shared access if the keys for that partition are unloaded with a RECOVER UNLOADKEYS operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER PLUS registers an object with no access status (blank) if you specify the following commands or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The ACCUM command</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ OUTCOPY ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ INDEPEND OUTSPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER PLUS registers the object with exclusive access (X) in all other cases.</td>
</tr>
<tr>
<td>RECOVERY MANAGER</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>REORG PLUS</td>
<td>X</td>
<td>If you specify PART, REORG PLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions while running this job.</td>
</tr>
<tr>
<td>UNLOAD PLUS</td>
<td>S</td>
<td>None</td>
</tr>
</tbody>
</table>

**WARNING**

Do not run an IBM utility, command, or SQL statement that attempts to manipulate the structure, data, or status of an object that a BMC utility is currently processing. For example, commands and SQL statements such as -STOP, -START, EXCHANGE, and ALTER will produce unpredictable results.

---

### Object status with DIRECT YES

This section describes initial database and table space status requirements, and describes how UNLOAD PLUS changes the status of the objects during and after processing.
Initial status for DIRECT YES

To unload your table space when using DIRECT YES, the table space that you are unloading and the database associated with that table space cannot be in any of the following statuses:

- Auxiliary CHECK pending (ACHKP)
- CHECK pending (CHKP)
- Group RECOVER pending (GRECP)
- Logical page list (LPL)
- Page set REBUILD pending (PSRBD)
- REBUILD pending (RBDP)
- Logical part REBUILD pending (RBDP*)
- RECOVER pending (RECP)
- Refresh pending (REFP)
- REORG pending (REORP)
- Restart pending (RESTP)
- Utility restrictive state, utility exclusive control (UTUT)
- Write error page range (WEPR)

In addition, the database that is associated with the table space that you are unloading cannot be in utility only (UT) status. This restriction does not apply when you specify SHRLEVEL CHANGE CONSISTENT NO QUIESCE NO.

If you are unloading from copies or VSAM linear data sets, the status of the table space is irrelevant.

Status changes during DIRECT YES processing

The following sections describe how object status changes during unload processing for different types of unload jobs.

For more information about the SHRLEVEL option, see “SHRLEVEL” on page 103.
**SHRLEVEL REFERENCE**

SHRLEVEL REFERENCE restricts object status to read-only during UNLOAD PLUS processing. UNLOAD PLUS then externalizes the pages of the table space. When operating on partitioned objects, access is restricted only to those partitions that are being unloaded. When UNLOAD PLUS processing completes, the utility restores the object to its original status.

**SHRLEVEL CHANGE**

SHRLEVEL CHANGE allows read/write access to the table space during unload processing. UNLOAD PLUS does not have concurrency issues when you specify SHRLEVEL CHANGE CONSISTENT NO. However, if you specify SHRLEVEL CHANGE CONSISTENT YES, access to the table space is briefly restricted. For more details, see the following sections.

**CONSISTENT NO**

When you specify SHRLEVEL CHANGE CONSISTENT NO, access to the object is not restricted.

The following table describes status changes based on the value of the QUIESCE option when you specify CONSISTENT NO.

### Table 8: Status changes based on the value of QUIESCE

<table>
<thead>
<tr>
<th>QUIESCE option</th>
<th>Status change</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUIESCE or QUIESCE YES</td>
<td>When you specify SHRLEVEL CHANGE CONSISTENT NO QUIESCE YES, UNLOAD PLUS briefly restricts access to the object while externalizing pages from the DB2 buffer pool.</td>
</tr>
<tr>
<td>QUIESCE NO</td>
<td>When you specify SHRLEVEL CHANGE CONSISTENT NO QUIESCE NO, UNLOAD PLUS does not restrict access to the objects nor does it externalize pages from the DB2 buffer pool. <strong>WARNING:</strong> Due to this process, UNLOAD PLUS might not process any updated pages in the buffer pool. In addition, if encountering any pages that have uncommitted data, UNLOAD PLUS unloads the uncommitted data.</td>
</tr>
</tbody>
</table>

**CONSISTENT YES**

When you specify SHRLEVEL CHANGE CONSISTENT YES, UNLOAD PLUS briefly restricts access to the table space while externalizing the pages.
Object status with DIRECT NO

When DIRECT NO is in effect, UNLOAD PLUS terminates when a participating object is in a restrictive status that is not supported for DB2 dynamic SQL.

UNLOAD PLUS does not support the SHRLEVEL option with DIRECT NO. Instead, you can use isolation levels, available in the DB2 SQL WITH clause, to control access to objects during the unload process. For example, to allow read-only access to the objects while using DIRECT NO, you can specify WITH RR in the SELECT statement.

Drain and lock table processing in UNLOAD PLUS

UNLOAD PLUS can drain or lock objects, depending on certain conditions.

**Note**

When you are performing a partial unload (using the PART keyword or FILTERPART), the drain information in this section applies to the participating partitions.

UNLOAD PLUS locks tables only when all of the following conditions exist:

- You are unloading a nonpartitioned table space with multiple tables.
- The table space is not in UT status.
- You are not unloading all tables in the table space.

In all other cases, UNLOAD PLUS drains the objects when required.

**How long the drain or lock is held**

The drain or lock is held throughout the entire unload process with the following exceptions:

- For SHRLEVEL CHANGE CONSISTENT YES, the drain or lock is held throughout the XBM or SUF registration process (including quiesce).
- For SHRLEVEL CHANGE CONSISTENT NO QUIESCE YES, the drain or lock is held throughout the quiesce process.

**Controlling drain behavior**

You can use the following command options to control drain behavior:
■ DRAIN_WAIT

■ RETRY

■ RETRY_DELAY

■ FORCE

If UNLOAD PLUS cannot drain all of the objects within the time period specified by DRAIN_WAIT, UNLOAD PLUS releases the drains that it has obtained so far, waits for the length of time specified in the RETRY_DELAY command option, and tries to drain the objects again. UNLOAD PLUS retries for the number of times specified in the RETRY command option. The FORCE option can change this process by enabling you to specify if and when you want to cancel any threads that might prevent a successful drain.

**Note**
You can also use installation options to set defaults for these command options.

### Controlling lock behavior

To control lock behavior, you can use the SQLDELAY and SQLRETRY installation options. If UNLOAD PLUS cannot acquire a lock on a table, it waits for the length of time specified in the SQLDELAY option and then tries to acquire a lock again. UNLOAD PLUS retries for the number of times specified in the SQLRETRY option.

---

**Dynamic data set allocation in UNLOAD PLUS**

UNLOAD PLUS allows you to dynamically allocate your unload data sets and sort work data sets.

UNLOAD PLUS dynamically allocates any data sets needed for referenced files when unloading LOB or XML data. This section describes considerations for dynamic allocation.

Allocating unload data sets dynamically with UNLOAD PLUS allows you to:

■ Eliminate large, complex DD statements

■ Release unused space when UNLOAD PLUS closes an unloaded data set

■ Use symbolic variables and generation data groups (GDGs) to support data set name generation
Reduce JCL maintenance when object definitions change (such as when partitions are added, or a new table is added to a table space)

Reduce the manual process of sizing unload data sets by performing automatic size estimation based on object characteristics

Leverage your storage management subsystem (SMS) configuration to minimize x37 abends or inadequate space abends

For information about allocating a particular data set type, see “UNLOAD PLUS DD statements” on page 276.

Running SHRLEVEL CHANGE CONSISTENT YES

To enable read/write access to your data during unload processing, UNLOAD PLUS uses the snapshot-processing features of XBM or SUF to unload a consistent image of the data.

This feature is available only when DIRECT YES is in effect.

XBM and SUF are licensed, installed, and maintained separately from UNLOAD PLUS. You can use either XBM or SUF, depending on the license that you have obtained:

- A license for the full version of the XBM product authorizes you to use all XBM features.

- A license for SUF authorizes you to use only the snapshot and zIIP-processing features of XBM.

Note

If you are licensed only for a BMC solution that contains UNLOAD PLUS, your license authorizes you to use SUF, not the full version of XBM.

For detailed information about XBM and SUF, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

To unload a consistent data image

1. Ensure that a currently supported version of XBM or SUF is installed and available to UNLOAD PLUS.

XBM and SUF require a started task.
2 Create and activate the XBM management set that contains the appropriate snapshot object definition.

3 Create and activate the XBM configuration that contains the appropriate cache attributes.

For proper UNLOAD PLUS performance, ensure that XBM or SUF is configured with sufficient cache for the number of concurrent snapshots and the level of DB2 update activity at your site. For specific cache configuration information, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

4 Specify SHRLEVEL CHANGE CONSISTENT YES on your UNLOAD command statement.

5 (optional) To use a specific XBM subsystem, specify that subsystem ID with either the XBMID installation or command option.

6 (optional) Specify the WTOMSG option to write a message to the MVS log that indicates that snapshot-processing initialization for UNLOAD PLUS has successfully completed.

Unloading LOB and XML data

You can unload LOB and XML data to either your unload data sets or files that are referenced in your unload data set.

**Note**

You can unload inline LOB data only to unload data sets.

Unloading LOB and XML data to unload data sets

You can unload LOB (including inline LOB) or XML data to the following types of unload data sets:

- (DIRECT YES only) VBS unload data sets
- (DIRECT NO only) standard unload data sets

**To unload LOB or XML data to unload data sets**

1 Ensure that the following options are in effect, based on the type of unload data set that you intend to use:
2 Use one of the following options for your field specification:

- Include no field specification on your UNLOAD command.

- Include a field specification for the column that you are unloading:
  - For an XML column, specify the XML data type.
  - For a LOB column, specify the BLOB, CLOB, or DBCLOB data type.

3 Adjust your job based on following the considerations:

- “General considerations when unloading LOB and XML data” on page 57

- “Considerations when unloading LOB or XML data to unload data sets” on page 58

### Unloading LOB and XML data to referenced files

You can unload LOB or XML data to referenced files when either DIRECT YES or DIRECT NO is in effect.

**Note**

You cannot unload inline LOB data to referenced files.

**To unload LOB or XML data to referenced files**

1 If you are unloading to a hierarchical file system (HFS), ensure that the file system is preallocated and mounted.

   The file system must be mounted on the same MVS system on which you are running UNLOAD PLUS.

2 Specify an OUTPUT statement for each referenced file to which you are unloading.

   Include the following options:

   - Descriptor name
DSNAME

— For DSNTYPE PDS or LIBRARY, the data set name that you specify must not currently exist and must not include a member name. UNLOAD PLUS dynamically allocates this data set.

— For DSNTYPE HFS, you must specify the fully qualified path name for the file system, but no file name. UNLOAD PLUS generates the files.

DSNTYPE (PDS, LIBRARY, or HFS)

(required for DSNTYPE PDS or LIBRARY) DIR

(required for DSNTYPE PDS or LIBRARY) SPACE

(required for DIRECT NO) 0 or 1 for UNITCNT

(optional for DSNTYPE PDS or LIBRARY) Additional disk options that you want UNLOAD PLUS to use to dynamically allocate the PDS or PDSE

Note

Tape options are not valid for a referenced file.
For DSNTYPE HFS, UNLOAD PLUS honors only the descriptor name, DSNAME, DSNTYPE, and SUBSETS options.

For more information about these options, see “OUTPUT command options” on page 182.

3 Specify the appropriate CHAR or VARCHAR subtype (BLOBF, CLOBF, or DBCLOBF) in your field specification.

Ensure that you include the associated output descriptor name, as described in “Considerations when unloading LOB or XML data to referenced files” on page 60.

4 If you are unloading XML data to a BLOB file, specify the BINARYXML keyword with BLOBF.

5 Adjust your job based on the following considerations:

“General considerations when unloading LOB and XML data” on page 57

“Considerations when unloading LOB or XML data to referenced files” on page 60
General considerations when unloading LOB and XML data

The following considerations apply to both methods of unloading LOB or XML data.

DIRECT YES restrictions

When DIRECT YES is in effect, UNLOAD PLUS terminates when any of the following conditions exists:

- You specify multiple SELECT statements.
- UNLOAD PLUS encounters an error during LOB or XML data processing (for example, a translation error).

DIRECT NO restrictions

When DIRECT NO is in effect, UNLOAD PLUS terminates if you specify BINARYXML on your field specification.

Incompatible options

The following table lists the command and installation options that are incompatible when you unload LOB or XML data. The table indicates how UNLOAD PLUS responds if your job contains these options.

<table>
<thead>
<tr>
<th>Command or installation option</th>
<th>UNLOAD PLUS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZE</td>
<td>Ignores this option for referenced files</td>
</tr>
<tr>
<td>CNTLCARDS other than DB2, DB2LOAD, or DB2DDL</td>
<td>Terminates</td>
</tr>
<tr>
<td>ESTROWS</td>
<td>Ignores this option for referenced files</td>
</tr>
<tr>
<td>FIXEDVARCHAR YES</td>
<td>Ignores this option for referenced files</td>
</tr>
<tr>
<td>FORMAT BMCLOAD</td>
<td>Terminates</td>
</tr>
<tr>
<td>FORMAT CSV</td>
<td></td>
</tr>
<tr>
<td>FORMAT DSNTIAUL</td>
<td></td>
</tr>
<tr>
<td>FORMAT EXTERNAL</td>
<td></td>
</tr>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>Terminates</td>
</tr>
<tr>
<td>INFILE</td>
<td>Terminates</td>
</tr>
<tr>
<td>MAXBLKSIZE</td>
<td>Ignores this option for referenced files</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Terminates if specified on a LOB or XML column</td>
</tr>
</tbody>
</table>
Other considerations

- To unload LOB or XML data, specify the base table that contains the LOB or XML column. You cannot unload LOB or XML data directly from the auxiliary or XML table.

- You can specify a LOB or XML column on a WHERE predicate only if the right side of the predicate is NULL. For any other predicates, UNLOAD PLUS terminates.

- UNLOAD PLUS terminates if you attempt to convert BLOB, CLOB, DBCLOB, or XML data to a different data type.

- You cannot unload LOB or XML data from a LOB or XML table space copy. However, you can unload the base table from a copy if you do not select a LOB or XML column.

- When unloading LOB or XML data, you must supply sufficient above-the-bar storage to allow UNLOAD PLUS to process. BMC recommends that you specify REGION=0M or that you specify at least 32 GB for the MEMLIMIT parameter.

Considerations when unloading LOB or XML data to unload data sets

In addition to the general considerations, the considerations in this section apply when unloading LOB or XML data to unload data sets.

In addition to the general LOB and XML data considerations ("General considerations when unloading LOB and XML data" on page 57), the following considerations apply when you unload your LOB or XML data to unload data sets.

Inline LOB data

For inline LOB data, UNLOAD PLUS includes the INLINE LENGTH keyword in the field specification of your generated control cards only if DIRECT YES is in effect.
Standard unload data sets

For standard unload data sets, the output record for an unload job that includes LOB or XML data cannot be greater than 32 KB. If you attempt to unload LOB or XML columns that create an output record that is greater than 32 KB, UNLOAD PLUS discards that record.

Note
For larger output records, consider unloading to VBS data sets using SPANNED YES.

VBS unload data sets with SPANNED YES

When you specify SPANNED YES, UNLOAD PLUS unloads the data to a standard VBS data set. Unloading to these VBS data sets provides the following advantages:

- You can unload output records that are greater than 32 KB.
- You can take advantage of DIRECT YES features when unloading to an unload data set.

Note
The format of this data set is similar to the IBM proprietary spanned record format. However, the FORMAT SPANNED YES option of the IBM DB2 LOAD utility requires that the input be in this proprietary spanned record format. Therefore, you cannot use the IBM DB2 LOAD utility to load the VBS data set that UNLOAD PLUS produces.

General considerations

The following general considerations apply when you unload to VBS unload data sets with SPANNED YES:

- UNLOAD PLUS terminates if you use SYSOUT=* as your SYSREC DD specification.
- UNLOAD PLUS overrides any record format or record size that you specify in your JCL for output data sets.
- You can use only LOADPLUS to load these data sets that UNLOAD PLUS creates.
- UNLOAD PLUS adds the LOADPLUS option FORMAT SPANNED YES to the control cards that it generates.

Field specification considerations
The following considerations apply to your field specifications when you unload to VBS unload data sets with SPANNED YES:

- The XML and LOB columns must be listed at the end of the field specification.
- UNLOAD PLUS ignores any referenced file data types included in your field specification.

**Additional option considerations**

The following considerations apply to the options that you specify when you unload to VBS unload data sets with SPANNED YES:

- UNLOAD PLUS terminates when you specify SPANNED YES and any of the following options are also in effect:
  - ORDER YES
  - ORDER BY
  - Multiple SELECT statements
- If you specify NULLTYPE with an L value, UNLOAD PLUS changes it to the comparable T value.
- When you specify SPANNED YES, UNLOAD PLUS ignores FIXEDVARCHAR YES.

### Considerations when unloading LOB or XML data to referenced files

In addition to the general considerations, the following considerations apply when unloading LOB or XML data to referenced files.

#### Linking referenced files to the output fields

To identify which LOB or XML data to unload to which referenced file, you specify the output descriptor name with the BLOBF, CLOBF, or DBCLOBF subtype of the CHAR or VARCHAR data type.

The following figure shows how the OUTPUT statements relate to the field specification.

**Figure 3: Example OUTPUT statements and field specification for unloading to referenced files**

```
OUTPUT SYSREF1 DSNAME('/home/rdajdm/lobfr1')
DSNTYPE(HFS)
OUTPUT SYSREF2 DSNAME('RDAJDM.UNLOAD.LOBFR2')
```
OUTPUT statement considerations

This section describes considerations regarding OUTPUT statements for referenced files.

Note

When unloading LOB or XML data, any SYSREC or SYSRED OUTPUT statements apply to unloading the base table only.

PDS and PDSE referenced files

The following considerations apply to OUTPUT options for PDS and PDSE referenced files:

- You can use the VOLUMES option to specify the volume that you want UNLOAD PLUS to use. However, if you specify more than one volume, UNLOAD PLUS writes to only the first volume in the list.

- If you include the &PART variable in a pattern for DSNAME, UNLOAD PLUS creates one data set for each partition of the base table space. If you do not include the &PART variable, UNLOAD PLUS creates a single data set.

- The following OUTPUT options are incompatible:

Table 10: Incompatible OUTPUT options for PDS and PDSE referenced files

<table>
<thead>
<tr>
<th>Option</th>
<th>UNLOAD PLUS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any tape option</td>
<td>Terminates</td>
</tr>
<tr>
<td>DISKEXPD</td>
<td>Terminates</td>
</tr>
<tr>
<td>DISKRETN</td>
<td>Terminates</td>
</tr>
<tr>
<td>PCTPRIM</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>MAXPRIM</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>MAXSECD</td>
<td>Ignores this option</td>
</tr>
</tbody>
</table>
### Option and UNLOAD PLUS response

<table>
<thead>
<tr>
<th>Option</th>
<th>UNLOAD PLUS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBRSECD</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>FILESZPCT</td>
<td>Ignores this option</td>
</tr>
</tbody>
</table>

### HFS referenced files

For HFS referenced files, UNLOAD PLUS honors only the following OUTPUT options:

- Descriptor name
- DSNAME
- DSNTYPE
- SUBSETS

UNLOAD PLUS responds as follows to the other OUTPUT options:

- If you specify any tape options, UNLOAD PLUS terminates.
- UNLOAD PLUS ignores all other OUTPUT options.

The following information applies to specifying a pattern with the DSNAME option for HFS referenced files:

- When you specify DIRECT YES, you can use patterns to have UNLOAD PLUS create additional subdirectories in your file system path. In the following example, the root portion of the path (/home/rdacxb) already exists and is the mount point for the file system. UNLOAD PLUS adds subdirectories to this path for the subsystem ID and partition number based on the two variables in the DSNAME pattern.

  ```sql
  OUTPUT CLOB01 DSNAME('/home/rdacxb/&SSID/p&PART') DSNTYPE(HFS)
  ```

  **Note**

  When you specify DIRECT NO, specifying a pattern does not result in additional subdirectories in your file system path. UNLOAD PLUS terminates if it cannot find a file system path that uses the name that resolves from your pattern.

- If you include the &PART variable, UNLOAD PLUS creates one HFS path subdirectory for each partition of the base table space.

### Restrictions

The following restrictions apply to referenced output files:
You cannot unload XML data to referenced files defined as DBCLOB.

You cannot unload LOB data to referenced files of a different type (for example, you cannot unload CLOB data to a BLOB referenced file).

The number of data sets that UNLOAD PLUS will allocate for referenced files is limited to 256. If the options that you have specified cause UNLOAD PLUS to attempt to allocate more than 256 data sets, UNLOAD PLUS terminates.

The following restrictions apply when you specify DELETEFILES YES:

— When DIRECT NO is in effect, UNLOAD PLUS does not delete any referenced files.

— UNLOAD PLUS does not delete any HFS files. If UNLOAD PLUS abends when unloading to an HFS file, you must manually restore that file.

**Additional considerations**

The following additional considerations apply to referenced output files:

- When DIRECT YES is in effect, you can unload your data to multiple referenced files. For additional information about using multiple referenced files, see “Using referenced files” on page 287.

- UNLOAD PLUS unloads each XML document or LOB to a separate member (for a PDS or PDSE) or file (for an HFS).

- UNLOAD PLUS generates HFS file names and PDS and PDSE member names as eight-byte names. The first character of the name is a letter and the remaining seven characters are alphanumeric characters.

- When DIRECT YES is in effect, UNLOAD PLUS always creates a single, empty PDS or PDSE member to represent all rows that contain empty LOB and XML columns. UNLOAD PLUS creates this member regardless of the existence of an empty column.

**Considerations for unloading DB2 catalog data**

DB2 stores catalog data as Unicode and stores many commonly used fields as UTF-8 data. To unload catalog data, you must specify DIRECT NO.

The following considerations apply when unloading catalog data:
Considerations for unloading to a BatchPipes file

UNLOAD PLUS allows you to unload to an IBM BatchPipes file.

The following considerations apply when using data pipes:

- UNLOAD PLUS cannot multitask when unloading to a single data pipe. When unloading to a single pipe, set the appropriate option based on whether you are running DIRECT YES or DIRECT NO:
  - For DIRECT YES, specify MAXSORTS 1
  - For DIRECT NO, specify MAXCONNECT 1

- If you are migrating data by using BatchPipes with UNLOAD PLUS and LOADPLUS and you are unloading data to multiple pipes, you must use a separate unload job for each pipe. You cannot unload data to multiple pipes that are generated by multiple steps within a single job.

For additional considerations that apply to BatchPipes input, see the LOADPLUS for DB2 Reference Manual.

UNLOAD PLUS output format

You can use the FORMAT option to specify the format of your output data.

The following table briefly describes the keywords of the FORMAT option so that you can quickly compare the types of formats and determine which format best meets your needs.
# Table 11: FORMAT option keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
</table>
| STANDARD   | Provides output data in DB2 internal format except date, time, and timestamp data types, which are in external format  
This is the default for the FORMAT option.                                                                                                   | “DB2 output with external date and time (FORMAT STANDARD)” on page 66                         |
| EXTERNAL   | Provides all output data in external format  
This output enables porting to other databases and applications, but might not be as optimal as the other options.                                                                 | “External-formatted output (FORMAT EXTERNAL)” on page 66                                         |
| INTERNAL   | Provides all output data in DB2 internal format                                                                                                                                                            | “DB2 output with internal date and time (FORMAT INTERNAL)” on page 67                           |
| BMCLOAD    | Provides data in an internal format that only LOADPLUS can use  
This internal format is a high-speed migration solution for DB2 data.                                                                                                                                  | “Data migration using LOADPLUS (FORMAT BMCLOAD)” on page 67                                     |
| DSNTIAUL   | Provides output data that matches the output of the IBM DSNTIAUL program                                                                                                                                  | “DSNTIAUL-formatted output (FORMAT DSNTIAUL)” on page 71                                        |
| CSV        | Provides output data in comma-separated-value form  
This output enables porting to other databases and applications.                                                                                                                                       | “CSV output (FORMAT CSV)” on page 72                                                            |
| XML        | Provides output data in which each field is enclosed by XML tags  
This output enables porting to other databases and applications.                                                                                                                                       | “XML output (FORMAT XML)” on page 74                                                            |

The topics that follow describe how to use UNLOAD PLUS to unload data in specific formats. For additional information, see the following references:

- For examples, see “Examples of FORMAT option output” on page 76.
- For the FORMAT option, see “FORMAT” on page 149.
- For performance information, see “Specifying command options for optimal performance in UNLOAD PLUS” on page 401.
DB2 output with external date and time (FORMAT STANDARD)

To produce an output file in which the unloaded data (except date, time, and timestamp data types) is in DB2 internal format, use FORMAT STANDARD (the default).

Date, time, and timestamp data is in external format.

**Note**
FORMAT STANDARD does not produce output that matches that of DSNTIAUL.

Use this option when either of the following conditions exists:

- You plan to load the data into a DB2 subsystem where the date and time formats differ from those of the source DB2 subsystem and application programs require dates in a specified format.
- You are unloading LOB or XML data.

**Note**
Alternatively, you can specify FORMAT INTERNAL to unload LOB or XML data.

External-formatted output (FORMAT EXTERNAL)

To produce an output file in which all of the unloaded data is in external format, specify FORMAT EXTERNAL.

Use this option only when applications require data in this format, or when you are loading the data to a non-MVS target and you cannot use one of the other format options.

**Note**
You cannot use this option when you are unloading LOB or XML data. UNLOAD PLUS terminates in this case.
DB2 output with internal date and time (FORMAT INTERNAL)

To produce an output file in which all of the unloaded data is in internal format, specify FORMAT INTERNAL.

Use this option when either of the following conditions exists:

- You plan to load the data into a DB2 subsystem where the date and time formats are the same as those of the source DB2 subsystem and application programs can process dates in internal DB2 format.
- You are unloading LOB or XML data.

**Note**
Alternatively, you can specify or default to FORMAT STANDARD to unload LOB or XML data.

**Restriction**

If you specify USELRECL YES with FORMAT INTERNAL, UNLOAD PLUS ignores USELRECL YES.

Data migration using LOADPLUS (FORMAT BMCLOAD)

To unload data that you plan to reload by using LOADPLUS, specify FORMAT BMCLOAD.

This option provides a high-speed DB2 data migration solution. When you specify this option, UNLOAD PLUS performs the following functions:

- Unloads data in an internal format that only LOADPLUS can read
- Creates verification records to ensure that the data format matches the table definition
- Generates LOADPLUS control cards that you can use to reload the data

**Unloading data with FORMAT BMCLOAD**

This topic describes how to use FORMAT BMCLOAD for high-speed data migration using UNLOAD PLUS and LOADPLUS.

For an example of this feature, see “Example 19 - Unloading data to LOADPLUS” on page 389.
To unload data that you plan to reload by using LOADPLUS

1. Ensure that the table that you are unloading meets the requirements specified in “Table structure considerations for FORMAT BMCLOAD” on page 68.

2. Specify FORMAT BMCLOAD with your UNLOAD command.

3. Specify CNTLCARDS BMCLOAD with your UNLOAD command.

   Note
   If you specify or default to any other CNTLCARDS option, UNLOAD PLUS changes the value to BMCLOAD.

4. If you are unloading an identity column that is defined as GENERATED ALWAYS, include the string 'IDENTITYOVERRIDE YES' on your CNTLCARDS BMCLOAD option.

   Additionally, you must edit your generated LOAD control cards to add a field specification for the identity column.

5. If you are unloading a system-period temporal table, include the string 'PERIODOVERRIDE TRANSIDOVERRIDE' on your INTO option.

6. (recommended) Dynamically allocate your SYSREC data set.

7. Ensure that your UNLOAD command does not contain any of the invalid or restricted commands described in “Command options that are incompatible with FORMAT BMCLOAD” on page 70.

8. Ensure that your job meets additional requirements described in “Additional restrictions and considerations for FORMAT BMCLOAD” on page 70.

9. After running your unload job, edit generated LOAD control cards as needed.

   For example, if you are unloading columns that are defined as GENERATED ALWAYS, you must add field specifications for those columns.

Table structure considerations for FORMAT BMCLOAD

In general, when you use the FORMAT BMCLOAD option, the structure of the table that you are unloading must be nearly identical to the structure of the table that you plan to load with LOADPLUS. This topic describes considerations regarding the structure of the two tables.
**General structure considerations**

The following characteristics of the table structure do not have to be the same on the target as they are on the source:

- Type of table space (for example, segmented)
- Indexes
- Index and table space versions

The following specific characteristics must match:

- Coded character set identifier (CCSID)
- Row format (basic or reordered)

**Column definitions**

The following considerations apply to column definitions in the two tables:

- The number, order, and data type of the columns in the two tables must be the same, but column names may be different.

- For VARCHAR, VARGRAPHIC, and VARBINARY columns, target table columns can have a length that is greater than the source table columns.

- Row ID columns that are defined in the source table must be defined in the target table as GENERATED BY DEFAULT.

  When a table contains a row ID column that is defined as GENERATED ALWAYS, UNLOAD PLUS unloads the data. However, LOADPLUS fails to load the data if the column is not defined in the target table as GENERATED BY DEFAULT.

**DB2 user exits**

The following considerations apply to user exits that are defined for the two tables:

- If the source has an EDITPROC, the target must have an EDITPROC with the same name.

- If the source has a VALIDPROC or FIELDPROC, the target must have one, but the names can differ. LOADPLUS displays a warning but loads the rows.
Command options that are incompatible with FORMAT BMCLOAD

The following table describes the options that are not valid when you specify FORMAT BMCLOAD, and how UNLOAD PLUS responds if your job contains these options:

Table 12: Options that are incompatible with FORMAT BMCLOAD

<table>
<thead>
<tr>
<th>Command option</th>
<th>Response from UNLOAD PLUS if you include the option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>AUTOTAG YES</td>
<td>Terminates</td>
</tr>
<tr>
<td>CCSID</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>DATEFMT</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>DIRECT NO</td>
<td>Terminates</td>
</tr>
<tr>
<td>DIRECT AUTO</td>
<td>Terminates</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>FIXEDVARCHAR YES</td>
<td>Terminates</td>
</tr>
<tr>
<td>MAXBLKSIZE</td>
<td>Terminates</td>
</tr>
<tr>
<td>NOSUBS</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>NULLCHAR</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>NULLTYPE</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>TIMEFMT</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>TSFMT</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>UNLOADEXIT</td>
<td>Ignores this option</td>
</tr>
<tr>
<td>USELRECL YES</td>
<td>Ignores this option</td>
</tr>
</tbody>
</table>

Additional restrictions and considerations for FORMAT BMCLOAD

The following additional restrictions and considerations apply when you specify FORMAT BMCLOAD:

- The SELECT statement is limited as follows:
  - The SELECT statement cannot contain any column information. UNLOAD PLUS supports SELECT * only.
  - The SELECT INTO statement cannot contain any field information.
The record options block cannot contain RECORDID.

- If you unload tables that contain identity columns and you specify SHRLEVEL CHANGE CONSISTENT NO, the MAXASSIGNEDVAL value supplied in internal verification records to LOADPLUS might be incorrect because the unloaded table is available for updates.

- You cannot use FORMAT BMCLOAD when you are unloading LOB or XML data. UNLOAD PLUS terminates in this case.

- You cannot load output from FORMAT BMCLOAD by using a version of LOADPLUS that is earlier than the version of UNLOAD PLUS that you are running. For example, if you unload your data by using FORMAT BMCLOAD in version 10.2 of UNLOAD PLUS, you cannot load it by using FORMAT BMCUNLOAD in version 10.1 of LOADPLUS.

- You cannot use user exits other than those described in “DB2 user exits” on page 69.

- Message BMC51674I indicates the total number of records that UNLOAD PLUS wrote to the output file, including the control records. An additional message, BMC51679I, indicates the number of data records and the number of control records that UNLOAD PLUS wrote.

**DSNTIAUL-formatted output (FORMAT DSNTIAUL)**

To produce an output file in which the unloaded data matches the format of the IBM DSNTIAUL program, specify FORMAT DSNTIAUL.

Unlike the DSNTIAUL program, UNLOAD PLUS (using DIRECT YES) supports ASCII and Unicode output.

*Note*

FORMAT STANDARD does not produce output that matches that of DSNTIAUL.

When you specify FORMAT DSNTIAUL, UNLOAD PLUS performs the following functions:

- Provides data for all data types in internal format except for the date, time, and timestamp data types, which it provides in external format

- Pads all variable-length fields (VARCHAR, VARGRAPHIC, and VARBINARY) with X'00' to their maximum length

- Creates fixed-length output records with a maximum length of 32760 bytes
**Note**

If UNLOAD PLUS processes a LONG VARCHAR or LONG VARGRAPHIC column type, the maximum formatted row length may exceed the allowable output record length of 32760. You can use the INTO option with a field specification to reduce the length of the LONG column and avoid this error. To reduce the number of records that UNLOAD PLUS discards when they exceed the shortened length, specify the TRIM or TRUNCATE function to remove trailing spaces, or truncate the field to fit.

- Treats a string constant as variable length
- Puts a one-byte null indicator field after all nullable fields
  The null indicator field contains either X'6F' if the field is null or X'00' if it is not null.

**Restrictions**

Note the following restrictions when you specify FORMAT DSNTIAUL:

- You cannot use this option when unloading LOB or XML data. UNLOAD PLUS terminates in this case.
- When you specify FORMAT DSNTIAUL, UNLOAD PLUS sets the following options to the values shown:
  — FIXEDVARCHAR to YES
  — NULLCHAR to ‘?’
  — NULLTYPE to T1
  — CONSTRULES to STANDARD
- When you specify FORMAT DSNTIAUL, UNLOAD PLUS ignores the TRIM option.

**CSV output (FORMAT CSV)**

To produce an output file in which the unloaded data is in comma-separated-value (CSV) format, specify FORMAT CSV.

Optionally, you can specify additional keywords. CSV options are TERMINATEDBY, ENCLOSEDBY, AND, NULLSTRING, and RTRIM. When you use more than one option, the syntax must follow the sequence in the CSV syntax
For example, if you use TERMINATEDBY, that option must appear before ENCLOSEDBY or NULLSTRING.

For an example of this feature, see “Example 12 - Unloading to CSV format” on page 357.

When you specify FORMAT CSV, UNLOAD PLUS performs the following functions:

- Uses the character specified by the TERMINATEDBY option to separate each field from other fields
- Uses the character specified by the ENCLOSEDBY option and the character specified by the AND option to enclose each nonnumeric field with a pair of characters
- Produces all data in external format (even if you specify an explicit data type)
- Creates all fields as variable length except for those fields for which you specify an explicit data type and length

**Restrictions on FORMAT CSV**

The following restrictions apply when you specify FORMAT CSV:

- If you specify FORMAT CSV when you are unloading LOB or XML data, UNLOAD PLUS terminates.
- If you specify USELRECL YES with FORMAT CSV, UNLOAD PLUS ignores USELRECL YES.
- UNLOAD PLUS does not support conversions to VARCHAR, VARGRAPHIC, BINARY, VARBINARY, and ROWID data types when you specify FORMAT CSV.

**TERMINATEDBY, ENCLOSEDBY, and AND options**

This section describes guidelines for the TERMINATEDBY, ENCLOSEDBY, and AND options when you specify FORMAT CSV.

UNLOAD PLUS assumes that the values for the TERMINATEDBY, ENCLOSEDBY, and AND options are in EBCDIC format and translates them, if necessary, to the CCSID of the output. To ensure that applications can read your CSV output, note the following guidelines:

- Ensure that the values that you specify (or default to) for these options are not contained in your column data, or that your column data will not translate to these values. Otherwise, applications might read the resulting output incorrectly.
For these options, avoid specifying values that would expand to multiple-byte characters when the values are translated.

Avoid specifying any of the following characters:

— X'0A' (line feed in ASCII and Unicode)
— X'0D' (carriage return in ASCII, Unicode, and EBCDIC)
— X'15' (new line in EBCDIC)
— X'25' (line feed in EBCDIC)

**LOADPLUS interaction with FORMAT CSV**

Although supported, using the FORMAT CSV option of UNLOAD PLUS and LOADPLUS together is not the most efficient method for using these two products to migrate DB2 data.

Instead of FORMAT CSV, use the following guidelines for migrating DB2 data with UNLOAD PLUS and LOADPLUS:

- If your table structures are almost identical, BMC recommends that you specify FORMAT BMCLOAD (in UNLOAD PLUS) and FORMAT BMCUNLOAD (in LOADPLUS) to migrate DB2 data.

- If your table structures are not similar enough to use this method, BMC recommends that you use the FORMAT INTERNAL option.

If you must reload CSV data by using LOADPLUS, BMC recommends that you specify CNTLCARDS BMCLOAD in your UNLOAD PLUS job. Specifying this option tells UNLOAD PLUS to generate control cards for LOADPLUS.

**XML output (FORMAT XML)**

To produce an output file in which the unloaded data is formatted with XML tags, specify FORMAT XML.

Optionally, you can specify additional keywords. XML options are SELECT_ELEMENT, ENCLOSEDBY, AND, and NULLSTRING. When you use more than one option, the syntax must follow the sequence in the XML syntax diagram. For example, if you specify SELECT_ELEMENT, that option must appear before ENCLOSEDBY or NULLSTRING.

When you specify FORMAT XML, UNLOAD PLUS performs the following functions:
- Encloses each field in a pair of XML tags as specified by the ENCLOSEDBY and AND options

- Optionally encloses each record in a pair of XML tags as specified by the SELECT_ELEMENT option

UNLOAD PLUS does not validate the XML tags to ensure that they conform to any XML standard.

**Option restrictions**

UNLOAD PLUS ignores the following options when you specify FORMAT XML:

- CNTLCARDS
- AUTOTAG
- RECORDID
- NULLTYPE
- CONSTRULES
- USELRECL YES

The INTO option is not valid when you specify FORMAT XML. UNLOAD PLUS issues an error message and terminates.

**Additional considerations**

UNLOAD PLUS does not support FORMAT XML when you are unloading Unicode, LOB, or XML data.

**Importing XML output to other applications**

Before you import the XML output into another application, you might need to add the following items to the XML output. See your XML parser’s documentation for additional information.

- An XML declaration
  Each XML parser supports a different set of encoding.

- A set of tags for the root element
  Some XML parsers require that you include a root element as the first element in your XML document. The root element contains the start tag at the beginning of your document and an end tag at the end of your document.
The following example illustrates the output that UNLOAD PLUS produces when you specify FORMAT XML. The declaration statement and root element tags were added outside of UNLOAD PLUS.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<report>
  <C2269T10>COL01T10="ADUB38DB" COL02T10="ADUB39DB" COL03T10="ADUB35TS" COL04T10="ADUB37T" COL05T10="ADUB37T"
  <C2269T10>COL01T10="ADUB32TB" COL02T10="ADUB32DB" COL03T10="ADUB32TS" COL04T10="ADUB37T" COL05T10="ADUB37T"
</report>
```

### Examples of FORMAT option output

The tables in this section illustrate the results of using each FORMAT option.

Table 13 on page 76 lists different data types and sample data. Table 14 on page 77 shows the data that would be unloaded, based on the table definitions in Table 13 on page 76. The CSV format uses the default TERMINATEDBY and ENCLOSEDBY characters.

#### Table 13: Table definition for FORMAT example

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column definition</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR_COL</td>
<td>CHAR (10)</td>
<td>NOT NULL WITH DEFAULT  &quot;CHARACTERS&quot;</td>
</tr>
<tr>
<td>DEC_COL</td>
<td>DECIMAL(7,2)</td>
<td>NOT NULL WITH DEFAULT 142.23</td>
</tr>
<tr>
<td>INTEGER_COL</td>
<td>INTEGER</td>
<td>NOT NULL WITH DEFAULT 12996</td>
</tr>
<tr>
<td>DATE_COL</td>
<td>DATE</td>
<td>NOT NULL WITH DEFAULT  &quot;08/22/1994&quot;</td>
</tr>
<tr>
<td>TIME_COL</td>
<td>TIME</td>
<td>NOT NULL WITH DEFAULT  &quot;08:41 AM&quot;</td>
</tr>
</tbody>
</table>

The following table uses the data from the preceding table for the example formats.
Table 14: Examples of unloaded data when using the FORMAT option

<table>
<thead>
<tr>
<th>FORMAT option</th>
<th>Form</th>
<th>Example of unloaded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD</td>
<td>EBCDIC</td>
<td>CHARACTERS 08/22/199408:41 AM</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>CCCDCCECDE0123003CFF6FFFFFF7FF4CD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3819133592042C0024081221199408A41014</td>
</tr>
<tr>
<td>EXTERNAL</td>
<td>EBCDIC</td>
<td>CHARACTERS 142.23 1299608/22/199408:41 AM</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>CCCDCCECDE444FFF4FF444444F6FF6FFF4FFF4FFF4FF4CD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>381913359200014282300000012996081221199408A41014</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>EBCDIC</td>
<td>CHARACTERS D m</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>CCCDCCECDE0123003C1902042</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3819133592042C00249482813</td>
</tr>
<tr>
<td>BMCLOAD</td>
<td></td>
<td>Format is proprietary to BMC Software.</td>
</tr>
<tr>
<td>DSNTIAUL</td>
<td>EBCDIC</td>
<td>CHARACTERS 08/22/199408:41 AM</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>CCCDCCECDE0123003CFF6FFFFFF7FF4CD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3819133592042C0024081221199408A41014</td>
</tr>
<tr>
<td>CSV</td>
<td>EBCDIC</td>
<td>&quot;CHARACTERS&quot;,142.23,12996,&quot;08/22/1994&quot;,&quot;08:41 AM&quot;</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>7CCDCCECDE76FF4FF6FFF67FF66FF66FFFFFF7FF4CD7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3819133592FB142B23B12996BF0812211994FBF08A41014F</td>
</tr>
<tr>
<td>XML</td>
<td>EBCDIC</td>
<td>&lt;TABLE_NAME&gt;&lt;CHAR_COL&gt;&quot;CHARACTERS&quot;&lt;/CHAR_COL&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;DEC_COL&gt;142.23&lt;/DEC_COL&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;INTEGER_COL&gt;12996&lt;/INTEGER_COL&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;DATE_COL&gt;&quot;08/22/1994&quot;&lt;/DATE_COL&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;TIME_COL&gt;&quot;08:41 AM&quot;&lt;/TIME_COL&gt;&lt;/TABLE_NAME&gt;</td>
</tr>
</tbody>
</table>

DB2 user exits with UNLOAD PLUS

UNLOAD PLUS can invoke the following types of DB2 user exits during processing:

- EDITPROCs
- FIELDPROCs
- Date exits
  - DSNXVDTX (EBCDIC date exit routine)
  - DSNXVDTA (ASCII date exit routine)
  - DSNXVDTU (Unicode date exit routine)
Time exits

- DSNXVTMX (EBCDIC time exit routine)
- DSNXVTMA (ASCII time exit routine)
- DSNXVTMU (Unicode time exit routine)

Authorization exit

By default, UNLOAD PLUS invokes these exits in supervisor state (and PSW key=7). If you can ensure that all EDITPROCs called by UNLOAD PLUS are able to run in problem state, you can specify UXSTATE=PROB in the UNLOAD PLUS installation options module.

UNLOAD PLUS invokes DB2 user exits only when DIRECT YES is in effect.

### DB2 features that UNLOAD PLUS does not support

This version of UNLOAD PLUS does not support the following features of DB2:

- Columns that contain a CCSID specification
- Encrypted data (except encrypted copies made by COPY PLUS)
- Unicode object names
- Striped data sets for DB2 VSAM objects
- XML schema validation based on the schema specified in the XML type modifier
Syntax of the UNLOAD command

This chapter describes the command options that you can use with the UNLOAD command.

Command syntax rules for UNLOAD PLUS

The following general rules apply to the UNLOAD PLUS command syntax:

- In an UNLOAD command, UNLOAD PLUS considers any line beginning with an asterisk (*) in column 1 to be a comment and ignores it.

- When encountering two consecutive hyphens in a line (except within a delimited token), UNLOAD PLUS considers the remainder of the line to be a comment, which UNLOAD PLUS ignores.

- When you use a signed token, do not place a space between the sign and the value.

- You can split a token (such as a keyword, identifier, or constant) across a line. However, UNLOAD PLUS ignores anything in columns 73 through 80. A blank, or a delimiter if the token is delimited, indicates the end of the token. If an undelimited token ends in column 72, column 1 on the next line must be blank.

  **Note**
  If you use applications that automate JCL submission and resolve symbolic variables within your JCL, the resulting control cards might not appear as they do in the JCL that you created. These applications might produce an invalid command statement.

- If you specify the same command option more than once, UNLOAD PLUS uses only the last option that you specify. For example, if you specify the following options, UNLOAD PLUS accepts FORMAT INTERNAL as the processing option:

  ```
  UNLOAD
  FORMAT EXTERNAL
  FORMAT INTERNAL
  ```

  The following exceptions apply:
— You can specify UNLOAD TABLESPACE only once.

— You can specify the DIRECT option only once.

- The SELECT statement must be the last set of options in your SYSIN command stream.

- In the syntax diagrams in the following pages, underlined options indicate default options.

## Specifying object names in your UNLOAD PLUS syntax

Note the following information about specifying object names in your UNLOAD PLUS syntax:

- You can use an alias or synonym wherever the syntax diagram shows a table name or view name.

- If you use SQL reserved words or UNLOAD PLUS syntax keywords as object names, BMC recommends that you delimit the reserved words or keywords to prevent syntax errors.

- UNLOAD PLUS does not support Unicode table or view names.

- UNLOAD PLUS does not support delimited object names that do not have a character representation in EBCDIC.

- UNLOAD PLUS supports the use of the double-byte character set (DBCS). You can use DBCS characters in DB2 identifiers such as tables, columns, or view names.

## Alphabetical listing of UNLOAD PLUS options

The following table lists the UNLOAD PLUS command options alphabetically and indicates where to find each option description. The final two columns in the table indicate whether the option is supported with DIRECT NO or DIRECT YES.

### Table 15: UNLOAD PLUS command options

<table>
<thead>
<tr>
<th>Command option</th>
<th>Reference</th>
<th>DIRECT NO</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>“ACTIVE” on page 122</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Command option</td>
<td>Reference</td>
<td>DIRECT NO</td>
<td>DIRECT YES</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>ANALYZE</td>
<td>“ANALYZE” on page 123</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>ASCII</td>
<td>“ASCII” on page 158</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AUTOTAG</td>
<td>“AUTOTAG” on page 145</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CCSID</td>
<td>“CCSID” on page 159</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CENTURY</td>
<td>“CENTURY” on page 175</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CLONE</td>
<td>“CLONE” on page 98</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CNTLCARDS</td>
<td>“CNTLCARDS” on page 130</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CNTLDDN</td>
<td>“CNTLDDN” on page 139</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CURRENTDEGREE</td>
<td>“CURRENTDEGREE” on page 102</td>
<td>Yes</td>
<td>Ignored</td>
</tr>
<tr>
<td>DATACLAS</td>
<td>“DATACLAS” on page 195</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DATEFMT</td>
<td>“DATEFMT” on page 160</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DDLDDN</td>
<td>“DDLDDN” on page 121</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DECFLOAT_ROUNDMODE</td>
<td>“DECFLOAT_ROUNDMODE” on page 165</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DELETEFILES</td>
<td>“DELETEFILES” on page 173</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DIR</td>
<td>“DIR” on page 190</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DIRECT</td>
<td>“DIRECT” on page 98</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DISCARDS</td>
<td>“DISCARDS” on page 145</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DISKEXPD</td>
<td>“DISKEXPD” on page 203</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DISKRETN</td>
<td>“DISKRETN” on page 202</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DRAIN_WAIT</td>
<td>“DRAIN_WAIT” on page 175</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>DSNAME</td>
<td>“DSNAME” on page 184</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DSNSTYPE</td>
<td>“DSNSTYPE” on page 189</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DSPLOCKS</td>
<td>“DSPLOCKS” on page 178</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>“EBCDIC” on page 158</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ENUMROWS</td>
<td>“ENUMROWS” on page 141</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>ESTROWS</td>
<td>“ESTROWS” on page 221</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EXPDT</td>
<td>“EXPDT” on page 205</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FILESZPCT</td>
<td>“FILESZPCT” on page 200</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FILL</td>
<td>“FILL” on page 157</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FILTERPART</td>
<td>“FILTERPART” on page 109</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>Command option</td>
<td>Reference</td>
<td>DIRECT NO</td>
<td>DIRECT YES</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>FIXEDVARCHAR</td>
<td>“FIXEDVARCHAR” on page 167</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FORCE</td>
<td>“FORCE” on page 179</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>FORCE_AT</td>
<td>“FORCE” on page 179</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>FORCE_DELAY</td>
<td>“FORCE” on page 179</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>FORCE_RPT</td>
<td>“FORCE” on page 179</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>FORMAT</td>
<td>“FORMAT” on page 149</td>
<td>Yes, except FORMAT BMCLOAD</td>
<td>Yes</td>
</tr>
<tr>
<td>GDGLIMIT</td>
<td>“GDGLIMIT” on page 193</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IMPLICIT_TZ</td>
<td>“IMPLICIT_TZ” on page 164</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>INFILE</td>
<td>“INFILE” on page 110</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>“INTERVAL” on page 144</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INTO</td>
<td>“INTO” on page 208</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LIMIT</td>
<td>“LIMIT” on page 144</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOGICAL_PART</td>
<td>“PART or LOGICAL PART” on page 119</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>LRECL</td>
<td>“LRECL” on page 192</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MAXBLKSIZE</td>
<td>“MAXBLKSIZE” on page 127</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MAXCONNECT</td>
<td>“MAXCONNECT” on page 148</td>
<td>Yes</td>
<td>Ignored</td>
</tr>
<tr>
<td>MAXPRIM</td>
<td>“MAXPRIM” on page 198</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MAXSECD</td>
<td>“MAXSECD” on page 199</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MAXSORTS</td>
<td>“MAXSORTS” on page 147</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>“MGMTCLAS” on page 195</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NAME</td>
<td>“INTO” on page 208</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NBRSECD</td>
<td>“NBRSECD” on page 200</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NOSUBS</td>
<td>“NOSUBS” on page 160</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>NULLCHAR</td>
<td>“NULLCHAR” on page 155</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NULLTYPE</td>
<td>“NULLTYPE” on page 156</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OBID</td>
<td>“FROM” on page 211</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>ON FAILURE</td>
<td>“ON FAILURE” on page 171</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ON MESSAGE</td>
<td>“ON MESSAGE” on page 170</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>Command option</td>
<td>Reference</td>
<td>DIRECT NO</td>
<td>DIRECT YES</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>“OPTIONS” on page 221</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ORDER</td>
<td>“ORDER” on page 139</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>“ORDER BY” on page 219</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>“OUTPUT” on page 182</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PART</td>
<td>“PART or LOGICAL PART” on page 119</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>PCTPRIM</td>
<td>“PCTPRIM” on page 197</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECFM</td>
<td>“RECFM” on page 128</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECORDID</td>
<td>“INTO” on page 208</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RETPD</td>
<td>“RETPD” on page 204</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RETRY</td>
<td>“RETRY” on page 177</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>RETRY_DELAY</td>
<td>“RETRY_DELAY” on page 178</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>SELECT</td>
<td>“SELECT” on page 205</td>
<td>Yes, with documented exceptions</td>
<td>Yes</td>
</tr>
<tr>
<td>SET CURRENT QUERY ACCELERATION ALL</td>
<td>“SET CURRENT QUERY ACCELERATION ALL” on page 182</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>“SHRLEVEL” on page 103</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>SORTDEVT</td>
<td>“SORTDEVT” on page 142</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>SORTNUM</td>
<td>“SORTNUM” on page 142</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>SPACE</td>
<td>“SPACE” on page 197</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SPANNED</td>
<td>“SPANNED” on page 129</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>STORCLAS</td>
<td>“STORCLAS” on page 194</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSETS</td>
<td>“SUBSETS” on page 191</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>SYNC</td>
<td>“SYNC” on page 148</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>“UNLOAD TABLESPACE” on page 97</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>TIMEFMT</td>
<td>“TIMEFMT” on page 161</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRTCH</td>
<td>“TRTCH” on page 203</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TSFMT</td>
<td>“TSFMT” on page 162</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNICODE</td>
<td>“UNICODE” on page 158</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNIT</td>
<td>“UNIT” on page 183</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNITCNT</td>
<td>“UNITCNT” on page 196</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## UNLOAD PLUS command syntax diagrams

The figures in this topic illustrate the syntax of the UNLOAD command.

<table>
<thead>
<tr>
<th>Command option</th>
<th>Reference</th>
<th>DIRECT NO</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>“UNLOAD” on page 97</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOADADDN</td>
<td>“UNLOADADDN” on page 121</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOADEXIT</td>
<td>“UNLOADEXIT” on page 169</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>USELRECL</td>
<td>“USELRECL” on page 126</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VOLCNT</td>
<td>“VOLCNT” on page 193</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VOLUMES</td>
<td>“VOLUMES” on page 201</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WATOMSG</td>
<td>“WATOMSG” on page 108</td>
<td>Ignored</td>
<td>Yes</td>
</tr>
<tr>
<td>XBMID</td>
<td>“XBMID” on page 108</td>
<td>Yes for zIIP processing Ignored for snapshot processing</td>
<td>Yes</td>
</tr>
<tr>
<td>ZIIP</td>
<td>“ZIIP” on page 106</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ZONEDDECOVP</td>
<td>“ZONEDDECOVP” on page 165</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Figure 4: UNLOAD PLUS command syntax diagrams (part 1 of 12)

- UNLOAD
- UNLOAD TABLESPACE `databaseName` tableSpaceName
  - DIRECT
    - YES
    - NO
      - ROWSETZ
    - AUTO
  - CURRENTDEGREE
    - NONE
    - ANY
    - 1
  - SHRLEVEL
    - REFERENCE
    - CHANGE
      - CONSISTENT
        - NO
        - QUIESCE
          - YES
          - NO
  - ZIIP
    - ENABLED
    - DISABLED
  - XBMID `ssid`
  - WTOMSG SUFFIX `text`
  - FILTERPART
    - NO
    - YES

- INFILE
  - IMAGECOPY
    - FULL
    - INCREMENTAL
      - FULL
      - INLINE
        - YES
        - NO
  - SNAPCOPY
    - INCREMENTAL
      - integer
  - VSAMDD
  - VSAMDDPREFIX `prefix`

- PART `partitionNumber`

* Option has a corresponding installation option
Figure 5: UNLOAD PLUS command syntax diagrams (part 2 of 12)

- UNLOADDN\(^a\) - (SYSREC
  \(\text{ddname1}\)
  \text{outputDescriptor}
  \text{SYSRED}
  \(\text{ddname2}\)
  \text{outputDescriptor}
  ACTIVE\(^a\) - (YES
  NO
  NO
  YES
)

- ANALYZE\(^a\) DB2STATS
  AGE
  NOLIMIT
  integer
  HURBA
  CMRATIO\(^a\)
  50
  integer
  USELRECL\(^a\)
  NO
  YES

- MAXBLKSIZE integer
  RECFM\(^a\)
  AUTO
  VB
  SPANNED
  NO
  YES

- CNTLCARDS control card formats block
  CNTLDDN
  SYSCNTL
  ddname

- ORDER
  NO
  YES
  ENUMROWS
  0
  integer
  SORTDEVT devioType

- SORTNUM\(^a\)
  32
  integer
  LIMIT
  0
  integer
  INTERVAL
  0
  integer

- DISCARDS
  0
  integer
  AUTOTAG
  NO
  YES
  MAXSORTS\(^a\)
  16
  integer
  MAXCONNECT\(^a\)
  16
  integer

\(^a\) Option has a corresponding installation option
Figure 6: UNLOAD PLUS command syntax diagrams (part 3 of 12)

SYNC integer

FORMAT

STANDARD

EXTERNAL

INTERNAL

BMCLOAD

DSNTIAUL

CSV

XML

CSV and XML format block

See CSV and XML format block details

NULLCHAR\(^a\)

? 

NULLTYPE\(^a\)

T1

T2

FILL\(^a\)

NO

YES

EBCDIC

ASCII

UNICODE

CCSID (sbcS, mixed, dbcs)

NOSUBS

DATEFMT (’string’)

TIMEFMT (’string’)

TSFMT (’string’)

IMPLICIT_TZ ’timeZoneString’

ZONEDECIMAL\(^a\)

C

A

B

D

E

F

\(^a\) Option has a corresponding installation option
Figure 7: UNLOAD PLUS command syntax diagrams (part 4 of 12)
Figure 8: UNLOAD PLUS command syntax diagrams (part 5 of 12)

DELETENES\textsuperscript{a} \quad YES \quad NO

CENTURY\textsuperscript{a} (ccy,ccyy)

DRAIN\_WAIT\textsuperscript{a} \quad NONE \quad UTIL \quad SQL \quad integer

RETRY\textsuperscript{a} \quad 255 \quad integer

RETRY\_DELAY\textsuperscript{a} \quad 1 \quad integer

DSPLOCKS\textsuperscript{a} \quad DRNFAIL \quad NONE \quad RETRY

FORCE\textsuperscript{a} \quad NONE \quad ALL \quad REPORTONLY

FORCE\_AT\textsuperscript{a} \quad START \quad RETRY \quad LASTRETRY

FORCE\_DELAY\textsuperscript{a} \quad 0 \quad integer

FORCE\_RPT\textsuperscript{a} \quad NO \quad YES

SET CURRENT QUERY ACCELERATION ALL

OUTPUT block

SELECT block

\textsuperscript{a} Option has a corresponding installation option

See OUTPUT block detail

See SELECT block detail
Figure 9: UNLOAD PLUS command syntax diagrams (part 6 of 12)

**OUTPUT block – dynamic allocation options**

- **OUTPUT descriptorName**
  - Common options block
  - Disk options block
  - Tape options block

**Common options**

- **UNIT\(^a\)**
  - SYSALLDA
  - $NO\_UNIT\$ name
- **DSNAME\(^b\)**
  - 'USERID.&JOBNAME.&TS.&TYPE.&SELNUM'
  - 'patem'
  - 'dataSetName'
- **DSNTPBE**
  - POS
    - LIBRARY
    - HFS
    - NONE
    - LARGE
    - BASIC
    - EXTREQ
    - EXTPREF
- **DIR integer**
- **SUBSETS 1 integer**
- **LRECL integer**
- **VOLCNT\(^a\)**
  - 25 integer
- **GDGLIMIT\(^a\)**
  - 5 integer
- **STORCLAS\(^a\)**
  - class NONE
- **DATACLAS\(^a\)**
  - class NONE
- **MGMTCLAS\(^a\)**
  - class NONE
- **UNITCNT\(^a\)**
  - 0 integer

\(^a\) Option has a corresponding installation option
Figure 10: UNLOAD PLUS command syntax diagrams (part 7 of 12)

Disk options

- `SPACE^a` (primary,secondary)
- `CYL`
- `TRK`
- `PCTPRIM^a` integer
- `AUTO` integer
- `MAXPRIM^a` integer
- `MAXSECD^a` integer
- `NBRSECD^a` integer
- `FILESZPCT^a` integer
- `VOLUMES^a` volume
- `DISKRETN^a` integer
- `DISKEXP^a` date

Tape options

- `TRTC^a` NONE
- `TRTC^a` COMP
- `TRTC^a` NOCOMP
- `RETPD^a` integer
- `EXPDT^a` date

^a Option has a corresponding installation option
Figure 11: UNLOAD PLUS command syntax diagrams (part 8 of 12)

Control card formats block detail

- CNTLCARDS
- DBLOAD
- DB2
- BMCLOAD
- DB2DDL
- EASYTRIEVE
- FOCUS
- NOMAD
- SAS
- SQL/DS
- SQL/DS-DDL
- SQL/DS-LOAD
- TERADATA
- TERADATA-BULK
- TERADATA-FAST
- TERADATA-MULT

CSV and XML format block detail

- TERMINATEDBY
  - 'char'
  - X'hexValue'
- SELECT_ELEMENT
- AUTO
  - NONE
  - 'string'
- ENCLOSEDBY
  - 'char'
  - X'hexValue'
- NULLSTRING
  - 'NULL'
  - 'string'
- RTRIM

*a Option is valid for CSV format only
*b Option is valid for XML format only
Figure 12: UNLOAD PLUS command syntax diagrams (part 9 of 12)
Figure 13: UNLOAD PLUS command syntax diagrams (part 10 of 12)

Field specification block detail

```
fieldName

CHAR

VARCHAR

BLOBF

outputDescriptorName

CLOBF

DBCLOBF

GRAPHIC

EXTERNAL

 Barang

VARGRAPHIC (length)

SMALLINT

INTEGER

BIGIN

decimal specification block

FLOAT

EXTERNAL

ROUND

BINARY

VARBINARY

EXTERNAL

VARYING

DATE

TIME

TIMESTAMP

EXTERNAL

BLOB

CLOB

DBCLOB

DECIMAL

XML

EXIT

programName

PARM (constant)

TRIM and TRUNCATE are not valid for BLOBF, CLOBF, or DBCLOBF

* Option has a corresponding installation option
```
Figure 14: UNLOAD PLUS command syntax diagrams (part 11 of 12)

Field specification block detail
(continued)

IF
  NULL
  ERROR
  predicate block
  VALUE
  (constant
    NULL
    CURRENT DATE
    CURRENT TIME
    CURRENT TIMESTAMP)

See predicate block detail

Decimal specification block detail

DECIMAL
  (precision, scale)
  PACKED
  ZONED
  (precision, scale)
  EXTERNAL
    (length, scale)
    ROUND
    FILL a
      NO
      YES

Condition block detail

AND
  OR
  NOT
  predicate block
  (condition block)

See predicate block detail

a Option has a corresponding installation option
UNLOAD PLUS options

The rest of this chapter contains descriptions of each UNLOAD PLUS option. The options are in order of their appearance in the syntax diagrams.
Basic processing options

The basic unload processing options control most aspects of UNLOAD PLUS execution.

UNLOAD

Use the UNLOAD command to unload data from one or more tables in a table space. In certain circumstances, you can use UNLOAD TABLESPACE instead to unload all tables in a table space.

UNLOAD TABLESPACE

You can use the UNLOAD TABLESPACE command to unload all data from the designated table space.

This option is valid only when DIRECT YES is in effect.

UNLOAD PLUS generates a SELECT statement for each table in the table space. When you specify UNLOAD TABLESPACE, you obtain the same results as if you had specified SELECT * FROM for every table in the table space. Therefore, the rules for assigning the SYSREC and SYSCNTL output data sets are the same as if you had specified SELECT * FROM for every table in the table space.

This option can help you avoid maintenance of a large list of SELECT statements. When new tables are added to the table space or deleted from it, you need not change the syntax to unload the data.

If you specify MSGLEVEL(1) on your EXEC statement or in your installation options, UNLOAD PLUS prints a cross-reference of generated SELECT statements with corresponding OBIDs. For an example of the statements that are automatically generated, see “Example 18 - Unloading a table space and unloading data as Unicode” on page 381.

Restrictions
The following restrictions apply to UNLOAD TABLESPACE:

- If you specify any of the following options with the UNLOAD TABLESPACE command, UNLOAD PLUS issues an error message and terminates:
  - DIRECT NO, or DIRECT AUTO that results in DIRECT NO processing
  - Any SELECT statement clause
  - FORMAT XML when also specifying the UNICODE option

- If you specify UNLOAD TABLESPACE when unloading a LOB or XML column, UNLOAD PLUS terminates.

- If you specify UNLOAD TABLESPACE more than once in your JCL, UNLOAD PLUS terminates.

Additional considerations

The following considerations apply to the UNLOAD TABLESPACE command:

- The SELECT statements are generated in OBID order.

- If you specify INFILE *ddname* using a DDLIN data set to unload an image copy, each CREATE TABLE statement must specify the OBID of the table.
  If you are unloading a single-table table space, you do not need to specify the OBID.

- If you do not specify the name of the database, UNLOAD PLUS uses the default database name DSNDB04.

- When combining data from multiple tables into a single data set, consider using AUTOTAG YES. For details, see “AUTOTAG” on page 145.

CLONE

The CLONE option indicates that you want to unload only the clone table in the specified table space. (You cannot unload the clone table and base table in the same unload job.)

If you specify this option but no clone table exists in the specified table space, UNLOAD PLUS terminates.

DIRECT

The DIRECT option allows you to specify how UNLOAD PLUS processes table data.
Requirements and restrictions

The following requirements and restrictions apply to the DIRECT option:

- In your SYSIN command stream, place any DIRECT option specification immediately after your UNLOAD or UNLOAD TABLESPACE option.

- If you specify the DIRECT option more than once in your JCL, UNLOAD PLUS terminates.

- You must specify DIRECT NO when either of the following conditions exists:
  
  — You are unloading catalog data.

  For more information, see “Considerations for unloading DB2 catalog data” on page 63

  — You are unloading LOB or XML data to standard unload data sets (SYSREC and SYSRED)

  **Note**

  You can specify DIRECT YES to unload LOB or XML data to VBS unload data sets. You can specify either DIRECT YES or DIRECT NO to unload LOB or XML data to files that are referenced in your standard unload data sets. For more information, see “Unloading LOB and XML data” on page 54

**YES**

If you specify DIRECT YES, UNLOAD PLUS uses its SELECT-like syntax to read table data directly from the table space data set. If the SELECT statement contains syntax that is not within the scope of the UNLOAD PLUS syntax, the utility terminates. This option provides high-performance unloads of DB2 table data, but might limit the functionality that the SELECT statement provides.

UNLOAD PLUS ignores the CURRENTDEGREE and MAXCONNECT options when DIRECT YES is in effect. Additionally, UNLOAD PLUS terminates when you specify DIRECT YES for any of the structures and data types that require DIRECT NO.

For more information about DIRECT YES mode, see “DIRECT YES” on page 28.
NO

If you specify DIRECT NO, UNLOAD PLUS uses DB2 dynamic SQL to process the SELECT statement and read the table data. The DIRECT NO option enables the full range of functionality that the DB2 SQL SELECT statement provides.

**Note**

UNLOAD PLUS also uses the DIRECT NO option to enable new DB2 features quickly. UNLOAD PLUS generally will provide DIRECT YES support for these features in a later version.

DIRECT NO is not a high-performance solution for unloading large volumes of data. For more information about DIRECT NO mode, see “DIRECT NO” on page 29.

**Restrictions**

Note the following restrictions on the DIRECT NO option:

- Dynamic SQL processing cannot occur on identity columns or ROWID data types that are defined as GENERATED ALWAYS. If generating load control statements and DIRECT NO is in effect, UNLOAD PLUS generates the control statements as if the column is defined as GENERATED BY DEFAULT.

- When DIRECT NO is in effect, UNLOAD PLUS bypasses ANALYZE processing.

- UNLOAD PLUS does not support all options when you specify DIRECT NO. For example:

  — UNLOAD PLUS terminates when you specify DIRECT NO with FORMAT BMCLOAD.

  — UNLOAD PLUS ignores ORDER YES when you specify DIRECT NO.

For a full list of the options that are not available with DIRECT NO, see the table of options in “Alphabetical listing of UNLOAD PLUS options” on page 80.

**Additional consideration**

The following additional considerations apply when DIRECT NO is in effect:

- When you are dynamically allocating output data sets, you must also specify the ESTROWS, LIMIT, or SPACE command option.

- The default table name supplied in the INTO statement of the generated control statements is $user$ID.$TABLEn. To generate control statements with the correct table name, use INTO NAME ownerName.tableName.
ROWSETSZ

The ROWSETSZ option allows you to tell UNLOAD PLUS how many rows to include in a rowset for a single FETCH request. You can specify one of the values described in the following table:

Table 16: Values for the ROWSETSZ option

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or 1</td>
<td>Tells UNLOAD PLUS to fetch a single row at a time</td>
</tr>
<tr>
<td>2 through 200</td>
<td>Tells UNLOAD PLUS to fetch the specified number of rows (as a rowset)</td>
</tr>
</tbody>
</table>

BMC recommends a value of 100 for most jobs. However, see “Specifying command options for optimal performance in UNLOAD PLUS” on page 401 for more recommendations.

Restriction

UNLOAD PLUS ignores ROWSETSZ when you are unloading LOB or XML data.

Specifying the default

You can specify the default for the ROWSETSZ command option in your installation options module by using the ROWSETSZ installation option. UNLOAD PLUS was shipped with a default value of 100 for this option. The command option overrides the default that is in the installation options module.

AUTO

If you specify DIRECT AUTO, UNLOAD PLUS chooses between processing the job as if you specified DIRECT YES, or processing it as if you specified DIRECT NO. If a SELECT statement contains statements that are not within the scope of the UNLOAD PLUS SELECT-like syntax, UNLOAD PLUS uses DB2 dynamic SQL to read the data when processing the SELECT statement.

To ensure consistent behavior of your UNLOAD PLUS jobs, specify DIRECT YES or DIRECT NO. UNLOAD PLUS enhancements could change the behavior of DIRECT AUTO.

Restrictions

UNLOAD PLUS terminates when you specify DIRECT AUTO and any of the following conditions exists:

- You also specify FORMAT BMCLOAD.
- You are unloading any of the types of data that require DIRECT NO, as described in “NO” on page 100.

- The SELECT statement is processing a view that does not conform to UNLOAD PLUS restrictions for view processing.

**CURRENTDEGREE**

You can use the CURRENTDEGREE keyword to specify that you want DB2 to attempt to use parallelism.

This option is meaningful only when DIRECT NO is in effect.

Specifying CURRENTDEGREE ANY or CURRENTDEGREE 1 causes UNLOAD PLUS to issue the SET CURRENT DEGREE command before executing any dynamically executed SELECT statements. For more information about the SET CURRENT DEGREE command, see the appropriate IBM DB2 documentation.

**Restriction**

UNLOAD PLUS ignores this option when DIRECT YES is in effect.

**Specifying the default**

You can specify the default for the CURRENTDEGREE command option in your installation options module by using the CURRENTDEGREE installation option. UNLOAD PLUS was shipped with a default value of NONE for this option. The command option overrides the default that is in the installation options module.

**NONE**

Specify NONE if you do not want UNLOAD PLUS to issue the SET CURRENT DEGREE command to DB2. DB2 uses the system DB2 installation default values.

**ANY**

Specify ANY if you want UNLOAD PLUS to issue SET CURRENT DEGREE ANY before executing any dynamically executed SELECT statements.
1

Specify 1 if you want UNLOAD PLUS to issue SET CURRENT DEGREE 1 before executing any dynamically executed SELECT statements.

**SHRLEVEL**

The SHRLEVEL option specifies the level of access that concurrently operating DB2 applications and utilities have to the target spaces during UNLOAD PLUS processing.

**Restriction**

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**

You can specify the default for the SHRLEVEL command option in your installation options module by using the SHRLEVEL installation option. UNLOAD PLUS was shipped with a default value of REFERENCE for this option. The command option overrides the default that is in the installation options module.

**REFERENCE**

SHRLEVEL REFERENCE restricts object status to read-only during UNLOAD PLUS processing. UNLOAD PLUS then externalizes the pages of the table space. When UNLOAD PLUS is operating on partitioned objects, access is restricted only to the partitions that are being unloaded. When UNLOAD PLUS processing completes, the utility restores the object to its original status.

If you are unloading from a full or incremental image copy (including an encrypted copy or cabinet copy), UNLOAD PLUS unloads only from image copies that are marked as SHRLEVEL REFERENCE in SYSIBM.SYSCOPY or, for an encrypted copy or cabinet copy, the BMCXCOPY table.
**CHANGE**

If you are unloading from a table space, specify SHRLEVEL CHANGE to allow read/write access to the table space during unload processing.

If you are unloading from a full or incremental image copy (including an encrypted copy or cabinet copy), UNLOAD PLUS unloads from image copies marked as either SHRLEVEL REFERENCE or SHRLEVEL CHANGE in SYSIBM.SYSCOPY or, for an encrypted copy or cabinet copy, the BMCXCOPY table.

**CONSISTENT NO**

This option tells UNLOAD PLUS not to attempt to maintain consistency with a point-in-time image of the data when using SHRLEVEL CHANGE. When you specify CONSISTENT NO, UNLOAD PLUS does not restrict access to the object except for the brief time required to externalize pages if you specify QUIESCE YES. CONSISTENT NO is the default for SHRLEVEL CHANGE.

**Specifying the default**

You can specify the default for the SHRLEVEL CHANGE CONSISTENT command option in your installation options module by using the CHANGE_CONSISTENT installation option. UNLOAD PLUS was shipped with a default value of NO for this option. The command option overrides the default that is in the installation options module.

**QUIESCE**

This option tells UNLOAD PLUS whether to externalize the unloaded table space’s pages from the DB2 buffer pool. If you specify QUIESCE without a value, UNLOAD PLUS assumes QUIESCE YES. The following table describes the values that you can specify with QUIESCE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>YES tells UNLOAD PLUS to restrict access briefly to the object while UNLOAD PLUS externalizes pages from the DB2 buffer pool</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| NO    | Specify this value if you do not want UNLOAD PLUS to restrict access to the object and externalize pages from the DB2 buffer pool. **WARNING:** If you specify QUIESCE NO, UNLOAD PLUS does not restrict access to the objects, nor does it externalize pages from the DB2 buffer pool. These actions might result in the following consequences:  
- UNLOAD PLUS might not process any updated pages in the buffer pool.  
- If you run UNLOAD PLUS on a DB2 object immediately following the creation of the data set for that object, any data involved might exist only in DB2 buffer pools, causing UNLOAD PLUS to terminate.  
- If UNLOAD PLUS encounters any pages that have uncommitted data, it unloads the uncommitted data. |

**Specifying the default**

You can specify the default for the SHRLEVEL CHANGE CONSISTENT NO QUIESCE command option in your installation options module by using the CHANGE_QUIESCE installation option. UNLOAD PLUS was shipped with a default value of NO for this option. The command option overrides the default that is in the installation options module.

**CONSISTENT YES**

Specify CONSISTENT YES to have UNLOAD PLUS maintain consistency with a point-in-time image of the data while enabling read/write access to the data. UNLOAD PLUS provides this capability by using XBM or SUF to produce a snapshot image of the data in a table space.

When you specify SHRLEVEL CHANGE CONSISTENT YES, UNLOAD PLUS externalizes the pages of the table space, then activates XBM or SUF to begin caching pre-update page images. For more information about using XBM or SUF with UNLOAD PLUS, see “Running SHRLEVEL CHANGE CONSISTENT YES” on page 53. For detailed information about XBM and SUF, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.
Tip
You can use the following additional options:

- You can specify a particular XBM subsystem ID by using the XBMID installation or command option.

- To write a message to the MVS system log to indicate that snapshot-processing initialization for UNLOAD PLUS has successfully completed, specify the WTOMSG option.

Restriction
This option is not valid when you specify INFILE.

Specifying the default
You can specify the default for the SHRLEVEL CHANGE CONSISTENT command option in your installation options module by using the CHANGE_CONSISTENT installation option. UNLOAD PLUS was shipped with a default value of NO for this option. The command option overrides the default that is in the installation options module.

ZIIP
The ZIIP option tells UNLOAD PLUS whether to attempt to use IBM z Integrated Information Processors (zIIPs).

UNLOAD PLUS can use enclave service request blocks (SRBs) to enable zIIP processing automatically while running jobs. Using zIIP processing can reduce the overall CPU time for UNLOAD PLUS jobs.

Specifying the default
You can specify the default for the ZIIP command option in your installation options module by using the ZIIP installation option. UNLOAD PLUS was shipped with a default value of ENABLED for this option. The ZIIP command option overrides the default that is in the installation options module.
**ENABLED**

ZIIP ENABLED tells UNLOAD PLUS to attempt to offload eligible processing to an available zIIP. If the zIIP is busy or not available, normal processing continues on a general-purpose processor.

To enable and use zIIP processing with UNLOAD PLUS, you must meet the following requirements:

- Have an installed authorized version of XBM or SUF
- Start and maintain an XBM subsystem in your environment
- Have a zIIP available in your environment

**Using XBM or SUF**

You can specify a particular XBM subsystem to use by specifying a value for the XBMID installation or command option.

XBM and SUF are licensed, installed, and maintained separately from UNLOAD PLUS. You can use either XBM or SUF, depending on the license that you have obtained:

- A license for the full version of the XBM product authorizes you to use all features of XBM.
- A license for SUF authorizes you to use only the snapshot and zIIP-processing features of XBM.

**Note**

If you are licensed only for a BMC solution that contains UNLOAD PLUS, your license authorizes you to use SUF, not the full version of XBM.

For more information about XBM and SUF, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

---

**DISABLED**

ZIIP DISABLED tells UNLOAD PLUS to not attempt to use zIIP processing.
**XBMID**

Specify XBMID to identify a specific active XBM subsystem to use for snapshot processing and zIIP processing in UNLOAD PLUS.

The variable `ssid` (subsystem ID) is the unique identifier that you specified when you installed XBM or SUF. If you are using XBM or SUF in a DB2 data sharing environment, you can use the value of the `XBMGROUP` parameter in place of the `ssid`. The `XBMGROUP` is the name of the cross-system coupling facility (XCF) group that is defined to the XBM subsystem, and its default value is `XBMGROUP`.

If you specify an XBM subsystem, it must be:

- Available
- At a supported maintenance level
- Enabled for the required function

If you do not specify an XBM subsystem (either here or with the XBMID installation option), UNLOAD PLUS automatically searches for an XBM subsystem that meets the same criteria. If you have multiple subsystems that meet these criteria, UNLOAD PLUS can use any one of these subsystems.

For more information about using XBM or SUF for snapshot processing with UNLOAD PLUS, see “Running SHRLEVEL CHANGE CONSISTENT YES” on page 53.

**Specifying the default**

You can specify the default for the XBMID command option in your installation options module by using the XBMID installation option. The XBMID command option overrides the default that is in the installation options module.

**WTOMSG**

Specify WTOMSG SUFSTART to write message BMC50008I to the MVS system log to indicate that the snapshot-processing initialization has successfully completed.
This option is applicable only with SHRLEVEL CHANGE CONSISTENT YES.

You can use the text of this message to trigger the submission of jobs that you want to run concurrently with the utility job.

Specify a text string of up to 50 characters enclosed in single quotes to indicate the message to print in the MVS system log. The utility truncates strings that are greater than 50 characters. Quotes cannot appear within the text string.

**FILTERPART**

This option allows UNLOAD PLUS to filter out partitions that do not meet the WHERE clause criteria when you use the first column of the partitioning key in your WHERE clause.

[Diagram]

```
FILTERPART
  NO
  YES
```

Specifying PART within a SELECT OPTIONS statement overrides the FILTERPART command option for the corresponding SELECT statement.

*Restriction*

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**NO**

FILTERPART NO is the default. When you specify FILTERPART NO, UNLOAD PLUS reads all partitions of the partitioned table space.

**YES**

When you specify FILTERPART YES, UNLOAD PLUS filters out partitions that do not meet the selection criteria. UNLOAD PLUS uses only the SYSREC data sets that are related to the unloaded partitions.
Example

In the following example, COL1 is the column that causes the break of partitions. If you use COL1 in the WHERE clause, UNLOAD PLUS filters out partitions that do not meet the criteria of the WHERE clause. If you specify FILTERPART YES, UNLOAD PLUS reads only partitions 2 and 3.

Three-column partitioning key:

<table>
<thead>
<tr>
<th>Partition number</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 1</td>
<td>199</td>
<td>xxx</td>
<td>005</td>
</tr>
<tr>
<td>Partition 2</td>
<td>299</td>
<td>xxx</td>
<td>010</td>
</tr>
<tr>
<td>Partition 3</td>
<td>399</td>
<td>xxx</td>
<td>020</td>
</tr>
<tr>
<td>Partition 4</td>
<td>499</td>
<td>xxx</td>
<td>030</td>
</tr>
</tbody>
</table>

SELECT statement:

```
SELECT * FROM your.tableName
WHERE COL1 > '200'
AND COL1 < '399'
```

Restrictions

FILTERPART YES has the following restrictions:

- UNLOAD PLUS does not support this option for image copies. If you specify FILTERPART YES, UNLOAD PLUS issues message BMC50138I and functions as if you specified FILTERPART NO.

- UNLOAD PLUS ignores this option when unloading partition-by-growth table spaces.

Additional consideration

For the predicate on your WHERE clause, if you specify a value that equals the limit key, UNLOAD PLUS might read an additional partition.

INFILE

You can specify the INFILE option to use a copy or a data set other than the DB2 table space data set as the source of the input data. When you are unloading from multiple table spaces, the INFILE option applies to all table spaces.
This option is valid only when DIRECT YES is in effect.

The following table lists the alternate sources of input data that you can use, and tells you which INFILE option to use for each input type:

Table 18: Alternate sources of UNLOAD PLUS input

<table>
<thead>
<tr>
<th>Input type</th>
<th>INFILE option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full or incremental image copy data sets, including inline copies</td>
<td>IMAGECOPY or ddname</td>
</tr>
<tr>
<td>Encrypted copies created by COPY PLUS</td>
<td>IMAGECOPY</td>
</tr>
<tr>
<td>Cabinet copies created by the Recovery Management or Recovery solution</td>
<td>IMAGECOPY</td>
</tr>
<tr>
<td>DSN1COPY sequential data sets</td>
<td>ddname</td>
</tr>
<tr>
<td>Instant Snapshot copies created by COPY PLUS</td>
<td>SNAPCOPY</td>
</tr>
<tr>
<td>Online consistent copies created by the Online Consistent Copy component of the Recovery Management or Recovery solution</td>
<td>SNAPCOPY</td>
</tr>
<tr>
<td>VSAM linear data sets</td>
<td>VSAMDD or VSAMDDPREFIX</td>
</tr>
<tr>
<td>VSAM FlashCopy image copies</td>
<td>VSAMDD or VSAMDDPREFIX</td>
</tr>
</tbody>
</table>

Restrictions

The following restrictions apply to the INFILE option:

- You cannot specify FILTERPART YES when unloading from image copies.
- UNLOAD PLUS does not unload from LOB or XML copy data sets. However, you can unload from a copy of the base table if you do not select any LOB or XML columns.
**Additional considerations**

The following additional considerations apply to the INFILE option:

- Unpredictable results might occur if the data sets do not match the options that you specify:
  - If the input data set is an inline copy, specify INLINE YES.
  - If the input data set is an incremental image copy, specify the INCREMENTAL option.
  - If you are unloading all partitions of a partitioned table space, ensure that the data set includes a full image copy of all of the partitions.
  - If you are unloading specific partitions, ensure that the input provides all of the necessary partitions. Provide the full image copy data set by using one ddname, or provide image copies for each partition by using multiple ddnames that end with a partition number.

- When you specify INFILE to unload any of the following types of copy data sets, all selected rows must be at the current version (as defined in SYSIBM.SYSTABLES):
  - An incremental image copy
  - Any image copy created with the SYSTEMPAGES NO copy option

- When you specify INFILE, UNLOAD PLUS bypasses ANALYZE processing. Therefore, if you are dynamically allocating output data sets, you must also specify one of the following command options to provide data set sizing information:
  - ESTROWS
  - LIMIT
  - SPACE

**Note**

The SPACE *installation* option does not provide this information.

- To unload all rows of a copy data set that contains more partitions than the corresponding DB2 table space, specify the following:
  - INFILE *ddname*
  - A DDLIN data set that contains DDL that matches the copy data set that you are unloading
**IMAGECOPY**

Specify INFILE IMAGECOPY to have UNLOAD PLUS use the specified full or incremental image copy data set that is in the SYSIBM.SYSCOPY table or the BMCXCOPY table; BMCXCOPY applies only if you are unloading an encrypted image copy created by COPY PLUS or a cabinet copy.

*Note*
Encrypted image copies are registered in BMCXCOPY as STYPE e. Cabinet copies are registered as COPY_TYPE C.

If the table space is partitioned, UNLOAD PLUS uses either individual partition image copies (DSNUM is not equal to 0) or, if there is not one copy for each partition that you unload, UNLOAD PLUS uses the first image copy that fulfills your specifications and that contains all partitions (DSNUM is equal to 0).

UNLOAD PLUS can unload multi-data-set, nonpartitioned table spaces only from a single image copy that contains all of the individual data sets. You cannot unload nonpartitioned table spaces from separate image copy data sets where DSNUM is not equal to 0.

**Additional restrictions and considerations**

In addition to the general restrictions and considerations for the INFILE option, the following restrictions and considerations apply to INFILE IMAGECOPY:

- To unload an encrypted copy that was created by COPY PLUS, you must
  - Run UNLOAD PLUS on a processor that supports encryption
  - Use the KEYDSNAM installation option to specify your key data set name

  *Note*
  For more information about creating encrypted copies and about the key data set, see the *COPY PLUS for DB2 Reference Manual*.

- UNLOAD PLUS does not unload from copies created with the SYSTEMPAGES NO option if those copies contain compression dictionaries created during DB2 SQL INSERT processing.

- UNLOAD PLUS does not automatically use local backup or recovery copies. To unload a local backup or recovery copy, you must specify it with the INFILE *ddname* option.

- If you specify -integer and you are running SHRLEVEL REFERENCE, UNLOAD PLUS looks only for image copies that are registered as REFERENCE copies and
ignores image copies that are registered as CHANGE. If you are running SHRLEVEL CHANGE, UNLOAD PLUS looks for copies that are registered as REFERENCE and copies that are registered as CHANGE.

FULL

FULL, which is the default, tells UNLOAD PLUS to unload from a full image copy. By default, or if you specify 0 (with no minus sign), UNLOAD PLUS unloads from the most recent full image copy found in the SYSIBM.SYSCOPY table or the BMCXCOPY table; BMCXCOPY applies only if you are unloading an encrypted copy created by COPY PLUS or a cabinet copy.

To unload from a full image copy before the most recent one, specify -integer to direct UNLOAD PLUS to count the specified number of full image copies back from the most recent one, and unload that image copy. For example, if you specify INFILE IMAGECOPY FULL -1, UNLOAD PLUS unloads the full image copy immediately preceding the most recent one. Note the minus sign in front of the integer.

If you specify -integer with the FULL keyword, UNLOAD PLUS looks only for full image copies. UNLOAD PLUS ignores any incremental image copies that it finds.

INCREMENTAL

This option tells UNLOAD PLUS to unload from an incremental image copy. When unloading from an incremental image copy, UNLOAD PLUS performs the following actions:

- Accesses the previous full copy for data set information (but does not unload from that copy)
- Reads all rows in the incremental image copy (not only the rows that changed) and unloads those that match the SELECT statement criteria

By default, or if you specify 0 (with no minus sign), UNLOAD PLUS unloads the most recent incremental image copy found in the SYSIBM.SYSCOPY table or the BMCXCOPY table; BMCXCOPY applies only if you are unloading an encrypted copy created by COPY PLUS or a cabinet copy.

If you want to unload from an incremental image copy that was created before the most recent one, specify -integer to direct UNLOAD PLUS to count the specified number of incremental image copies back from the most recent one and unload that copy. For example, if you specify INFILE IMAGECOPY INCREMENTAL -1, UNLOAD PLUS unloads the incremental image copy that was created just before the most recent one.

Additional considerations
In addition to the general restrictions and considerations for the INFILE option and the considerations for INFILE IMAGECOPY, the following considerations apply to the INCREMENTAL keyword:

- If you specify -integer with the INCREMENTAL keyword, UNLOAD PLUS counts only incremental image copies. UNLOAD PLUS does not include any full image copies that it finds in the -integer count.

- UNLOAD PLUS cannot use the INFILE option to unload incremental image copies that do not contain dictionary pages when the image copy contains compressed data.

**ddname**

Use this option to specify a ddname or ddname prefix when unloading from a DSN1COPY sequential data set, a specific full or incremental image copy, or an inline image copy. For more information about specifying ddnames, see “UNLOAD PLUS input copy data set” on page 279.

**Note**

Do not use this option to unload from a cabinet copy; use INFILE IMAGECOPY instead.

If you specify INFILE ddname, UNLOAD PLUS unloads all rows from the specified data set that match your SELECT statement criteria.

**INFILE ddname with DDL**

When you specify INFILE ddname and supply DDL (using a DDLIN data set), the following warnings and restrictions apply:

- The DDL must match the definition of the object that you are unloading. If they do not match, results are unpredictable.

- Ensure that the DDL reflects the limit key specifications that applied to the table space when the image copy was taken. This requirement is particularly important if you specify PART on the UNLOAD command and any of the following actions were performed on the table space after the copy was taken:
  - Rotated partitions
  - Altered limit keys
  - Rebalanced partitions during a reorganization

If the DDL does not reflect the appropriate limit key specifications, results are unpredictable.
UNLOAD PLUS options

- All selected rows must be at version 0. Otherwise, UNLOAD PLUS terminates.

- If either of the following conditions exists, you must specify the OBID of the table, either with each SELECT statement or on each CREATE TABLE statement in the DDLIN data set:
  - You specify multiple SELECT statements.
  - You are unloading a multi-table table space.

- If you also specify LOGICAL PART, UNLOAD PLUS ignores the LOGICAL keyword and considers the specified partitions to be physical partitions.

Additional restrictions and considerations

In addition to the general restrictions and considerations on the INFILE option, the following restrictions and considerations apply to INFILE ddname:

- The following DB2 catalog information must match the copy that you are unloading:
  - The database identifier (DBID) and page set identifier (PSID) of the database and table space that contain the specified tables, unless you specify ON MESSAGE 50253 CONTINUE UTILITY
  - The OBIDs of the tables that you specify, unless you specify the OBID for the table by using the OBID option

  - The table definitions

  However, if you added more columns to a table since the copy was made and you are unloading those columns, the new columns contain their default values.

  If the table definitions that are in the DB2 catalog do not match the table in the image copy, you can provide DDL by using the DDLIN data set. UNLOAD PLUS can then use DDL for DB2 object definitions, instead of using the object definitions in the DB2 catalog. For more information about this data set, see “DDLIN data sets in UNLOAD PLUS” on page 278.

- UNLOAD PLUS does not unload copies that contain compression dictionaries created during DB2 SQL INSERT processing for either of the following types of copies:
  - Copies created with the DSN1COPY utility
  - Copies created with the SYSTEMPAGES NO option

- Concatenating multiple image copy data sets, particularly when mixing copy data sets from tape with copy data sets from DASD, produces unpredictable results.
When unloading an image copy on a DB2 subsystem other than the one on which the copy was created, the following information applies:

— For a range-partitioned table space, UNLOAD PLUS unloads only the number of partitions that are defined on the target subsystem. To ensure that you unload the partitions that you intend, BMC recommends that you use a DDLIN data set.

— For a partition-by-growth table space, UNLOAD PLUS unloads only the number of partitions that exist on the target subsystem. To ensure that you unload the partitions that you intend, BMC recommends that you use a DDLIN data set and specify INLINE YES.

**FULL**

Specify this option to tell UNLOAD PLUS that the specified file is a full image copy, a DSN1COPY data set, or an inline copy.

**INLINE**

Use this option to specify whether UNLOAD PLUS should treat the copy that is associated with the specified ddname as an inline copy. The following table describes the values that you can specify for this option:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>Specify INLINE YES after FULL to tell UNLOAD PLUS to treat the copy that is associated with the ddname as an inline copy. This option enables UNLOAD PLUS to unload data from out-of-sequence pages. When you specify INLINE YES, UNLOAD PLUS treats all image copies as inline copies and reads the data twice to ensure data integrity. This action can result in performance degradation especially for copies that reside on tape.</td>
</tr>
<tr>
<td>NO</td>
<td>Specify INLINE NO after FULL to tell UNLOAD PLUS not to treat the copy as an inline copy. <strong>WARNING:</strong> UNLOAD PLUS supports unloading data from an inline image copy when you specify INFILE <em>ddname</em> and INLINE YES. Inline image copies can have out-of-sequence pages. However, UNLOAD PLUS cannot determine pages that might be out of sequence before processing begins unless you also specify INLINE YES. Using an inline copy that contains pages that are out of sequence can cause a serious page error (see message BMC50251S) if you specify INLINE NO.</td>
</tr>
</tbody>
</table>

**INCREMENTAL**

Specify this option to tell UNLOAD PLUS that the specified file is an incremental image copy. If INFILE *ddname* specifies an incremental image copy, you **must** use this option or UNLOAD PLUS assumes that FULL (the default) applies, and unpredictable results might occur. When using an
incremental image copy, UNLOAD PLUS reads all rows in that image copy, not only the rows that changed.

--- Note ---
UNLOAD PLUS cannot use the INFILE option to unload incremental image copies that do not contain dictionary pages when the image copy contains compressed data.

SNAPCOPY

If you specify INFILE SNAPCOPY, UNLOAD PLUS uses the data set for the specified Instant Snapshot copy or online consistent copy that is registered in the BMCXCOPY table (ICTYPE F or C).

-integer

UNLOAD PLUS always unloads from a full copy. If you specify 0 (with no minus sign), UNLOAD PLUS unloads from the most recent Instant Snapshot copy or online consistent copy that is registered in the BMCXCOPY table.

To unload from a copy that precedes the most recent one, specify -integer to tell UNLOAD PLUS which preceding Instant Snapshot copy or online consistent copy to unload (where -integer indicates the copy’s relation to the most recent copy). For example, specifying INFILE SNAPCOPY -1 unloads the copy immediately preceding the most recent one. INFILE SNAPCOPY -3 unloads the third copy preceding the most recent one.

UNLOAD PLUS can unload multi-data-set, nonpartitioned table spaces only from a single image copy that contains all of the individual data sets. You cannot unload nonpartitioned table spaces from separate image copy data sets where DSNUM is greater than 0.

VSAMDD

Specify this option to unload from a VSAM linear data set or VSAM FlashCopy image copy data set. The data set must be specified as a VSAMDD DD statement in your JCL.

For more information about specifying ddnames, see “VSAMDD data sets in UNLOAD PLUS” on page 281.

--- Restriction ---
You cannot specify multiple VSAM data sets for nonpartitioned objects.

--- Considerations ---
The following considerations apply to the VSAMDD option:

- When unloading multiple partitions, add the partition number as a suffix to the VSAMDD DD statements in your JCL.

- With this option, you can specify up to 99 VSAM data sets for partitioned objects. If you need to unload from more than 99 data sets, use the VSAMDDPREFIX option.

**VSAMDDPREFIX prefix**

Use this option to specify a prefix other than VSAMDD to use when unloading from a VSAM linear data set or VSAM FlashCopy image copy data set. This option enables you to unload from more than 99 data sets.

Your JCL must include DD statements that match the prefix that you specify. For multiple data sets, you must append \( n \) to the ddnames in the DD statements in your JCL, where \( n \) is a numeric value. These ddnames must not exceed eight characters.

For more information about specifying ddnames, see “VSAMDD data sets in UNLOAD PLUS” on page 281.

**Restriction**

You cannot specify multiple VSAM data sets for nonpartitioned objects.

**PART or LOGICAL PART**

The PART option specifies the partition numbers of a partitioned table space to unload.

```
PART partitionNumber
```

*This option is valid only when DIRECT YES is in effect.*

You can specify one, some, or all of the partitions either individually or by specifying a range. If you do not specify PART, UNLOAD PLUS unloads all partitions of the table space.

Optionally, you can specify the LOGICAL keyword with PART to indicate that you are specifying the logical partitions to unload rather than the physical partitions.

*Specifying partition numbers*
Note the following information about how to specify partition numbers with the PART option:

- Individual partitions in a list can be in any order (but partitions within a range must be in ascending order).
- You can specify a mixture of individual partitions and ranges of partitions.
- If you specify a partition number more than once, UNLOAD PLUS ignores any occurrence after the first.
- You can specify integer values from 1 through 4096.

The following example illustrates a valid PART specification for partitions 1 through 5, 7, and 10:

```
PART 10,1:5,7
```

**Restrictions**

The following restrictions apply to the PART option:

- UNLOAD PLUS terminates if you specify both logical and physical partitions in the same job.
- If you specify INFILE `ddname` and supply DDL (using a DDLIN data set), UNLOAD PLUS ignores the LOGICAL keyword and considers the specified partitions to be physical partitions.

**Additional considerations**

The following considerations apply to the PART option:

- When you are unloading XML data, the PART option applies only to the base table space.
- When you are unloading from multiple table spaces, the PART option applies to all table spaces.
- Specifying PART within a SELECT OPTIONS statement overrides the PART command option for that SELECT statement.
- Unloading objects with a large number of partitions increases the potential for encountering such issues as performance problems and memory restrictions. For recommendations, see “Tuning when unloading objects with a large number of partitions” on page 415.
**DDLDDN**

The DDLDDN option allows you to override the default ddname of the DDL input data set.

This option is valid only when DIRECT YES is in effect.

This optional data set contains DDL that defines all of the objects that are related to the tables that you are unloading when you specify the INFILE ddname option. The default is DDLIN.

When you specify the INFILE ddname option, and the table definitions in the DB2 catalog do not match the table in the image copy, you can provide DDL by using the DDLIN data set. UNLOAD PLUS can then use this DDL for DB2 object definitions, instead of using the object definitions in the DB2 catalog.

**UNLOADDN**

The UNLOADDN option provides UNLOAD PLUS with a default ddname, prefix, or output descriptor for the unload data sets.

This option provides the following information:

- When unload data sets are allocated in the JCL, an override for the default ddname or prefix of those data sets
- When allocating unload data sets dynamically, the output descriptor or prefix to match to the OUTPUT statement

Specify the second ddname or output descriptor if you want to produce dual unload data sets. If you specify primary and secondary values, they must be unique. The values that you specify with this option must also be unique when compared to the output descriptors of any referenced file data sets.
For more information about specifying unload data sets, see “SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286.

**Multiple data sets**

When specifying multiple unload data sets, the ddname that you specify becomes the ddname prefix. You must append *n* to the ddnames in the DD statements in your JCL, where *n* is a numeric value. These ddnames must not exceed eight characters.

If you specify more than 99 data sets, use the UNLOADDN command option to override the default data set names of SYSREC and SYSRED, specifying a ddname prefix that results in eight characters or less after appending the data set number.

**Dynamic allocation**

When UNLOAD PLUS dynamically allocates unload data sets, the *outputDescriptor* variable represents an output descriptor name or prefix. This name or prefix enables UNLOAD PLUS to match the dynamic allocation option values that you specify on the OUTPUT command with the correct unload data sets. For more information, see “OUTPUT” on page 182.

BMC does not recommend combining dynamically allocated SYSREC data sets with SYSRED data sets that are allocated in your JCL.

**Specifying the default**

You can specify the default for the UNLOADDN command option in your installation options module by using the UNLOADDN installation option. UNLOAD PLUS was shipped with a default value of (SYSREC,SYSRED) for this option. The command option overrides the default that is in the installation options module.

**ACTIVE**

The ACTIVE option tells UNLOAD PLUS whether to allocate your primary and secondary unload data sets dynamically. The first position applies to the primary data set; the second position applies to the secondary data set. For example, you can specify UNLOADDN(SYSREC,SYSRED) ACTIVE(YES,YES) to enable dynamic allocation for both primary and secondary unload data sets. The following table describes the values that you can specify for this option:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>Activates dynamic allocation for the corresponding unload data set. To activate dynamic allocation for both data sets, specify (YES,YES).</td>
</tr>
</tbody>
</table>
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>Does not activate dynamic allocation for the corresponding unload data set To not allocate either data set dynamically, specify (NO,NO).</td>
</tr>
</tbody>
</table>

### Additional consideration

When at least one ACTIVE parameter is YES and one of the following conditions exists, you must also specify the ESTROWS, LIMIT, or SPACE command option to provide UNLOAD PLUS with data set sizing information:

- DIRECT NO is in effect.
- You specify the INFILE option.

**Note**
The SPACE *installation* option does not provide this information.

In all other cases, UNLOAD PLUS uses ANALYZE processing to obtain data set sizing information for dynamic allocation.

### Specifying the default

You can specify the default for the ACTIVE command option in your installation options module by using the UNLOADDN_ACTIVE installation option. UNLOAD PLUS was shipped with a default value of (YES,NO) for this option. The command option overrides the default that is in the installation options module.

### ANALYZE

The ANALYZE option tells UNLOAD PLUS to estimate the number of rows for a specific table or partition.

This option is meaningful only when DIRECT YES is in effect.

UNLOAD PLUS uses the results of the ANALYZE option to calculate the sizes of unload data sets during dynamic allocation.
Considerations

The following considerations apply to the ANALYZE option:

- If you are using DIRECT NO or INFILE processing, UNLOAD PLUS bypasses ANALYZE processing.

- If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

- BMC recommends that you use DB2STATS for the most accurate row estimations. HURBA calculations are based on the size of the DB2 VSAM file and the average maximum row length of all tables in a table space. In situations where compressed or variable-length fields are used, ANALYZE HURBA might report fewer rows than actual for a table space.

Specifying the default

You can specify the default for the ANALYZE command option in your installation options module by using the ANALYZE installation option. UNLOAD PLUS was shipped with a default value of (DB2STATS,NOLIMIT) for this option. The command option overrides the default that is in the installation options module.

DB2STATS

Specify DB2STATS if you want UNLOAD PLUS to use statistical information from the DB2 catalog to estimate the number of rows for a specific table and partition. When you specify ANALYZE DBSTATS and no statistics are available, UNLOAD PLUS defaults to HURBA.

AGE

You can specify the age of the statistics in the DB2 catalog to use when analyzing and calculating the sizes of unload data sets during dynamic allocation.

NOLIMIT

Specify AGE NOLIMIT if you want UNLOAD PLUS to use the catalog information and disregard the age of the statistics.

integer

Specify the number of days at which you want to limit the age of the statistics that UNLOAD PLUS uses. If you do not specify a value, or if the value that you specify is less than the age of the statistics in the DB2
To determine the age of the statistics, use the following formula:

\[(\text{DAYS (CURRENT\_DATE)} - \text{DAYS (STATSTIME)})\]

**HURBA**

Specify HURBA if you want UNLOAD PLUS to use the high-used RBA (HURBA) from the DB2 table space to estimate the number of rows for a specific table and object. UNLOAD PLUS does not use statistics when you specify this option.

When you specify HURBA to size a table or table space, UNLOAD PLUS uses the following formula to calculate the estimated number of rows.

**Formula**

\[
\text{estimated rows per table} = \frac{\text{the number of estimated rows per object}}{\text{estimated rows per object}} = ((\text{HURBA}/\text{PGSZ}-2) \times \text{RPP})
\]

**Table 21: ANALYZE HURBA formula parameters**

<table>
<thead>
<tr>
<th>Formula parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HURBA</td>
<td>Sum of the VSAM high-used RBAs for the DB2 page set or object</td>
</tr>
<tr>
<td>PGSZ</td>
<td>Page size of the table space in bytes</td>
</tr>
<tr>
<td>RPP</td>
<td>Calculated rows per page</td>
</tr>
<tr>
<td></td>
<td>- Uncompressed objects: (\text{PGSZ}/(\text{AVGROWSZ})) limited by MAXROWS</td>
</tr>
<tr>
<td></td>
<td>- Compressed objects: (\text{PGSZ}/(\text{AVGROWSZ} \times (\text{CMRATIO}/100))) limited by MAXROWS</td>
</tr>
<tr>
<td>AVGROWSZ</td>
<td>Average row length</td>
</tr>
<tr>
<td>MAXROWS</td>
<td>MAXROWS value (default 255) specified on the table space definition</td>
</tr>
<tr>
<td>CMRATIO</td>
<td>Compression ratio</td>
</tr>
</tbody>
</table>

**CMRATIO**

For compressed DB2 objects, specify CMRATIO if you want UNLOAD PLUS to use a compression ratio to calculate the estimated number of rows for a table space when any of the following conditions exists:

- When HURBA is specified
- As a default because there are no DB2 statistics
When the statistics are out of date

The value (1 through 100) that you specify for this option represents the percentage to use. For example, if a page would normally hold 10 rows in an uncompressed state, specifying a compression ratio of 50 indicates that these 10 rows occupy only 50% of the page so that the number of rows that would fit in the page would now be 20.

**Specifying the default**

You can specify the default for the CMRATIO command option in your installation options module by using the CMRATIO installation option. UNLOAD PLUS was shipped with a default value of 50 for this option. The command option overrides the default that is in the installation options module.

**USELRECL**

You can specify the USELRECL option to tell UNLOAD PLUS how to determine the record length.

- **NO**
  - Specify USELRECL NO to have UNLOAD PLUS use the record length that it calculates instead of one of the specified LRECL values.

- **YES**
  - When you specify USELRECL YES, UNLOAD PLUS

  - Checks and uses one of the following values:
    - The logical record length (LRECL) value in the data control block (DCB) for the primary unload data set DD statement in your JCL (that is, SYSREC)
— The LRECL option specified on the OUTPUT statement that corresponds to each SELECT statement.

- Uses fixed block format for the data set records and pads the records with spaces of the appropriate encoding scheme when necessary

**Note**

If you specify RECFM VB on your UNLOAD command (or it is in effect by default from the RECFM installation option) UNLOAD PLUS uses variable block format for the data set records.

- Forces the same DCB values in the corresponding secondary unload data set (SYSRED)

- Calculates the LRECL if you did not specify an LRECL value on the DD statement or in your OUTPUT statement

  If UNLOAD PLUS calculates the LRECL because you did not specify one in your JCL DD statement, the data set will be fixed, variable, or variable-spanned, according to the standard record length and record format calculations for UNLOAD PLUS.

**Restrictions**

Note the following restrictions on USELRECL YES:

- If the specified LRECL is shorter than the length that is required to contain the largest formatted row, UNLOAD PLUS ends with error message BMC51649E.

- UNLOAD PLUS ignores USELRECL YES if you specify any of the following options:
  
  - FORMAT CSV
  - FORMAT XML
  - FORMAT INTERNAL
  - FORMAT BMCLOAD

- If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

**MAXBLKSIZE**

The MAXBLKSIZE option allows you to override the SYSREC block size, which UNLOAD PLUS automatically calculates.
When you specify the MAXBLKSIZE option, UNLOAD PLUS uses the largest even multiple of the record size that is less than or equal to the value that you specify. To accommodate access method requirements, you must specify a value for MAXBLKSIZE that is greater than the formatted record length plus 8 bytes.

To prevent creating a variable-block spanned (VBS) file when unloading data from a table where the LRECL is greater than half a track, you must specify MAXBLKSIZE.

Restrictions

Note the following restrictions on MAXBLKSIZE:

- UNLOAD PLUS terminates when you specify MAXBLKSIZE with FORMAT BMCLOAD.
- If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

RECFM

The RECFM option allows you to specify the record block format for the unload files (SYSREC and SYSRED).

If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

Specifying the default

You can specify the default for the RECFM command option in your installation options module by using the RECFM installation option. UNLOAD PLUS was shipped with a default value of AUTO for this option. The command option overrides the default that is in the installation options module.

AUTO

Specify AUTO to tell UNLOAD PLUS to determine the optimal record block format.
**VB**

Specify VB to tell UNLOAD PLUS to use variable block format (VB) for your unload file, regardless of any other specification in your JCL or on your UNLOAD command.

---

**Note**

If the record length of your unload file is greater than the system maximum block size, UNLOAD PLUS defines the data set as variable-block spanned (VBS) instead of VB. In that case, UNLOAD PLUS issues message BMC51745W.

---

**SPANNED**

The SPANNED option tells UNLOAD PLUS whether you want to unload your LOB and XML data to a standard variable-block spanned (VBS) data set. The format of this data set is similar to the IBM proprietary spanned record format.

\[
\text{SPANNED} \quad \text{NO} \quad \text{YES}
\]

This option is valid only when DIRECT YES is in effect.

---

**Note**

The FORMAT SPANNED YES option of the IBM DB2 LOAD utility requires that the input be in the IBM proprietary spanned record format. Therefore, you cannot use the IBM DB2 LOAD utility to load the VBS data set that UNLOAD PLUS produces.

---

**NO**

NO, the default, tells UNLOAD PLUS not to unload LOB and XML data to a VBS data set.

**YES**

YES tells UNLOAD PLUS to unload LOB and XML data to a VBS data set. For more information, see “Unloading LOB and XML data” on page 54.

For considerations that apply when you specify SPANNED YES, see “Considerations when unloading LOB or XML data to unload data sets” on page 58.
The CNTLCARDS option specifies the type of control statements that UNLOAD PLUS writes to the SYSCNTL data set.

For more information about generating control statements and about inserting additional syntax into control statements, see “Generating control statements for DB2 or other software products” on page 499.

Additional considerations

The following requirements and restrictions apply to the CNTLCARDS option:

- You must specify a SYSCNTL data set in your JCL.
- The value that you specify for your CNTLCARDS option applies to all SELECT statements in the job.
- If you specify FORMAT XML, UNLOAD PLUS ignores the CNTLCARDS option.
DB2LOAD

CNTLCARDS DB2LOAD is the default (when your JCL contains a SYSCNTL DD statement). This value tells UNLOAD PLUS to generate only DB2 LOAD utility control statements in the SYSCNTL data set.

Additional considerations

The following considerations are specific to CNTLCARDS DB2LOAD. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

- When you specify ENUMROWS as the first string in the CNTLCARDS, UNLOAD PLUS generates an ENUMROWS value 'ENUMROWS (, nn)' during the unload phase. This value indicates the number of rows that UNLOAD PLUS unloads.

- When you specify SPANNED YES, UNLOAD PLUS adds the FORMAT SPANNED YES option to the control cards that it generates.

- When you are unloading XML data, UNLOAD PLUS adds the PRESERVE WHITESPACE option to the field specification for the XML column.

- When you are unloading row ID columns and DIRECT NO is in effect, UNLOAD PLUS generates load control statements for those columns as if they were defined as GENERATED BY DEFAULT.

- The following considerations apply when you are unloading a system-period temporal object:
  - When DIRECT YES is in effect, UNLOAD PLUS does not generate field specification control cards for temporal-specific fields.
  - When DIRECT NO is in effect, UNLOAD PLUS generates field specification control cards for temporal-specific fields.

- When you are unloading application-period temporal data, UNLOAD PLUS generates field specification control cards for business-begin and business-end fields.

'string'

Use this option to specify a string of additional LOAD syntax to insert into the IBM DB2 LOAD utility or BMC LOADPLUS product control statements. You can specify multiple strings of up to 72 bytes each. Enclose each string in single quotes. To embed a quoted string (a string enclosed by single quotation marks) within a string, add a single quotation mark before and after the quoted string (for example, 'CITY=' 'AUSTIN' '). For more information, including examples, see “Inserting additional command options into utility control statements” on page 500.
You can also insert additional load syntax to follow the INTO option for the IBM DB2 LOAD or the BMC LOADPLUS generated utility control statements. For more information, see “INTO” on page 208

**DB2**

CNTLCARDS DB2 tells UNLOAD PLUS to generate DB2 CREATE TABLE DDL and DB2 LOAD utility control statements in the SYSCNTL data set. UNLOAD PLUS does not generate EDITPROCs or FIELDPRECs.

Additional considerations

The following considerations are specific to CNTLCARDS DB2. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

- If you specify ENUMROWS as the first string in the CNTLCARDS, UNLOAD PLUS generates an ENUMROWS value 'ENUMROWS (,nn)' during the unload phase. This value indicates the number of rows that UNLOAD PLUS unloads.

- When you are unloading inline LOB data and DIRECT YES is in effect, UNLOAD PLUS adds the INLINE LENGTH option to the column definition or field specification for the LOB column.

- When you specify SPANNED YES, UNLOAD PLUS adds the FORMAT SPANNED YES option to the control cards that it generates.

- When you are unloading XML data, UNLOAD PLUS adds the PRESERVE WHITESPACE option to the column definition or field specification for the XML column.

- When you are unloading row ID columns and DIRECT NO is in effect, UNLOAD PLUS generates load control statements for those columns as if they were defined as GENERATED BY DEFAULT.

- The following considerations apply when you are unloading a system-period temporal object:

  — When DIRECT YES is in effect, UNLOAD PLUS

  — Generates temporal-specific column definitions for the system-begin, system-end, and transaction-start-ID fields

  — Generates the PERIOD SYSTEM_TIME definition

  — *Does not* generate field specification control cards for temporal-specific fields
When DIRECT NO is in effect, UNLOAD PLUS

— Generates standard TIMESTAMP column definitions for the system-begin, system-end, and transaction-start-ID fields, but does not include temporal-specific definitions

—*Does not* generate the PERIOD SYSTEM_TIME definition

— Generates field specification control cards for temporal-specific fields.

When you are unloading application-period temporal data, UNLOAD PLUS generates column definition and field specification control cards for business-begin and business-end fields. UNLOAD PLUS generates the PERIOD BUSINESS_TIME definition only when DIRECT YES is in effect.

'\textit{string}'

Use this option to specify a string of additional LOAD syntax to insert into the IBM DB2 LOAD utility or BMC LOADPLUS product control statements. You can specify multiple strings of up to 72 bytes each. Enclose each string in single quotes. To embed a *quoted string* (a string enclosed by single quotation marks) within a string, add a single quotation mark before and after the quoted string (for example, \texttt{\textquoteleft CITY='AUSTIN'\textquoteleft}). For more information, including examples, see “Inserting additional command options into utility control statements” on page 500.

\textbf{Note}

You can also insert additional load syntax to follow the INTO option for the IBM DB2 LOAD or the BMC LOADPLUS generated utility control statements. For more information, see “\texttt{INTO}” on page 208

\textbf{BMCLOAD}

CNTLCARDS BMCLOAD tells UNLOAD PLUS to generate load control statements for LOADPLUS. You can use CNTLCARDS BMCLOAD to create INDSN load control statements when any of the primary unload data sets are dynamically allocated.

The following restrictions and considerations are specific to CNTLCARDS BMCLOAD. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

\textbf{Restrictions}

UNLOAD PLUS terminates if you specify CNTLCARDS BMCLOAD and SPANNED NO is in effect when unloading LOB or XML columns.

\textbf{Additional considerations}
The following considerations apply to CNTLCARDS BMCLOAD:

- If you specify ENUMROWS as the first string in the CNTLCARDS, UNLOAD PLUS generates an ENUMROWS value 'ENUMROWS (,nn)' during the unload phase. This value indicates the number of rows that UNLOAD PLUS unloads.

- When you specify SPANNED YES, UNLOAD PLUS adds the FORMAT SPANNED YES option to the control cards that it generates.

- If you specify CNTLCARDS BMCLOAD and the following conditions exist, also specify AUTOTAG YES to have UNLOAD PLUS generate one LOAD statement with multiple INTO clauses:
  - You are using a single SYSCNTL data set.
  - You specify multiple SELECT statements on tables that are in the same table space.

  **Note**
  If you also specify FORMAT BMCLOAD, you cannot specify AUTOTAG YES.

- When you are unloading row ID columns and DIRECT NO is in effect, UNLOAD PLUS generates load control statements for those columns as if they were defined as GENERATED BY DEFAULT.

- The following considerations apply when you are unloading a system-period temporal object:
  - When DIRECT YES is in effect, UNLOAD PLUS does not generate field specification control cards for temporal-specific fields.
  - When DIRECT NO is in effect, UNLOAD PLUS generates field specification control cards for temporal-specific fields.

- When you are unloading application-period temporal data, UNLOAD PLUS generates field specification control cards for business-begin and business-end fields.

'**string**'

Use this option to specify a string of additional LOAD syntax to insert into the LOADPLUS control statements. You can specify multiple strings of up to 72 bytes each. Enclose each string in single quotes. To embed a quoted string (a string enclosed by single quotation marks) within a string, add a single quotation mark before and after the quoted string (for example, 'CITY='''AUSTIN''). For more information, including examples, see “Inserting additional command options into utility control statements” on page 500.
**Note**

You can also insert additional load syntax to follow the INTO option for the BMC LOADPLUS generated utility control statements. For more information, see “INTO” on page 208.

---

**DB2DDL**

CNTLCARDS DB2DDL tells UNLOAD PLUS to generate DB2 CREATE TABLE DDL in the SYSCNTL data set. UNLOAD PLUS does not generate EDITPROCs or FIELDPROCs.

**Considerations**

The following considerations are specific to CNTLCARDS DB2DDL. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

- The following considerations apply when you are unloading a system-period temporal object:
  - When DIRECT YES is in effect, UNLOAD PLUS
    - Generates temporal-specific column definitions for the system-begin, system-end, and transaction-start-ID fields
    - Generates the PERIOD SYSTEM_TIME definition
  - When DIRECT NO is in effect, UNLOAD PLUS
    - Generates standard TIMESTAMP column definitions for the system-begin, system-end, and transaction-start-ID fields, but does not include temporal-specific definitions
    - Does not generate the PERIOD SYSTEM_TIME definition

- When you are unloading application-period temporal data, UNLOAD PLUS generates column definition control cards for business-begin and business-end fields. UNLOAD PLUS generates the PERIOD BUSINESS_TIME definition only when DIRECT YES is in effect.

---

**EASYTRIEVE**

CNTLCARDS EASYTRIEVE tells UNLOAD PLUS to generate Easytrieve data definition statements in the SYSCNTL data set.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.
Restrictions

UNLOAD PLUS terminates if you specify CNTLCARDS EASYTRIEVE when unloading LOB or XML columns.

FOCUS

CNTLCARDS FOCUS tells UNLOAD PLUS to generate FOCUS file definition statements in the SYSCNTL data set.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

Restrictions

UNLOAD PLUS terminates if you specify CNTLCARDS FOCUS when unloading LOB or XML columns.

NOMAD

CNTLCARDS NOMAD tells UNLOAD PLUS to generate NOMAD data definition statements in the SYSCNTL data set.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

Restrictions

UNLOAD PLUS terminates if you specify CNTLCARDS NOMAD when unloading LOB or XML columns.

SAS

CNTLCARDS SAS tells UNLOAD PLUS to generate SAS external file INPUT statements in the SYSCNTL data set.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

Restrictions

UNLOAD PLUS terminates if you specify CNTLCARDS SAS when unloading LOB or XML columns.
SQL/DS

CNTLCARDS SQL/DS tells UNLOAD PLUS to generate SQL/DS CREATE TABLE DDL and SQL/DS DATA LOAD utility control statements in the SYSCNTL data set. UNLOAD PLUS does not generate EDITPROCs or FIELDPROCs.

Restrictions

The following restrictions are specific to CNTLCARDS SQL/DS. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

- UNLOAD PLUS terminates if you specify CNTLCARDS SQL/DS when unloading LOB or XML columns.

- When you use this option, the only valid specifications for NULLTYPE and NULLCHAR are T1 and ? (the defaults), respectively. If you try to use any other combination of NULLTYPE and NULLCHAR values, UNLOAD PLUS issues message BMC51610E and terminates.

SQL/DS-DDL

CNTLCARDS SQL/DS-DDL tells UNLOAD PLUS to generate SQL/DS CREATE TABLE DDL in the SYSCNTL data set. UNLOAD PLUS does not generate EDITPROCs or FIELDPROCs.

Restrictions

The following restrictions are specific to CNTLCARDS SQL/DS-DDL. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

- UNLOAD PLUS terminates if you specify CNTLCARDS SQL/DS-DDL when unloading LOB or XML columns.

- When you use this option, the only valid specifications for NULLTYPE and NULLCHAR are T1 and ? (the defaults), respectively. If you try to use any other combination of NULLTYPE and NULLCHAR values, UNLOAD PLUS issues message BMC51610E and terminates.

SQL/DS-LOAD

CNTLCARDS SQL/DS-LOAD tells UNLOAD PLUS to generate only SQL/DS DATA LOAD utility control statements in the SYSCNTL data set.

Restrictions

The following restrictions are specific to CNTLCARDS SQL/DS-LOAD. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.
• UNLOAD PLUS terminates if you specify CNTLCARDS SQL/DS-LOAD when unloading LOB or XML columns.

• When you use this option, the only valid specifications for NULLTYPE and NULLCHAR are T1 and ? (the defaults), respectively. If you try to use any other combination of NULLTYPE and NULLCHAR values, UNLOAD PLUS issues message BMC51610E and terminates.

TERADATA

CNTLCARDS TERADATA tells UNLOAD PLUS to generate all of the CREATE TABLE DDL and load utility control statements that are included in the related TERADATA options. You must edit the data set to indicate the utility to run.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

Restrictions

UNLOAD PLUS terminates if you specify CNTLCARDS TERADATA when unloading LOB or XML columns.

TERADATA-BULK

CNTLCARDS TERADATA-BULK tells UNLOAD PLUS to generate CREATE TABLE DDL and load utility control statements in the SYSCNTL data set for use with the Teradata Bulk Data Load utility.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

Restrictions

UNLOAD PLUS terminates if you specify CNTLCARDS TERADATA-BULK when unloading LOB or XML columns.

TERADATA-FAST

CNTLCARDS TERADATA-FAST tells UNLOAD PLUS to generate CREATE TABLE DDL and load utility control statements in the SYSCNTL data set for use with the Teradata Fast Data Load utility.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

Restrictions
UNLOAD PLUS terminates if you specify CNTLCARDS TERADATA-FAST when unloading LOB or XML columns.

**TERADATA-MULT**

CNTLCARDS TERADATA-MULT tells UNLOAD PLUS to generate CREATE TABLE DDL and load utility control statements in the SYSCNTL data set for use with the Teradata Multiload utility.

The following restriction is specific to this value. For general considerations for the CNTLCARDS option, see “CNTLCARDS” on page 130.

**Restrictions**

UNLOAD PLUS terminates if you specify CNTLCARDS TERADATA-MULT when unloading LOB or XML columns.

**CNTLDDN**

The CNTLDDN option allows you to override the default ddname or the default ddname prefix of the data set that contains the control statements that UNLOAD PLUS generates.

The default is SYSCNTL. For more information about SYSCNTL data sets, see “SYSCNTL data sets in UNLOAD PLUS” on page 284.

**ORDER**

Specify the ORDER option to tell UNLOAD PLUS whether to sort the output records.

The encoding scheme of the table determines the collating sequence for character data. For example, if the encoding scheme of the table is EBCDIC, the collating sequence is EBCDIC.
The ORDER BY option overrides the ORDER option. Use the ORDER BY option to order by anything other than the data-sorting key.

**NO**

ORDER NO is the default. ORDER NO unloads the output records in the same order in which UNLOAD PLUS found the data rows in the table space.

**YES**

_This option is meaningful only when DIRECT YES is in effect._

ORDER YES sorts the selected rows in clustering key order or, if there is no clustering key, in partitioning key order. If a selected table does not have a clustering or partitioning key, UNLOAD PLUS changes this option to ORDER NO. If you are unloading a multi-table table space, UNLOAD PLUS sorts first by table, then by data-sorting key.

**Note**

To simplify terminology, this book refers to a key that is used to sort data (using ORDER YES) as a _data-sorting key_. For traditional table spaces, this is a clustering key. For table-controlled partitioned table spaces, this is either a clustering key or, if there is no clustering key, a partitioning key.

When multiple tasks unload rows to a single output data set, all tasks execute concurrently to perform the sorting. However, UNLOAD PLUS writes the selected rows in task sequence so that each output process completes before the next starts, unless you specify BYTASK.

**Restrictions**

Note the following restrictions when you specify ORDER YES:

- If you also specify SPANNED YES, UNLOAD PLUS terminates.

- When you are unloading a table-controlled partitioned table space with a clustering index, UNLOAD PLUS uses a single task when either of the following conditions exists:
  - The clustering key does not match, or is not a subset of, the partitioning key.
  - The clustering index is not partitioned and the clustering key is a subset of the partitioning key.

This restriction does not apply when you specify ORDER YES BYTASK.
When you are unloading a partition-by-growth table space, UNLOAD PLUS uses a single task.

When DIRECT NO is in effect, UNLOAD PLUS ignores ORDER YES and does not sort on the data-sorting key.

**BYTASK**

ORDER YES BYTASK orders the selected rows of a table by its data-sorting key within each task. This option is significant only when multiple tasks, executing concurrently, unload rows to a single unload data set. In that case, BYTASK removes serialization from the output process, which enhances performance. All rows that UNLOAD PLUS unloads from a single task are in order, but might be intermixed with other tasks' unloaded rows.

**ENUMROWS**

Specify this option to estimate the total number of rows that BMCSORT processes during the UNLOAD phase.

\[
\text{ENUMROWS} \quad 0 \quad \text{integer}
\]

*This option is meaningful only when DIRECT YES is in effect.*

UNLOAD PLUS uses the ESIZE sort parameter to pass the value to BMCSORT. The default is ENUMROWS 0.

To allow BMCSORT to allocate sort work space dynamically, specify a value that is greater than 0 for ENUMROWS when you are unloading from one of the following sources:

- A full or incremental image copy
- A DSN1COPY sequential data set
- An inline copy
- A VSAM linear data set

*Restriction*

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

*Partitioned table space*
If you are unloading from a partitioned table space, BMC recommends that you do not use ENUMROWS. When UNLOAD PLUS multitasks the sorting of a partitioned table space, UNLOAD PLUS divides the ENUMROWS option by the maximum number of concurrent tasks.

**Multiple table spaces**

When UNLOAD PLUS multitasks the sorting of multiple table spaces, the number that you specify with ENUMROWS applies to each of those table spaces. To unload multiple table spaces, therefore, base your estimate on the table space that contains the greatest number of rows that UNLOAD PLUS will sort.

**SORTDEVT**

The SORTDEVT option specifies the device type for the sort work files that BMCSORT will allocate dynamically.

```
SORTDEVT deviceType
```

*This option is meaningful only when DIRECT YES is in effect.*

This option overrides the value specified in the BMCSORT DYNALOC installation option.

If the value of the third parameter in the BMCSORT DYNALOC installation option is OFF, specifying the SORTDEVT option turns BMCSORT dynamic allocation on.

**Restriction**

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**SORTNUM**

The SORTNUM option affects the number of sort work files allocated when BMCSORT is allocating your sort work files dynamically.

```
SORTNUM integer
```

*This option is meaningful only when DIRECT YES is in effect.*

You can specify any integer value from 0 through 255.
The following table describes the action that BMCSORT takes for each value that you can specify for this option. The table also provides any additional considerations for these values.

### Table 22: SORTNUM values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Additional considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BMCSORT honors the value of the third parameter of the BMCSORT DYNALOC installation option. (This parameter tells BMCSORT whether to dynamically allocate sort work files.)</td>
<td>For more information about how this parameter affects dynamic allocation, see “Dynamically allocating SORTWK data sets” on page 281. For more information about this parameter, see “DYNALOC installation option” on page 463.</td>
</tr>
<tr>
<td>1-32</td>
<td>BMCSORT dynamically allocates the number of sort work files that it needs, up to 32 minus any sort work files that are allocated in your JCL. This number is per task</td>
<td>None</td>
</tr>
<tr>
<td>33-255</td>
<td>BMCSORT dynamically allocates the number of sort work files that it needs, up to the number that you specify minus any sort work files that are allocated in your JCL. This number is per task.</td>
<td>In this case, BMC strongly recommends that you change the second value of the SMCORE installation option to 384K.</td>
</tr>
</tbody>
</table>

If the value of the third parameter in the BMCSORT DYNALOC installation option is OFF, specifying a value greater than 0 for the SORTNUM option turns BMCSORT dynamic allocation on, and BMCSORT allocates sort work files as needed.

The sort work files that BMCSORT allocates dynamically might be used in one of the following circumstances:

- If you do not specify any SORTWK DD statements in your JCL
- In addition to any SORTWK data sets that you specify in your JCL if BMCSORT determines that there is a need for additional data sets

For information about circumstances under which BMCSORT allocates your sort work files dynamically, see “SORTWK data sets in UNLOAD PLUS” on page 281.

If you depend on BMCSORT to dynamically allocate your sort work data sets, BMC recommends that you also specify the ENUMROWS option.

**Restriction**

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**
You can specify the default for the SORTNUM command option in your installation options module by using the SORTNUM installation option. UNLOAD PLUS was shipped with a value of 32. The command option overrides the default that is in the installation options module.

**LIMIT**

The LIMIT option defines the limit on the number of rows that UNLOAD PLUS selects from each table or partition of the table space.

```plaintext
LIMIT integer
```

The default is LIMIT 0 (no limit). If UNLOAD PLUS reaches the limit, it ends normally.

**Additional information**

Note the following additional information about this option:

- When unloading from a partitioned table space using DIRECT NO, LIMIT functions as if the table space were nonpartitioned.

- When you are dynamically allocating output data sets, you must also specify the LIMIT, ESTROWS, or SPACE command option in either of the following cases:
  - DIRECT NO is in effect.
  - You specify the INFILE option.

When you are dynamically allocating output data sets, the LIMIT keyword tells UNLOAD PLUS to derive the total number of records per data set and calculate the data set size and space needed for allocating the data sets.

**INTERVAL**

The INTERVAL option allows you to unload a sample of data, rather than all rows that meet the selection criteria.

```plaintext
INTERVAL integer
```
The default is INTERVAL 0. The INTERVAL integer defines the interval of the selected rows that UNLOAD PLUS unloads. For example, INTERVAL 10 means that UNLOAD PLUS unloads every tenth row that it selects from each table or partition of the table space.

**Note**
When unloading from a partitioned table space using DIRECT NO, INTERVAL functions as if the table space were nonpartitioned.

**DISCARDS**

Specify the DISCARDS option to define the limit on the number of records that UNLOAD PLUS discards.

The default is DISCARDS 0 (no limit).

Records that UNLOAD PLUS discards for any reason count toward the limit that you specify with the DISCARDS option. UNLOAD PLUS terminates if it reaches the discard limit.

UNLOAD PLUS discards records that have the following types of errors:

- Conversion errors
- Rejected records based on a user-written exit routine
- SYSREC records truncated due to the length of LOB or XML data

UNLOAD PLUS does not write the discarded records to any data set.

For related details about the role that the DISCARDS option plays during character conversion when UNLOAD PLUS is unloading fields that result in SUBBYTE AND ERRORBYTE processing, see “SUBBYTE and ERRORBYTE” on page 265.

**AUTOTAG**

Specify the AUTOTAG option to tell UNLOAD PLUS whether to add a four-byte character constant value at the beginning of each output record and to generate appropriate load control statements.
When you specify CNTLCARDS BMCLOAD and the following conditions exist, specify AUTOTAG YES:

- You are using a single SYSCNTL data set.
- You specify multiple SELECT statements on tables that are in the same table space.

**Note**
You cannot specify AUTOTAG YES if you also specify FORMAT BMCLOAD.

**NO**
AUTOTAG NO is the default. If you specify AUTOTAG NO, UNLOAD PLUS does not add a value to the beginning of each output record for each SELECT statement. When generating load control statements, UNLOAD PLUS generates syntax for multiple LOAD statements, generating one LOAD for each SELECT statement.

**YES**
If you specify AUTOTAG YES, UNLOAD PLUS adds a four-byte value to the beginning of each output record for each SELECT statement, and increments the value for each successive SELECT statement. For example, the value for the first SELECT statement is 0001; the value of a second SELECT statement would be 0002.

When generating load control statements, UNLOAD PLUS

- Generates syntax for one LOAD statement with multiple INTO clauses, generating one INTO clause for each SELECT statement
- Generates appropriate WHEN syntax

**Restrictions**
The following restrictions apply to the AUTOTAG YES option:

- If you also specify FORMAT BMCLOAD, UNLOAD PLUS terminates.
- If you specify either of the following options, UNLOAD PLUS ignores the AUTOTAG option:
MAXSORTS

Use this option to specify the maximum number of tasks that UNLOAD PLUS can execute concurrently.

This option is meaningful only when executing multiple tasks to unload a partitioned table space or multiple table spaces.

Restricting the number of concurrent tasks is critical when BMCSORT is invoked, because sorting consumes a significant amount of system resources.

**WARNING**

Too many concurrent tasks to a single output data set can degrade performance. When you use a single SELECT statement to unload a partitioned table space, or when you are unloading from multiple table spaces, use multiple output data sets or use the MAXSORTS option to limit the number of concurrent tasks and reduce contention. For the ramifications of using the MAXSORTS option, see “Performance considerations for sort processing options in UNLOAD PLUS” on page 405.

**Considerations**

The following considerations apply to the MAXSORTS option:

- When unloading to a BatchPipes file, you must set MAXSORTS to 1.
- UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**

You can specify the default for the MAXSORTS command option in your installation options module by using the SMAX installation option. UNLOAD PLUS was shipped with a default value of 16 for this option. The command option overrides the default that is in the installation options module.
MAXCONNECT

MAXCONNECT specifies the maximum number of tasks that UNLOAD PLUS can execute concurrently when unloading data using DB2 dynamic SQL.

This option is meaningful only when DIRECT NO is in effect.

When unloading data using DB2 dynamic SQL, UNLOAD PLUS executes one SELECT statement per task. Each task uses a batch thread to process a SELECT statement. You can use MAXCONNECT to limit the number of batch threads that UNLOAD PLUS uses during the UNLOAD phase.

Considerations

The following considerations apply to MAXCONNECT:

- When unloading to a BatchPipes file, you must set MAXCONNECT to 1.
- UNLOAD PLUS ignores this option when DIRECT YES is in effect.

Specifying the default

You can specify the default for the MAXCONNECT command option in your installation options module by using the CMAX installation option. UNLOAD PLUS was shipped with a default value of 16 for this option. The command option overrides the default that is in the installation options module.

SYNC

Specify an integer with the SYNC option to identify the number of 1-KB rows that you want UNLOAD PLUS to process between writing to the BMCSYNC table.

This option is meaningful only when DIRECT YES is in effect.

UNLOAD PLUS writes records to the BMCSYNC table that show the number of 1-KB rows that UNLOAD PLUS processed during the UNLOAD phase. You can use this information to determine how far the UNLOAD job has progressed. By default,
UNLOAD PLUS writes records only after the last row is read or unloaded from a table space.

UNLOAD PLUS records monitoring sync points in the BMCSYNC table as the job progresses. You can issue an SQL statement to query this table to determine how far the unload has progressed and the status of objects that the utility is unloading.

Restriction

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

FORMAT

The FORMAT option allows you to specify the format of unloaded data. All SELECT statements of all rows use the specified format.
You can override the specified format for a field by specifying an explicit format for the field. For more information, see “field specification” on page 223.

For more information about using the different output formats and the restrictions associated with each option, see “UNLOAD PLUS output format” on page 64.

**STANDARD**

STANDARD (the default) tells UNLOAD PLUS to unload data of all data types in internal format except for the date, time, and timestamp data types, which it unloads in external format.

*Note*  
To produce unloaded data that matches the IBM DSNTIAUL program, specify FORMAT DSNTIAUL.

**EXTERNAL**

EXTERNAL tells UNLOAD PLUS to unload data for all data types in external format. If you do not specify a data type, UNLOAD PLUS uses the default length.

UNLOAD PLUS terminates if you specify this option when you are unloading LOB or XML data.

**INTERNAL**

INTERNAL tells UNLOAD PLUS to unload data in internal format. This option applies mainly to the date, time, and timestamp data types and is useful only if you subsequently use LOADPLUS to load the data. UNLOAD PLUS does not invoke date and time exit routines.

For restrictions associated with this option, see “DB2 output with internal date and time (FORMAT INTERNAL)” on page 67.

**BMCLOAD**

*This option is valid only when DIRECT YES is in effect.*

BMCLOAD produces a high speed unload and an output file that only LOADPLUS can use when you are moving data to tables with a nearly identical table structure. This option is useful for migrating identical data to duplicate tables or development databases.

Using FORMAT BMCLOAD instead of standard processing may help reduce CPU cycles and elapsed time. However, complicated table structures can cause variations in the CPU cycles and elapsed time.
For additional information, including restrictions associated with this option, see “Data migration using LOADPLUS (FORMAT BMCLOAD)” on page 67.

**DSNTIAUL**

DSNTIAUL allows UNLOAD PLUS to use its high-performance direct access to produce an output file in which the unloaded data matches the output of the IBM DSNTIAUL program. For additional information, including restrictions associated with using this option, see “DSNTIAUL-formatted output (FORMAT DSNTIAUL)” on page 71.

**CSV**

CSV allows you to output your data in a comma-separated-value (CSV) format that enables porting to other databases and applications. When you specify FORMAT CSV, UNLOAD PLUS formats the unloaded data so that a specific character separates each field from other fields, and encloses each nonnumeric field with a specific pair of characters.

For additional information, including restrictions associated with using this option, see “CSV output (FORMAT CSV)” on page 72.

CSV options are TERMINATEDBY, ENCLOSEDBY, AND, NULLSTRING, and RTRIM. When you use more than one option, the syntax must follow the sequence in the syntax diagram. For example, if you use TERMINATEDBY, that option must appear before ENCLOSEDBY or NULLSTRING.

**TERMINATEDBY**

TERMINATEDBY specifies a single character that UNLOAD PLUS uses to delimit fields of data. The default is a comma (,). You cannot specify a null value for TERMINATEDBY.

**ENCLOSEDBY**

ENCLOSEDBY specifies a single character that UNLOAD PLUS uses on the left side to enclose fields of data. The default is a double quotation mark ("'). You can use an empty string to specify a null value for this option.

**AND**

This option specifies a single character that UNLOAD PLUS uses on the right side to enclose fields of data. If you do not specify a value for this option, UNLOAD PLUS uses the value that you specified for the ENCLOSEDBY option or a double quotation mark if you did not specify a value for the ENCLOSEDBY option. You can use an empty string to specify a null value for this option.
NULLSTRING

NULLSTRING specifies up to a four-character string that indicates a null value in the output file. You can specify NULLSTRING as an empty string. The default is NULL.

RTRIM

RTRIM tells UNLOAD PLUS to strip trailing blanks from input character strings. With the exceptions described in the following restrictions, UNLOAD PLUS performs this operation for all CHAR and VARCHAR columns that you are unloading.

Restrictions

UNLOAD PLUS ignores the RTRIM option for a column when any of the following conditions exists:

- The column is defined as a data type other than CHAR or VARCHAR.
- The column’s field specification includes a length and the length differs from the actual length of the data.
- The column’s field specification includes any of the following keywords:
  - MIXED
  - TRIM
  - TRUNCATE
  - FILL

XML

XML allows you to unload your data in a format that enables porting to other databases and applications. When you specify the XML option, UNLOAD PLUS formats the unloaded data so that each field is enclosed by a specific pair of XML tags corresponding to the unloaded column name.

For additional information, including restrictions associated with using this option, see “XML output (FORMAT XML)” on page 74.

XML options are SELECT_ELEMENT, ENCLOSEDBY, AND, and NULLSTRING. When you use more than one option, the syntax must follow the sequence in the syntax diagram. For example, if you specify SELECT_ELEMENT, that option must appear before ENCLOSEDBY or NULLSTRING.
SELECT_ELEMENT

Specify SELECT_ELEMENT to identify characteristics of the XML tags that enclose each unloaded row.

AUTO

Specify AUTO to enclose each unloaded record with an XML tag. UNLOAD PLUS assigns a value to the tag name based on the following criteria listed in the order of precedence:

- The value of the INTO NAME option on the SELECT statement (if specified)
- A generated table name (if you specify DIRECT NO processing)
- The name of the table from which the unloaded row was extracted

The following examples illustrate the unloaded data when specifying AUTO with different options:

Table 23: Specifying AUTO with DIRECT YES, DIRECT NO, and INTO NAME

<table>
<thead>
<tr>
<th>Option</th>
<th>UNLOAD command statement</th>
<th>Unloaded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT YES</td>
<td>DIRECT YES FORMAT XML SELECT_ELEMENT AUTO SELECT NAME, ADDRESS, CITY, STATE FROM MYCONTACTS;</td>
<td>&lt;CONTACTS&gt;&lt;NAME&gt;&quot;BMC Software, Inc.&quot;&lt;/NAME&gt;&lt;ADDRESS&gt;&quot;10431 MORADO CIRCLE&quot;&lt;/ADDRESS&gt;&lt;CITY&gt;&quot;AUSTIN&quot;&lt;/CITY&gt;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;&lt;/CONTACTS&gt;</td>
</tr>
<tr>
<td>DIRECT NO</td>
<td>DIRECT NO FORMAT XML SELECT_ELEMENT AUTO SELECT NAME, ADDRESS, CITY, STATE FROM MYCONTACTS;</td>
<td>&lt;$TABLE1&gt;&lt;NAME&gt;&quot;BMC Software, Inc.&quot;&lt;/NAME&gt;&lt;ADDRESS&gt;&quot;10431 MORADO CIRCLE&quot;&lt;/ADDRESS&gt;&lt;CITY&gt;&quot;AUSTIN&quot;&lt;/CITY&gt;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;&lt;/$TABLE1&gt;</td>
</tr>
<tr>
<td>INTO NAME</td>
<td>DIRECT NO FORMAT XML SELECT_ELEMENT AUTO SELECT NAME, ADDRESS, CITY, STATE INTO NAME &quot;CUSTOMER&quot; FROM MYCONTACTS;</td>
<td>&lt;CUSTOMER&gt;&lt;NAME&gt;&quot;BMC Software, Inc.&quot;&lt;/NAME&gt;&lt;ADDRESS&gt;&quot;10431 MORADO CIRCLE&quot;&lt;/ADDRESS&gt;&lt;CITY&gt;&quot;AUSTIN&quot;&lt;/CITY&gt;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;&lt;/CUSTOMER&gt;</td>
</tr>
</tbody>
</table>

NONE

Specify NONE if you do not want to enclose each unloaded record with an XML tag. UNLOAD PLUS still encloses each individual field by the appropriate XML tag (for example, <COL1>data</COL1>). The following example illustrates the unloaded data when specifying NONE with DIRECT YES processing:

Chapter 3 Syntax of the UNLOAD command 153
Table 24: Unloaded data when specifying NONE with DIRECT YES

<table>
<thead>
<tr>
<th>UNLOAD command statement</th>
<th>Unloaded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT YES</td>
<td></td>
</tr>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>SELECT_ELEMENT NONE</td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td></td>
</tr>
<tr>
<td>NAME, ADDRESS, CITY, STATE</td>
<td>&lt;NAME&gt;“BMC Software, Inc.”&lt;/NAME&gt;&lt;ADDRESS&gt;“10431 MORADO CIRCLE”&lt;/ADDRESS&gt;&lt;CITY&gt;“AUSTIN”&lt;/CITY&gt;&lt;STATE&gt;“TX”&lt;/STATE&gt;</td>
</tr>
</tbody>
</table>

‘string’

Specify a name to identify the name of the XML tag with which you want UNLOAD PLUS to enclose each unloaded record.

Note

UNLOAD PLUS does not validate the string value for conformance to any XML standard.

The following example illustrates the unloaded data when you specify the string value ‘TRANSACTION’ with DIRECT YES processing:

Table 25: Unloaded data when specifying a string value with DIRECT YES

<table>
<thead>
<tr>
<th>UNLOAD command statement</th>
<th>Unloaded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT YES</td>
<td></td>
</tr>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>SELECT_ELEMENT ‘TRANSACTION’</td>
<td>&lt;TRANSACTION&gt;&lt;NAME&gt;“BMC Software, Inc.”&lt;/NAME&gt;&lt;ADDRESS&gt;“10431 MORADO CIRCLE”&lt;/ADDRESS&gt;&lt;CITY&gt;“AUSTIN”&lt;/CITY&gt;&lt;STATE&gt;“TX”&lt;/STATE&gt;&lt;/TRANSACTION&gt;</td>
</tr>
</tbody>
</table>

ENCLOSEDBY

ENCLOSEDBY specifies the character that UNLOAD PLUS uses on the left side to enclose fields of data. The default is a double quotation mark ("). You can use an empty string to specify a null value for this option.

AND

This option specifies the character that UNLOAD PLUS uses on the right side to enclose fields of data. If you do not specify a value for this option, UNLOAD PLUS uses the value that you specified for the ENCLOSEDBY option or a double quotation mark if you did not specify a value for the ENCLOSEDBY option. You can use an empty string to specify a null value for this option.
NULLSTRING

NULLSTRING specifies up to a four-character string that indicates a null value in the output file. You can specify NULLSTRING as an empty string. The default is NULL.

NULLCHAR

This option allows you to change the null indicator character in the output record.

With the exception of the HIVAL option, the values that you specify for the NULLCHAR option signify only a single byte. If the null indicator field is two bytes long, UNLOAD PLUS propagates the constant to the second byte. The following table describes the values that you can specify for this option:

Table 26: Values for the NULLCHAR option

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| ?     | This value tells UNLOAD PLUS to fill the null indicator field with the question mark character, '?'.
| HIVAL | This value tells UNLOAD PLUS to fill the null indicator field with high values. |
| 'c'   | This value tells UNLOAD PLUS to fill the null indicator field with the character c, where c is any constant that is valid for C-type assembler language. |
| X'xx' | This value tells UNLOAD PLUS to fill the null indicator field with xx, where xx is any valid hexadecimal assembler constant, except X'00'. |

Restrictions

Note the following restrictions when using the NULLCHAR option:

- If you specify FORMAT DSNTIAUL, UNLOAD PLUS overrides this value to '?'.
- UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.
- When you are using the CNTLCARDS SQL/DS, SQL/DS-DDL, or SQL/DS-LOAD options, the only valid specification for NULLCHAR is '. If you
try to use any other NULLCHAR value, UNLOAD PLUS issues message BMC51610E and terminates.

**Specifying the default**

You can specify the default for the NULLCHAR command option in your installation options module by using the NULLCHAR installation option. UNLOAD PLUS was shipped with a default value of ? for this option. The command option overrides the default that is in the installation options module.

**NULLTYPE**

This option allows you to specify the location and length of the null indicator field in the output record.

The following table describes the values that you can specify for this option:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Specifies that the null indicator is 1 byte long, following the column.</td>
</tr>
<tr>
<td>T2</td>
<td>Specifies that the null indicator is 2 bytes long, following the column.</td>
</tr>
<tr>
<td>L1</td>
<td>Specifies that the null indicator is 1 byte long, preceding the column.</td>
</tr>
<tr>
<td>L2</td>
<td>Specifies that the null indicator is 2 bytes long, preceding the column.</td>
</tr>
</tbody>
</table>

**Restrictions**

Note the following restrictions when using the NULLTYPE option:

- If you specify FORMAT DSNTIAUL, UNLOAD PLUS sets NULLTYPE to T1.
- If you specify SPANNED YES, UNLOAD PLUS changes an L NULLTYPE value to its comparable T value.
- UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD or FORMAT XML.
- When you are using the CNTLCARDS SQL/DS, SQL/DS-DDL, or SQL/DS-LOAD options, the only valid specification for NULLTYPE is T1. If you try to use any other NULLTYPE value, UNLOAD PLUS issues message BMC51610E and terminates.

**Specifying the default**

You can specify the default for the NULLTYPE command option in your installation options module by using the NULLTYPE installation option. UNLOAD PLUS was shipped with a default value of T1 for this option. The command option overrides the default that is in the installation options module.

**FILL**

The FILL option tells UNLOAD PLUS whether to fill numeric external fields with zeros when converting numeric columns to their external representations.

- **NO**
  
  Specify NO if you want to remove all leading zeros when converting numeric data types to their external representations.

- **YES**
  
  When you specify FILL YES, UNLOAD PLUS externalizes all leading zeros when converting numeric data types to their external representations. The first character in...
the external representation of the number is either a minus sign ('-') or a plus sign ('+').

**EBCDIC**

This option specifies that UNLOAD PLUS use the DB2 installation default (DSNHDECP) CCSIDs (coded character set identifiers) to encode the unloaded data in EBCDIC. This option is the default for the three encoding types.

For information about data translation, see “Data translation in UNLOAD PLUS” on page 261.

**Restriction**

UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.

**ASCII**

This option specifies that UNLOAD PLUS use the DB2 installation default (DSNHDECP) CCSIDs to encode the unloaded data in ASCII.

For information about data translation, see “Data translation in UNLOAD PLUS” on page 261.

**Restriction**

UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.

**UNICODE**

This option specifies that UNLOAD PLUS use the DB2 installation default (DSNHDECP) CCSIDs to encode the unloaded data in Unicode.
For information about data translation, see “Data translation in UNLOAD PLUS” on page 261.

For an example of unloading data from an EBCDIC table space using a Unicode encoding scheme (specified using the UNICODE keyword), see “Example 18 - Unloading a table space and unloading data as Unicode” on page 381.

Restrictions

Note the following restrictions when using the UNICODE option:

- When DIRECT YES is in effect, UNLOAD PLUS supports only UTF-8 (CCSID 1208) encoding.
- UNLOAD PLUS terminates if you also specify any of the following options:
  - When DIRECT YES is in effect, FORMAT XML
  - When DIRECT NO is in effect:
    - FORMAT BMCLOAD
    - FORMAT DSNTIAUL
    - FORMAT INTERNAL
    - FORMAT XML

CCSID

This option specifies that UNLOAD PLUS use up to three CCSIDs to encode the unloaded data.

The three CCSIDs are for single-byte character set (SBCS), MIXED data, and double-byte character set data (DBCS) respectively. If you omit a CCSID value or specify 0, UNLOAD PLUS uses the corresponding DB2 system default CCSID for the encoding.
scheme that you specify or default to (EBCDIC, ASCII, or UNICODE). For more
details, see “Data translation in UNLOAD PLUS” on page 261.

BMC recommends that you use this option only to override your DB2 system default
CCSID values.

Restrictions

UNLOAD PLUS supports translation from one CCSID to another with the following
restrictions:

■ Translations that require the use of a conversion procedure are not supported.

■ UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.

■ If you also specify the UNICODE option when DIRECT YES is in effect, UNLOAD
PLUS ignores any CCSID specification.

NOSUBS

Specify NOSUBS if UNLOAD PLUS should refuse to accept substitution characters
during translation between CCSIDs.

This option is meaningful only when DIRECT YES is in effect.

If you specify NOSUBS and UNLOAD PLUS encounters a record that requires
substitution, UNLOAD PLUS discards the record rather than unloading it.

Restrictions

UNLOAD PLUS ignores the option if you also specify FORMAT BMCLOAD or
DIRECT NO.

DATEFMT

The DATEFMT option allows you to override the default DB2 external data format.

This option also overrides your DB2 date exit routine. You can use this option to port
unloaded data to other relational databases.
**Restriction**

UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.

(`string`)  

This option allows you to specify a string of date format elements. Valid values are:

- MM for month
- DD for day
- YY or YYYY for year

You can specify these values in any order and delimit them with any character or number, or any series of characters or numbers, although UNLOAD PLUS does not require a delimiter. UNLOAD PLUS treats any other characters as constants in their specified position.

**Requirements**

The following requirements apply to the string that you specify with the DATEFMT option:

- The string must begin with one of the valid date format elements.
- Only one occurrence of each value is valid.
- If you specify this option, you must specify at least one value or the unload job terminates.

**TIMEFMT**

The TIMEFMT option allows you to override the default DB2 external time format.

This option also overrides your DB2 time exit routine. This option is useful for porting unloaded data to other relational databases.

**Restriction**

UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.
This option allows you to specify a string of time format elements. Valid values are:

- HH for hour
- MM for minute
- SS for second
- XM for AM or PM

You can specify these values in any order and delimit them with any character or number, or any series of characters or numbers, although UNLOAD PLUS does not require a delimiter.

Requirements

The following requirements apply to the string that you specify with the TIMEFMT option:

- The string must begin with one of the valid time format elements.
- Only one occurrence of each value is valid.
- If you specify this option, you must specify at least one value or the unload job terminates.

Additional considerations

The following considerations apply to this option:

- If you specify XM, UNLOAD PLUS converts the HH element to a 12-hour format.
- UNLOAD PLUS treats any other characters as constants in their specified position.

TSFMT

The TSFMT option allows you to override the default DB2 external timestamp format.

This option is useful for porting unloaded data to other relational databases.
Restriction

UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.

(string)

This option allows you to specify a string of timestamp format elements. Valid values are

- DD for day
- MM or MO for month
- YY or YYYY for year
- HH for hour
- MM or MI for minute
- SS for second
- NN, NNNN, or NNNNNN for microsecond
- XM for AM or PM

You can specify these values in any order and delimit them with any character or number, or any series of characters or numbers, although UNLOAD PLUS does not require a delimiter.

Requirements

The following requirements apply to the string that you specify with the TSFMT option:

- The string must begin with one of the valid timestamp format elements.
- Only one occurrence of each value is valid.
- If you specify this option, you must specify at least one value or the unload job terminates.

Additional considerations

The following considerations apply to this option:

- You can specify MM for both months and minutes within one TSFMT option. UNLOAD PLUS determines whether MM designates months or minutes based on the position of MM within the string. If UNLOAD PLUS cannot determine
whether MM designates months or minutes within the format, UNLOAD PLUS terminates.

- If you specify XM, UNLOAD PLUS converts the HH element to a 12-hour format.
- UNLOAD PLUS treats any other characters as constants in their specified position.

**IMPLICIT_TZ**

The IMPLICIT_TZ option enables you to specify a time zone to use in certain situations.

```
IMPLICIT_TZ 'timeZoneString'
```

UNLOAD PLUS uses this time zone for the following specifications:

- You are unloading a TIMESTAMP column (without a time zone) and the field specification for the column is TIMESTAMP WITH TIME ZONE.

- A SELECT statement has the following characteristics:
  - The field specification includes a TIMESTAMP WITH TIME ZONE field.
  - You are updating the field by specifying a timestamp constant that does not include a time zone.

- A WHERE clause has the following characteristics:
  - The predicate is comparing values to a TIMESTAMP WITH TIME ZONE column.
  - You specify a constant that does not include a time zone.

UNLOAD PLUS adds the time zone that you specify to the timestamp value.

Specify a time zone, within quotation marks, as the difference between local time and Coordinated Universal Time (UTC). You can specify a value from -12:59 through +14:00. If you do not specify a value for the IMPLICIT_TZ option, UNLOAD PLUS uses the DSNHDECP IMPLICIT_TIMEZONE value.

Any time zone that you specify in a SELECT statement overrides the value of this option.
**ZONEDDECOVP**

Traditionally, UNLOAD PLUS formats decimal-zoned values with the system default zone digit, which is C for positive values and D for negative values. You can specify the ZONEDDECOVP option to assign overpunch values to decimal-zoned numeric values.

The overpunched values can be positive or negative.

- The first operand specifies the zone overpunch value for positive numbers. Standard positive overpunch values are A, C, E, and F.
- The second operand specifies the zone overpunch value for negative numbers. Standard negative overpunch values are B and D.

This option can be useful when you plan to use the unloaded data with an application that requires a specific zone value that is not the traditional default.

To obtain the absolute value of a number or to ensure that the value is in data external format, you can specify ZONEDDECOVP (F,F).

For an example of assigned overpunch values using the ZONEDDECOVP option, see “Example 17 - Changing overpunch values in zoned decimal data” on page 377.

**Specifying the default**

You can specify the default for the ZONEDECOVP command option in your installation options module by using the ZONEDECOVP installation option. UNLOAD PLUS was shipped with a default value of C,D for this option. The command option overrides the default that is in the installation options module.

**DECFLOAT_ROUNDMODE**

The DECFLOAT_ROUNDMODE option tells UNLOAD PLUS how to round values in any DECFLOAT fields in your output.
If you do not specify a value, the utility uses the DECFLOAT round mode value from your DSNHDECP settings.

**ROUND_CEILING**

ROUND_CEILING rounds toward positive infinity. If the extra digits are all zero or the number is negative, the utility truncates the digits. Otherwise, the utility rounds the remaining value up (increments the rightmost digit by 1).

**ROUND_DOWN**

ROUND_DOWN truncates the extra digits (rounds toward 0).

**ROUND_FLOOR**

ROUND_FLOOR rounds toward negative infinity. If the extra digits are all zero or the number is positive, the utility truncates the digits. Otherwise, the remaining value is negative and the utility rounds up (increments the rightmost digit by 1).

**ROUND_HALF_DOWN**

ROUND_HALF_DOWN rounds to the nearest number as explained in the following table:

Table 28: ROUND_HALF_DOWN rounding

<table>
<thead>
<tr>
<th>Extra digits</th>
<th>Utility action</th>
</tr>
</thead>
<tbody>
<tr>
<td>.5 or less</td>
<td>Truncates the digits (rounds the remaining value down)</td>
</tr>
<tr>
<td>Greater than .5</td>
<td>Rounds the remaining value up (increments the rightmost digit by 1)</td>
</tr>
</tbody>
</table>
ROUND_HALF_EVEN

ROUND_HALF_EVEN rounds to the nearest number as explained in the following table:

<table>
<thead>
<tr>
<th>Extra digits</th>
<th>Rightmost digit of remaining value</th>
<th>Utility action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than .5</td>
<td>Not applicable</td>
<td>Rounds the remaining value up (increments the rightmost digit by 1)</td>
</tr>
<tr>
<td>Less than .5</td>
<td>Not applicable</td>
<td>Truncates the digits (rounds the remaining value down)</td>
</tr>
<tr>
<td>Equal to .5</td>
<td>Odd</td>
<td>Rounds the remaining value up (increments the rightmost digit by 1)</td>
</tr>
<tr>
<td>Equal to .5</td>
<td>Even</td>
<td>Truncates the digits (rounds the remaining value down)</td>
</tr>
</tbody>
</table>

ROUND_HALF_UP

ROUND_HALF_UP rounds to the nearest number as explained in the following table:

<table>
<thead>
<tr>
<th>Extra digits</th>
<th>Utility action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than .5</td>
<td>Truncates the digits (rounds the remaining value down)</td>
</tr>
<tr>
<td>.5 or greater</td>
<td>Rounds the remaining value up (increments the rightmost digit by 1)</td>
</tr>
</tbody>
</table>

ROUND_UP

ROUND_UP rounds away from 0. If the extra digits are all zero, the utility truncates the digits. Otherwise, the utility rounds the remaining value up (increments the rightmost digit by 1).

FIXEDVARCHAR

The FIXEDVARCHAR option allows you to specify whether to unload VARCHAR, VARGRAPHIC, and VARBINARY column values as variable-length or fixed-length fields when you do not explicitly specify a field type.
In either case, the field type is still VARCHAR, VARGRAPHIC, or VARBINARY. If you want any other field type, specify CHARACTER( n) as the field data type.

NO

FIXEDVARCHAR NO is the default and indicates that VARCHAR, VARGRAPHIC, and VARBINARY columns are unloaded as variable-length fields without any padding before the beginning of the next field.

Restriction

If you specify FORMAT DSNTIAUL, UNLOAD PLUS overrides FIXEDVARCHAR NO to YES.

YES

When you specify FIXEDVARCHAR YES, UNLOAD PLUS performs the following actions:

- Unloads VARCHAR, VARGRAPHIC, and VARBINARY columns as fixed-length VARCHAR, VARGRAPHIC, and VARBINARY fields (unless you explicitly specify a field type)

  A fixed-length VARCHAR, VARGRAPHIC, or VARBINARY field has a length equal to the maximum length of the column + 2. The actual length specification of the column remains unchanged.

- Sets any unused bytes in the field (after the value and before the beginning of the next field) to blanks (VARCHAR and VARGRAPHIC) or hexadecimal zeros (VARBINARY)

- Sets the record format for the unload data set to FB

  Exception: If you specify RECFM VB on your UNLOAD command (or it is in effect by default), UNLOAD PLUS uses variable block format for the data set records.

Restrictions

Note the following restrictions when specifying FIXEDVARCHAR YES:

- UNLOAD PLUS does not change the 2-byte length indicator when you specify FIXEDVARCHAR YES. If you plan to load the unloaded data, use the control
statements that UNLOAD PLUS generates to ensure proper table loading. Otherwise, UNLOAD PLUS might load incorrect or truncated data.

- If you specify FIXEDVARCHAR YES with FORMAT BMCLOAD, UNLOAD PLUS terminates.

- If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for either of the following types of output data sets:
  
  - Referenced files
  
  - VBS data sets when you specify SPANNED YES

**UNLOADEXIT**

The UNLOADEXIT option allows you to specify the name and programming language of a user-written exit routine.

```
UNLOADEXIT exitName
```

- ASM
- COBOL2
- LE_COBOL
- C
- LE_C

**Restriction**

UNLOAD PLUS ignores this option if you also specify FORMAT BMCLOAD.

**exitName**

UNLOAD PLUS passes every output record to the specified exit before writing the record to the unload data set. If UNLOAD PLUS performs sorting, the utility passes the record to the exit routine before the sort process, but after building the sort key. UNLOAD PLUS invokes BMCSORT after the user exit when you specify the ORDER YES or ORDER BY options. The exit routine can inspect, modify, or discard records before UNLOAD PLUS writes the output records to the unload data set.

For information about user-defined variables, and requirements and restrictions when creating an exit routine, see “UNLOAD PLUS user exits” on page 531.
program language

UNLOAD PLUS supports user-written exit routines in assembler, C, LE C, COBOL II, and LE COBOL. After you specify the exit name, specify the program language of the exit routine as shown in Table 31 on page 170.

For user exits in any program language except ASM, UNLOAD PLUS sets MAXSORTS to 1.

Table 31: Supported program languages for user exits

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>(default) Specifies an assembler language program</td>
</tr>
<tr>
<td>COBOL2</td>
<td>Specifies a COBOL II language program</td>
</tr>
<tr>
<td>LE_COBOL</td>
<td>Specifies an LE COBOL language program</td>
</tr>
<tr>
<td>C</td>
<td>Specifies a C language program</td>
</tr>
<tr>
<td>LE_C</td>
<td>Specifies an LE C language program</td>
</tr>
</tbody>
</table>

ON MESSAGE

ON MESSAGE allows you to specify whether UNLOAD PLUS should stop or continue processing when it encounters message BMC50253 or BMC50254.

This option is meaningful only when DIRECT YES is in effect.

If you do not specify ON MESSAGE, the utility stops when it encounters the specified message.

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

STOP UTILITY

STOP UTILITY is the default, and stops the job when UNLOAD PLUS issues either BMC50253 or BMC50254. The utility ID remains in a restartable state.
CONTINUE UTILITY

This option tells UNLOAD PLUS to continue processing when issuing either BMC50253 or BMC50254. However, under some fatal conditions, the utility does not continue, even though you specified CONTINUE. When you specify CONTINUE and UNLOAD PLUS can continue, UNLOAD PLUS issues an informational message.

BMC recommends that you specify CONTINUE only after an error stops your utility job, and you either correct the problem and restart or determine that you want to restart without making any changes.

RETCODE

This option tells UNLOAD PLUS which return code to issue when it encounters message BMC50253 or BMC50254 with CONTINUE UTILITY.

The following considerations apply to this option:

- BMC recommends that you avoid specifying a value of 0 or 4.
- If you do not specify a value for this option, UNLOAD PLUS issues the normal return code.

ON FAILURE

ON FAILURE tells UNLOAD PLUS how you want to handle an abnormal termination (when your job ends with a return code that is greater than or equal to 8, or abends).

For each phase, you can specify a particular return code to use when a failure occurs in that phase. Alternatively, you can specify ALL to indicate what return code to use when a failure occurs in any of the execution phases. For example, the following specification indicates that a failure in any phase should terminate with return code 12:

ON FAILURE ALL TERMINATE UTILITY RETCODE 12
The default for this option is ON FAILURE ALL TERMINATE UTILITY. In this case, if UNLOAD PLUS encounters a failure in any phase, the job terminates with the highest assigned return code.

**Note**
In version 10.1 and earlier, the default behavior was to stop the utility instead of terminating.

**TERMINATE UTILITY**

When TERMINATE UTILITY is in effect for an abend that occurs during any or all of the UTILINIT, UNLOAD, or UTILTERM phases, UNLOAD PLUS completes the following tasks:

- Deletes the row that contains the utility ID from the BMCUTIL and BMCSYNC tables
- Issues either the normal return code or the one that you specify with the RETCODE option
- With the following exceptions, deletes the SYSREC, SYSRED, SORTWK, and referenced files if you have specified DELETEFILES YES
  - If DIRECT NO is in effect, UNLOAD PLUS does not delete any referenced files.
  - UNLOAD PLUS does not delete any HFS referenced files.
  - If you are running the unload job in a worklist environment, UNLOAD PLUS deletes only the dynamically allocated SYSREC, SYSRED, and referenced file data sets.

**Multiple statements**

When you specify multiple ON FAILURE statements, UNLOAD PLUS functions as follows:

- For competing ON FAILURE statements, UNLOAD PLUS honors the last statement.
- When a phase-specific ON FAILURE statement follows an ON FAILURE ALL statement, UNLOAD PLUS honors the phase-specific statement for that phase and the ON FAILURE ALL statement for remaining phases.

In the following example, a failure in any phase results in return code 20, overriding the first statement:

```
ON FAILURE UNLOAD TERMINATE UTILITY RETCODE 25
ON FAILURE ALL TERMINATE UTILITY RETCODE 20
```
If you reverse these statements (next example), a failure in the UNLOAD phase results in return code 25, and a failure in the UTILINIT or UTILTERM phase results in return code 20:

```
ON FAILURE ALL TERMINATE UTILITY RETCODE 20
ON FAILURE UNLOAD TERMINATE UTILITY RETCODE 25
```

**RETCODE**

This option allows you to specify which return code that UNLOAD PLUS should issue when an abnormal termination occurs during the phase that you specified with TERMINATE UTILITY. Valid values are 0 through 4095.

**Additional considerations**

The following considerations apply to this option:

- If you do not specify a value for this option, UNLOAD PLUS issues the normal return code.
- BMC recommends that you avoid specifying a value of 0 or 4.
- If you specify an integer that is greater than 31, UNLOAD PLUS issues a user abend that is equal to the integer specified.
- When an abnormal termination occurs, UNLOAD PLUS does not always honor the RETCODE value you specified. In these cases, see the job log (JESMSGLG) for the system abend code.

**STOP UTILITY**

In versions 10.2 and later, UNLOAD PLUS no longer honors this syntax. If you specify STOP UTILITY, UNLOAD PLUS changes the syntax to TERMINATE UTILITY and issues message BMC51704I.

In versions 10.1 and earlier, if you specified STOP UTILITY for an abend that occurred during the UTILINIT, UNLOAD, or UTILTERM phase, UNLOAD PLUS stopped the job and left the utility ID in the BMCUTIL and BMCSYNC tables.

**DELETEFILES**

The DELETEFILES option tells UNLOAD PLUS whether to delete the SYSREC, SYSRED, SORTWK, and referenced files when the unload is unsuccessful.
For an example of using the DELETEFILES option when an unload job terminates, see “Example 14 - Handling an abnormal termination” on page 363.

**Specifying the default**

You can specify the default for the DELETEFILES command option in your installation options module by using the DELFILES installation option. UNLOAD PLUS was shipped with a default value of YES for this option. The command option overrides the default that is in the installation options module.

**YES**

DELETEFILES YES tells UNLOAD PLUS to perform DELETEFILES processing, which differs depending on whether the unload job is running in a worklist environment:

- During DELETEFILES processing outside of a worklist, UNLOAD PLUS deletes the following data sets:
  - All physical sequential data sets whose ddnames match the SYSREC, SYSRED, and SORTWK ddnames or ddname prefixes
  - All referenced file data sets

- If you specify DELETEFILES YES in a worklist, UNLOAD PLUS deletes only the dynamically allocated SYSREC, SYSRED, and referenced file data sets.

  In a worklist environment, UNLOAD PLUS ignores the value that you set for the DELFILES installation option and, by default, does not delete the files. UNLOAD PLUS functions this way so that the files exist for subsequent executions in the worklist.

To specify the final disposition of tape files when you specify DELETEFILES YES, you can specify the TAPEDISP installation option.

**Restrictions**

UNLOAD PLUS does not delete the following files:

- When DIRECT NO is in effect, any referenced files
- Any HFS referenced files
- Any files that are defined with an abnormal allocation disposition other than DELETE
- Any files that are defined with an expiration date or retention period that prevents the delete from being honored
UNLOAD PLUS does not delete files when the job terminates with an x22 abend.

**NO**

DELETEFILES NO tells UNLOAD PLUS not to delete any data sets after an unsuccessful unload.

**CENTURY**

This option specifies the 100-year range that determines the century for DATE and TIMESTAMP external formats that have two-digit year values.

```
CENTURY (ccyy,ccyy)
```

The first four-digit year value must be less than the second four-digit year. You must specify both values, and the values must span 100 years.

You can also specify CENTURY on the field specification option if you want to apply it to a particular DATE or TIMESTAMP column. For more information, see the individual data type.

**Specifying the default**

You can specify the default for the CENTURY command option in your installation options module by using the CENTURY installation option. UNLOAD PLUS was shipped with a default value of (1950,2049) for this option. The command option overrides the default that is in the installation options module.

```
(ccyy,ccyy)
```

Any two-digit year in the data that you are unloading that lies between the first yy specification and 99 is prefixed with the first cc value to create a four-digit year. Any two-digit year in the data that you are unloading that lies between 00 and the second yy specification is prefixed with the second cc value to create a four-digit year.

For example, if you specify CENTURY(1950,2049), UNLOAD PLUS places 19 in front of each two-digit year with a value 50 through 99, and places 20 in front of each two-digit year with a value 00 through 49. The date 99/12/31 becomes 1999/12/31 and 00/12/31 becomes 2000/12/31.

**DRAIN_WAIT**

The DRAIN_WAIT option specifies the drain timeout value to use.
This option is meaningful only when DIRECT YES is in effect.

If it cannot drain all of the objects within the time period specified by DRAIN_WAIT, UNLOAD PLUS completes the following process:

1. Releases the drains that it has obtained so far

2. Waits the length of time that you specify in the RETRY_DELAY command option (or DRNDELAY installation option)

3. Tries again to drain the objects for the number of times that you specify in the RETRY command option (or DRNRETRY installation option)

**Note**
The FORCE option tells UNLOAD PLUS whether to cancel DB2 threads that might prevent a successful drain and, if so, when to cancel them. Depending on the FORCE option that you specify, the drain process might differ from the process described here. For more information, see “FORCE” on page 179.

**Restriction**
UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**

You can specify the default for the DRAIN_WAIT command option in your installation options module by using the DRNWAIT installation option. UNLOAD PLUS was shipped with a default value of NONE for this option. The command option overrides the default that is in the installation options module.

**NONE**

If you specify NONE, the drain request that UNLOAD PLUS issues times out immediately if the drain cannot acquire the lock. NONE prevents any application transactions from being queued during the drain process. BMC recommends that you specify NONE in high-transaction environments.
**UTIL**

UTIL tells UNLOAD PLUS to use the standard DB2 utility timeout value defined in DSNZPARMS for your site (IRLMRWT multiplied by UTIMOUT). The wait time applies to each object involved in the unload.

**SQL**

If you specify SQL, UNLOAD PLUS uses the standard SQL timeout value (IRLMRWT) as the drain timeout value. The wait time applies to each object involved in the unload.

**integer**

Specify any integer value from 0 through 1800, as follows:

- 0 is equivalent to the value UTIL.
- 1 through 1800 specifies the number of seconds to wait to obtain the drain for each drain retry before timing out.

**RETRY**

The RETRY option specifies the maximum number of times that you want the utility to attempt to obtain a drain before it terminates.

The number of attempts can range from 0 through 255.

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**

You can specify the default for the RETRY command option in your installation options module by using the DRNRETRY installation option. The utility was shipped with a default value of 255 for this option. The command overrides the default that is in the installation options module.
**RETRY_DELAY**

After a drain times out, the RETRY_DELAY option specifies the minimum number of seconds that you want the utility to wait before it tries again to obtain the drain.

The number of seconds can range from 1 through 1800.

UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**

You can specify the default for the RETRY_DELAY command option in your installation options module by using the DRNDELAY installation option. The utility was shipped with a default value of 1 for this option. The command overrides the default that is in the installation options module.

**DSPLOCKS**

The DSPLOCKS option tells the utility what action to take regarding displaying claims and locks if a drain attempt times out.

You can specify one of the following keywords:

**Table 32: DSPLOCKS option keywords**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRNFAIL</td>
<td>DRNFAIL displays the claims and locks once, after the final attempt to obtain the drain times out.</td>
</tr>
<tr>
<td>NONE</td>
<td>NONE displays no claims or locks.</td>
</tr>
<tr>
<td>RETRY</td>
<td>RETRY displays claims and locks after each drain timeout.</td>
</tr>
</tbody>
</table>

**Restriction**
UNLOAD PLUS ignores this option when DIRECT NO is in effect.

**Specifying the default**

You can specify the default for the DSPLOCKS command option in your installation options module by using the DSPLOCKS installation option. The utility was shipped with a default value of DRNFAIL for this option. The command overrides the default that is in the installation options module.

**FORCE**

The FORCE option specifies whether to cancel DB2 threads that might prevent a drain process from completing.

- **FORCE**
  - **NONE**
  - **ALL**
  - **REPORTONLY**
  - **FORCE_AT**
    - **START**
    - **RETRY**
    - **LASTRETRY**
  - **FORCE_DELAY**
    - **integer**
  - **FORCE_RPT**
    - **NO**
    - **YES**

*This option is meaningful only when DIRECT YES is in effect.*

You cannot use this option to cancel threads when UNLOAD PLUS is attempting to acquire a lock on a table. For information about when UNLOAD PLUS uses lock table processing instead of drain processing, see “Drain and lock table processing in UNLOAD PLUS” on page 51.

**Specifying the default**

You can specify a default for the FORCE command option in your installation options module by using the FORCE installation option. UNLOAD PLUS was shipped with a default value of NONE for this option. The FORCE command option overrides the default that is in the installation options module.
NONE

FORCE NONE tells UNLOAD PLUS not to cancel DB2 threads that might prevent the drain process from completing.

ALL

FORCE ALL tells UNLOAD PLUS to cancel both read and write claimers at the point specified by the FORCE_AT option.

FORCE_AT

The FORCE_AT option tells UNLOAD PLUS at which point during the drain process to cancel DB2 threads that might prevent the drain process from completing. You can specify one of the following keywords:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>FORCE_AT START tells UNLOAD PLUS to start canceling threads when the drain request begins.</td>
</tr>
<tr>
<td>RETRY</td>
<td>FORCE_AT RETRY tells UNLOAD PLUS to start canceling threads the first time the drain process times out and UNLOAD PLUS attempts to retry the drain.</td>
</tr>
<tr>
<td>LASTRETRY</td>
<td>FORCE_AT LASTRETRY tells UNLOAD PLUS to start canceling threads at the beginning of the last retry attempt following a drain process timeout. You can control the number of drain retry attempts with the RETRY command option or DRNRETRY installation option.</td>
</tr>
</tbody>
</table>

Specifying the default

You can specify a default for the FORCE_AT command option in your installation options module by using the first parameter of the FORCE_AT installation option. UNLOAD PLUS was shipped with a default value of START for this option. The FORCE_AT command option overrides the default that is in the installation options module.

Additional considerations

The cancelation process includes a small amount of time to issue the cancel commands, plus any time that DB2 needs to roll back the canceled transactions. If the drain wait time (controlled by the DRAIN_WAIT option) is shorter than the total cancelation process time, the drain fails. FORCE_AT START and FORCE_AT RETRY allow for additional drain attempts after the start of the cancelation process; therefore, these options increase the likelihood that the drain will succeed when a short drain wait time is in effect. (Alternatively, you can increase the drain wait time.)
FORCE_DELAY

The FORCE_DELAY option tells UNLOAD PLUS how long to wait, in hundredths of a second, before starting the thread cancelation process. For example, specify 7 to tell UNLOAD PLUS to wait .07 seconds.

Specify an integer value of 0 or greater. A value of 0 tells UNLOAD PLUS to start the cancelation process immediately upon reaching the point specified by the FORCE_AT option.

Specifying the default

You can specify a default for the FORCE_DELAY command option in your installation options module by using the second parameter of the FORCE_AT installation option. UNLOAD PLUS was shipped with a default value of 3 for this parameter. The FORCE_DELAY command option overrides the default that is in the installation options module.

FORCE_RPT

The FORCE_RPT option tells UNLOAD PLUS whether to display a report of the canceled threads. You can specify one of the following keywords:

Table 34: FORCE_RPT option keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>NO tells UNLOAD PLUS not to display the report.</td>
</tr>
<tr>
<td>YES</td>
<td>YES tells UNLOAD PLUS to display the report. If your JCL includes a BMCFORCE DD statement, UNLOAD PLUS sends the canceled threads report to that data set. Otherwise, the report is displayed in the UNLOAD PLUS SYSPRINT.</td>
</tr>
</tbody>
</table>

Specifying the default

You can specify a default for the FORCE_RPT command option in your installation options module by using the FORCE_RPT installation option. UNLOAD PLUS was shipped with a default value of NONE for this option. The FORCE_RPT command option overrides the default that is in the installation options module.

REPORTONLY

FORCE REPORTONLY tells UNLOAD PLUS to display a report of the threads that it would have canceled if you had specified FORCE ALL. When reporting which threads it would have canceled, UNLOAD PLUS takes into account the current values of FORCE_AT and FORCE_DELAY.
If your JCL includes a BMCFORCE DD statement, UNLOAD PLUS sends the canceled threads report to that data set. Otherwise, the report is displayed in the UNLOAD PLUS SYSPRINT.

FORCE_AT

For information about this option, see “ALL” on page 180.

FORCE_DELAY

For information about this option, see “ALL” on page 180.

FORCE_RPT

UNLOAD PLUS ignores FORCE_RPT when you specify FORCE REPORTONLY.

**SET CURRENT QUERY ACCELERATION ALL**

This option tells UNLOAD PLUS to attempt to use an IBM DB2 Data Analytics Accelerator server.

This option is valid only when DIRECT NO is in effect.

Specify SET CURRENT QUERY ACCELERATION ALL to offload eligible query processing to the accelerator server for all SELECT statements in this UNLOAD step. To offload query processing, your system, object, and query must meet all requirements as described in the IBM documentation for the accelerator server.

This option overrides the value that is defined in the QUERY_ACCELERATION DSNZPARM on your system.

**OUTPUT command options**

This section describes the options that you can specify in an OUTPUT descriptor statement.

**OUTPUT**

The OUTPUT command tells UNLOAD PLUS to allocate output data sets (primary and secondary unload data sets and referenced file data sets) dynamically.
Specifying the OUTPUT command, output descriptors, and associated options on the UNLOAD command produces dynamically allocated output data sets that are allocated to tape or to disk. Specify the OUTPUT keyword to introduce a new output descriptor name.

**Considerations**

The following considerations apply to the OUTPUT command and its options:

- Descriptor names must be unique and cannot exceed 8 characters.
- UNLOAD PLUS processes all statements in the SYSIN data set sequentially. You can use more than one OUTPUT statement in a SYSIN data set, but each output descriptor must have a unique name. UNLOAD PLUS searches for a match between a descriptor on the list of output descriptors and the value or suffixed value of the UNLOADDN or FILEREFDN keyword. If a match occurs, UNLOAD PLUS merges the identified descriptor with the installation options that the $ADUOUTP module for the corresponding output data set specifies. If UNLOAD PLUS cannot find such a match, UNLOAD PLUS uses the default options.
- Options that are specific to disk data sets and those that are specific to tape data sets are mutually exclusive.
- You can specify the defaults for these options at installation in the $ADUOUTP macro for dynamic allocation of output data sets. For details about $ADUOUTP and these installation options, see “Dynamic allocation installation options” on page 447.

**Common options for the OUTPUT command**

This section describes options that you can use for output data sets that UNLOAD PLUS writes to disk or to tape.

**UNIT**

The UNIT option specifies the device type (tape or disk) for syntax checking and dynamic disk allocation.
UNLOAD PLUS compares the value of UNIT to a list of tape devices that UNLOAD PLUS retrieves from MVS. If the value does not match one of the devices on the list, UNLOAD PLUS assumes that the device is disk.

If you also specify DSNTYPE EXTPREF or DSNTYPE EXTREQ, ensure that the unit supports extended attributes.

**Specifying the default**

You can specify the default for the UNIT command option in your installation options module by using the WORKUNIT installation option. UNLOAD PLUS was shipped with a default value of SYSALLDA for this option. The command option overrides the default that is in the installation options module.

**$NO_UNIT$**

Specify $NO_UNIT$ to tell UNLOAD PLUS not to pass a unit value to dynamic allocation. This value is particularly useful in an SMS environment.

**DSNAME**

The DSNAME option specifies a data set name for the output data set.

You can use any of the symbolic variables in Table 35 on page 186 to construct names for your unload data sets.

When UNLOAD PLUS resolves the dynamic allocation specification for all of your SELECT statements, the utility generates data set names. UNLOAD PLUS uses the DSNAME value for an assigned output descriptor to determine the number of data sets that UNLOAD PLUS dynamically allocates for a specific SELECT statement. If
you allow multiple SELECT statements to share the same output descriptor whose
DSNAME value results in a single data set, UNLOAD PLUS creates a dynamically
allocated default data set.

Referenced files

For a referenced file, you must specify the fully qualified data set or file system name
with the DSNAME option. You can use a pattern with DSNAME as follows:

■ For DSNTYPE PDS or LIBRARY, you can use a pattern to create your data set
  names.

■ For DSNTYPE HFS when you specify DIRECT YES, you can use a pattern to have
  UNLOAD PLUS create additional subdirectories in your file system path. In the
  following example, the root portion of the path (/home/rdacxb) already exists
  and is the mount point for the file system. UNLOAD PLUS adds subdirectories to
  this path for the subsystem ID and partition number based on the two variables in
  the DSNAME pattern.

  OUTPUT CLOB01 DSNAME('/home/rdacxb/&SSID/p&PART') DSNTYPE(HFS)

  **Note**
  When you specify DIRECT NO, specifying a pattern does not result in
  additional subdirectories in your file system path. UNLOAD PLUS terminates
  if it cannot find a file system path that uses the name that resolves from your
  pattern.

Note the following considerations about the name that you specify:

■ For DSNTYPE PDS or LIBRARY, you cannot specify an existing data set.

■ You cannot specify a PDS member name or HFS file name. UNLOAD PLUS
  generates PDS members and HFS files.

Specifying the default

You can specify the default for the DSNAME command option in your installation
options module by using the DSNAME installation option. UNLOAD PLUS was
shipped with one of the following values for this option:

■ &USERID.&JOBNAME.&TS.&TYPE.S&SELNUM (for unload data sets)

■ &USREID.&JOBNAME.&TYPE.S&SELNUM.P&POSNUM (for referenced files)

The command option overrides the default that is in the installation options module.
Using symbolic variables

Table 35 on page 186 describes the symbolic variables that you can use to construct names for your output data sets.

The maximum total length that UNLOAD PLUS allows for a data set name is 44 bytes. UNLOAD PLUS removes any trailing blanks that result from resolution of a symbolic variable.

Table 35: Symbolic variables for output data set names

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ATTACH</td>
<td>DB2 group attachment name or subsystem ID</td>
<td>4 maximum</td>
</tr>
<tr>
<td>&amp;DATE a, b</td>
<td>Current date (in the format YYMMDD)</td>
<td>6</td>
</tr>
<tr>
<td>&amp;DAY a, b</td>
<td>Current day (in the format DD)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;DB</td>
<td>Database</td>
<td>8</td>
</tr>
<tr>
<td>&amp;HOUR a, b</td>
<td>Current hour (in the format HH)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;JDATE a, b</td>
<td>Current Julian date (in the format YYDDD)</td>
<td>5</td>
</tr>
<tr>
<td>&amp;JDAY a, b</td>
<td>Current Julian day (in the format DDD)</td>
<td>3</td>
</tr>
<tr>
<td>&amp;JOBNAME</td>
<td>Job name</td>
<td>8 maximum</td>
</tr>
<tr>
<td>&amp;MIN a, b</td>
<td>Current minute (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;MINUTE a, b</td>
<td>Current minute (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;MONTH a, b</td>
<td>Current month (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;OBID a</td>
<td>OBID of the table that UNLOAD PLUS is unloading</td>
<td>4 bytes maximum</td>
</tr>
<tr>
<td>Variable</td>
<td>Definition</td>
<td>Length (bytes)</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| &PART \[^{a,b}\] | Data set or partition that UNLOAD PLUS is unloading. Note the following information about &PART:  
- If you specify the LOGICAL keyword with the PART option, the &PART variable indicates the logical partition rather than the physical partition.  
- Consider using &PART to limit the number of partitions unloaded in a single step to avoid encountering data set allocation restrictions of the operating system.  
- For referenced file data sets, UNLOAD PLUS creates one data set for each base table space partition. If you use this variable, be aware that UNLOAD PLUS will terminate if it attempts to allocate more than 256 data sets for referenced files. | 3 bytes in either of the following circumstances:  
- When unloading a nonpartitioned table space (in which case, the expanded value is 000)  
- For partitions 1 through 999 when unloading from a partitioned table space  
- 4 bytes for partitions 1000 through 4096 when unloading from a partitioned table space |
| &POSNUM \[^{a}\] | Position number of the XML or LOB column within the select list (valid only for referenced file data sets) | 3 |
| &SEC \[^{a,b}\] | Current second (in the format SS) | 2 |
| &SECOND \[^{a,b}\] | Current second (in the format SS) | 2 |
| &SELNUM \[^{a}\] | SELECT statement number | 3 |
| &SSID | DB2 subsystem ID | 4 |
| &STEPNAME | Step name  
UNLOAD PLUS ignores PROC names. | 8 maximum |
| &TIME \[^{a,b}\] | Current time (in the format HHMMSS) | 6 |
| &TS | Table space | 8 maximum |
| &TYPE | Data set type that UNLOAD PLUS is allocating  
This symbol resolves to a value of SYSREC, SYSRED, or SYSREF (where SYSREF indicates a referenced file). | 6 |
| &USERID | Job or TSO user | 7 maximum |
Variable | Definition | Length (bytes)
---|---|---
&UTIL | Utility ID
  Note the following information about &UTIL:
  - UNLOAD PLUS truncates values longer than 8 bytes.
  - Utility IDs that include special characters might cause UNLOAD PLUS to generate invalid names. | 8 maximum

&YEAR<sup>a, b</sup> | Current year (in the format YY) | 2
--- | --- | ---

---

<sup>a</sup> You must prefix symbols with a numeric result with one or more alphabetic characters.

<sup>b</sup> UNLOAD PLUS assigns values for these variables when the utility allocates the output data set. All dynamically allocated data sets have the same value.

---

When you specify DIRECT NO and you are using dynamic allocation, UNLOAD PLUS accepts all symbolic variables, but externalizes &DB, &TS, &OBID, and &PART differently from the form in which it externalizes them when you specify DIRECT YES. Table 36 on page 188 shows the resolutions that UNLOAD PLUS provides when you specify DIRECT NO and include these specific symbolic variables in your output descriptor. The example that follows this table illustrates a sample output descriptor.

**Table 36: Resolution of symbolic variables when using DIRECT NO**

<table>
<thead>
<tr>
<th>Symbolic variable</th>
<th>Resolution of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DB</td>
<td>$DB n</td>
</tr>
<tr>
<td>&amp;TS</td>
<td>$TS n</td>
</tr>
<tr>
<td>&amp;OBID</td>
<td>000</td>
</tr>
<tr>
<td>&amp;PART</td>
<td>000</td>
</tr>
</tbody>
</table>

**Example of symbolic variables when using DIRECT NO**

When you specify DIRECT NO, UNLOAD PLUS resolves the symbolic variables in the following output descriptor

```
OUTPUT SYSREC DSNAME 'ADU.&DB.&TS.X&OBID.P&PART'
```

to the following data set name

```
ADU.$DB1.$TS1.X000.P000
```
**DSNTYPE**

The DSNTYPE option tells UNLOAD PLUS either what type of file you are using or what type of data set to allocate dynamically, depending on the value that you specify.

UNLOAD PLUS uses the DSNTYPE values for the following types of files:

- For referenced files, the PDS, LIBRARY, and HFS values for DSNTYPE tell UNLOAD PLUS which type of file you are using. UNLOAD PLUS requires this option when you are unloading LOB or XML data to a referenced file. For more information about using referenced files, see “Unloading LOB and XML data” on page 54.

- For unload data sets (SYSREC and SYSRED), the remaining values for DSNTYPE tell UNLOAD PLUS what type of data set to allocate dynamically.

You can specify one of the following values:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDS</td>
<td>Referenced files</td>
<td>PDS tells UNLOAD PLUS to dynamically allocate a partitioned data set (PDS) as your referenced file.</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>Referenced files</td>
<td>LIBRARY tells UNLOAD PLUS to dynamically allocate an extended partitioned data set (PDSE) as your referenced file.</td>
</tr>
<tr>
<td>HFS</td>
<td>Referenced files</td>
<td>HFS tells UNLOAD PLUS that the referenced file is a hierarchical file system (HFS). The HFS named in DSNAME must be preallocated and mounted before the UNLOAD PLUS job starts. The only other valid OUTPUT command options with DSNTYPE HFS are the descriptor name, DSNAME, and SUBSETS.</td>
</tr>
</tbody>
</table>
### UNLOAD PLUS options

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>Unload (SYSREC and SYSRED) data sets</td>
<td>NONE tells UNLOAD PLUS to not use any extended attributes for this data set allocation.</td>
</tr>
<tr>
<td>LARGE</td>
<td>Unload (SYSREC and SYSRED) data sets</td>
<td>LARGE tells UNLOAD PLUS to allocate this data set as a large format sequential data set. This option enables data sets larger than 65,535 tracks.</td>
</tr>
<tr>
<td>BASIC</td>
<td>Unload (SYSREC and SYSRED) data sets</td>
<td>BASIC tells UNLOAD PLUS to allocate this data set as a basic sequential data set. This data set will be limited to 65,535 tracks.</td>
</tr>
<tr>
<td>EXTREQ</td>
<td>Unload (SYSREC and SYSRED) data sets</td>
<td>EXTREQ tells UNLOAD PLUS to allocate this data set as an extended format data set. Ensure that the unit that you specify or default to with the UNIT option supports extended attributes.</td>
</tr>
<tr>
<td>EXTPREF</td>
<td>Unload (SYSREC and SYSRED) data sets</td>
<td>EXTPREF indicates that you prefer that UNLOAD PLUS allocate this data set as an extended format data set. If this allocation is not possible, the data set is allocated as a basic format data set. Ensure that the unit that you specify or default to with the UNIT option supports extended attributes.</td>
</tr>
</tbody>
</table>

**Specifying the default**

You can specify the default for the DSNTYPE command option in your installation options module by using the DSNTYPE installation option. UNLOAD PLUS was shipped with a default value of PDF for referenced files and NONE for unload data sets. The command option overrides the default that is in the installation options module.

**DIR**

DIR tells UNLOAD PLUS the number of 256-byte records to allocate for the directory of a new PDS or PDSE.

```
DIR integer
```

This option is valid only when you are unloading LOB or XML data to a referenced file, and DSNTYPE is PDS or LIBRARY. UNLOAD PLUS ignores this option when DSNTYPE is HFS. If you specify DIR for a primary or secondary output data set (SYSREC or SYSRED), UNLOAD PLUS terminates.
SUBSETS

The SUBSETS option tells UNLOAD PLUS how many data sets to allocate per base table space partition or per simple or segmented table space.

This option is meaningful only when DIRECT YES is in effect.

When UNLOAD PLUS allocates multiple data sets, a separate LOB or XML table space subtask unloads to each data set, enabling UNLOAD PLUS to multitask.

Valid values are 0 through 99. 0 or 1 (the default) tells UNLOAD PLUS to allocate a single data set for each partition or for each simple or segmented table space. For DSNTYPE HFS, UNLOAD PLUS creates a new subdirectory in your file system path for each subset.

UNLOAD PLUS adds a suffix to each of these data sets. The suffix indicates the subset to which the data set belongs. For example, given the following OUTPUT statement for a three-partition table space (Figure 16 on page 191), UNLOAD PLUS allocates the data sets listed in Figure 17 on page 191.

Figure 16: Sample OUTPUT statement with SUBSETS

```
OUTPUT LOBFR2 DSNAMES 'RDAJDM.UNLOAD.LOBFR2.P&PART'
  DSNTYPE PDS
  UNIT SYSDA
  SPACE 10,10 TRK
  DIR 10
  SUBSETS 3
```

Figure 17: Data set names resulting from SUBSETS option

```
RDAJDM.UNLOAD.LOBFR2.P0001.S01
RDAJDM.UNLOAD.LOBFR2.P0001.S02
RDAJDM.UNLOAD.LOBFR2.P0001.S03
RDAJDM.UNLOAD.LOBFR2.P0002.S01
RDAJDM.UNLOAD.LOBFR2.P0002.S02
RDAJDM.UNLOAD.LOBFR2.P0002.S03
RDAJDM.UNLOAD.LOBFR2.P0003.S01
RDAJDM.UNLOAD.LOBFR2.P0003.S02
RDAJDM.UNLOAD.LOBFR2.P0003.S03
```

Restrictions

The following restrictions apply to the SUBSETS option:

- If DIRECT NO is in effect, UNLOAD PLUS ignores this option.
- UNLOAD PLUS terminates in either of the following cases:
— If the following values indicate that UNLOAD PLUS is to allocate more than 256 data sets:

— Number of LOB and XML columns that you are unloading

— Value of this option

— Number of partitions in the base table space, if you include the &PART variable in your DSNAME specification

— If the following values indicate that your job would require more than 200 subtasks:

— Number of LOB and XML columns that you are unloading

— Value of this option

**LRECL**

LRECL specifies the logical record length to be used for any data set dynamically allocated with this output descriptor when you also specify USELRECL YES.

LRECL integer

If you specify USELRECL NO, the LRECL specification is ignored.

**Restrictions**

Note the following restrictions when specifying LRECL:

- If the LRECL that you specify is shorter than the length that is required to contain the largest formatted row, UNLOAD PLUS ends with error message BMC51649E.

- If you specify USELRECL YES with FORMAT CSV, FORMAT XML, FORMAT INTERNAL, or FORMAT BMCLOAD, UNLOAD PLUS ignores both USELRECL YES and LRECL.

**Additional considerations**

The following considerations apply to the LRECL option:

- When you specify LRECL, UNLOAD PLUS normally defines the dynamically allocated data set with a fixed record format. However, if you specify RECFM VB on your UNLOAD command (or it is in effect by default from the RECFM installation option) UNLOAD PLUS uses variable block format for the data set records.
If you do not specify LRECL in your OUTPUT statement and you specify USELRECL YES, UNLOAD PLUS calculates the LRECL value. The associated data set will be fixed, variable, or variable-spanned, according to the standard record length and record format calculations for UNLOAD PLUS.

**VOLCNT**

VOLCNT specifies the largest number of volumes that you expect UNLOAD PLUS to process when unloading a single data set.

For disk data sets, this option limits the values of the VOLUMES option. For both tape and disk data sets, the integer must be equal to or greater than the number of volumes that UNLOAD PLUS produces for the single largest output data set.

*Additional considerations*

Note the following considerations for VOLCNT:

- For a nonspecific disk data set allocation, UNLOAD PLUS ignores VOLCNT. Use UNITCNT to request a multi-volume disk data set.

- If you define the VOLUMES option for disk data sets, the default is the number of volumes in the list. To use the MVS default, set VOLCNT to 0.

- If you are using SMS in your system, BMC recommends that you set VOLCNT to 0 if your ACS routines are set up to provide a volume count.

- If you are unloading a large number of partitions, consider lowering the VOLCNT value from the default of 25 to avoid data set allocation limitations.

*Specifying the default*

You can specify the default for the VOLCNT command option in your installation options module by using the VOLCNT installation option. UNLOAD PLUS was shipped with a default value of 25 for this option. The command option overrides the default that is in the installation options module.

**GDGLIMIT**

Specify GDGLIMIT to set the maximum number of data sets in the group.
You can specify a number from 1 to 255. If the generation data group (GDG) base exists, UNLOAD PLUS ignores this option.

**Specifying the default**

You can specify the default for the GDGLIMIT command option in your installation options module by using the GDGLIMIT installation option. UNLOAD PLUS was shipped with a default value of 5 for this option. The command option overrides the default that is in the installation options module.

**STORCLAS**

Specify the STORCLAS name to provide an SMS storage class name for the named output descriptor.

The value of `class` must be valid and cannot exceed eight characters.

UNLOAD PLUS uses any storage management subsystem (SMS) DATACLAS, STORCLAS, and MGMTCLAS values that are in the default installation output descriptor for disk and tape data set allocations unless you override them in an associated OUTPUT descriptor statement. Check your options settings in the installation options module. If you do not want to use a value for STORCLAS, and a value exists in the current default output descriptor, specify STORCLAS NONE in the OUTPUT descriptor statement.

**Specifying the default**

You can specify the default for the STORCLAS command option in your installation options module by using the STORCLAS installation option. UNLOAD PLUS was shipped without a default value for this option. The command option overrides the default that is in the installation options module.
**DATACLAS**

Specify the DATACLAS name to provide an SMS data class name for the named output descriptor.

```
DATACLAS class
```

The value of `class` must be valid and cannot exceed eight characters.

UNLOAD PLUS uses any storage management subsystem (SMS) DATACLAS, STORCLAS, and MGMTCLAS values that are in the default installation output descriptor for disk and tape data set allocations unless you override them in an associated OUTPUT descriptor statement. Check your options settings in the installation options module. If you do not want to use a value for DATACLAS, and a value exists in the current default output descriptor, specify DATACLAS NONE in the OUTPUT descriptor statement.

**Specifying the default**

You can specify the default for the DATACLAS command option in your installation options module by using the DATACLAS installation option. UNLOAD PLUS was shipped without a default value for this option. The command option overrides the default that is in the installation options module.

**MGMTCLAS**

Specify the MGMTCLAS name to provide an SMS management class name for the named output descriptor.

```
MGMTCLAS class
```

The value of `class` must be valid and cannot exceed eight characters.

UNLOAD PLUS uses any storage management subsystem (SMS) DATACLAS, STORCLAS, and MGMTCLAS values that are in the default installation output descriptor for disk and tape data set allocations unless you override them in an associated OUTPUT descriptor statement. Check your options settings in the installation options module. If you do not want to use a value for MGMTCLAS, and a value exists in the current default output descriptor, specify MGMTCLAS NONE in the OUTPUT descriptor statement.
Specifying the default

You can specify the default for the MGMTCLAS command option in your installation options module by using the MGMTCLAS installation option. UNLOAD PLUS was shipped without a default value for this option. The command option overrides the default that is in the installation options module.

UNITCNT

Specify UNITCNT for the unit count for dynamic allocation.

Valid values are 0 through 59. The value 0 means that UNLOAD PLUS does not specify the unit count for the allocation.

If you do not want to use a value for UNITCNT and a value exists in the current default output descriptor, specify UNITCNT 0 in the OUTPUT descriptor statement.

If the OUTPUT statement is for a referenced file and you specify DIRECT NO, specify UNITCNT 0 or 1.

Considerations

The following considerations apply to the UNITCNT option:

- If you specify UNITCNT integer for tape output, UNLOAD PLUS allocates the number of tape drives that you specified. When a tape volume is at the end of tape, UNLOAD PLUS begins to write on the next drive immediately so that you do not need to wait for the tape to rewind.

- The value of UNITCNT affects the primary and secondary space allocation requests for SMS-managed data sets when you allow UNLOAD PLUS to optimize primary and secondary extent sizes automatically.

Specifying the default

You can specify the default for the UNITCNT command option in your installation options module by using the UNITCNT installation option. UNLOAD PLUS was shipped with a default value of 0 for this option. The command option overrides the default that is in the installation options module.
Disk options for the OUTPUT command

This section describes options that apply only to output data sets that UNLOAD PLUS writes to disk devices.

SPACE

Specify SPACE to provide UNLOAD PLUS with the allocation quantity and the allocation units (tracks or cylinders) for the named output descriptor.

Considerations

Note the following considerations regarding the SPACE option:

- When you use DIRECT NO, you must specify both the primary and secondary SPACE quantities.

- When unloading LOB or XML data to PDS or PDSE referenced data sets, you must specify this option to tell UNLOAD PLUS how to allocate your data sets. If you are allocating a data set for each partition, UNLOAD PLUS uses the specified quantity for each data set. UNLOAD PLUS ignores this option for HFS referenced files.

Specifying the default

You can specify the default for the allocation unit portion of the SPACE command option in your installation options module by using the SPACE installation option. UNLOAD PLUS was shipped with a default value of CYL for this option. The command option overrides the default that is in the installation options module.

PCTPRIM

Specify PCTPRIM for the named output descriptor to set a value for the percentage of the total space that UNLOAD PLUS must allocate as primary space.

You can specify AUTO or an integer from 1 through 100.
UNLOAD PLUS ignores this option for referenced files.

For large table spaces, the primary allocation that PCTPRIM calculates might be too large. To override the calculated value, specify the MAXPRIM option. For more information, see the description of the MAXPRIM option.

**Specifying the default**

You can specify the default for the PCTPRIM command option in your installation options module by using the PCTPRIM installation option. UNLOAD PLUS was shipped with a default value of AUTO for this option. The command option overrides the default that is in the installation options module.

**AUTO**

If you specify AUTO for SMS allocations, UNLOAD PLUS uses the largest available extent as the value for the primary allocation. For non-SMS allocations, specifying AUTO is equivalent to specifying a value of 100.

**integer**

You can specify an integer from 1 through 100.

**MAXPRIM**

Specify the MAXPRIM option for the named output descriptor to set the upper limit on the value that the PCTPRIM option calculates.

Specifying a nonzero value for integer establishes an upper limit for primary space allocation. A value of 0 indicates no limit.

UNLOAD PLUS ignores this option for referenced files.

When you are using the SMS-guaranteed space allocation, UNLOAD PLUS applies MAXPRIM to all of the volumes that you specified in the VOLUMES syntax option or in the installation options module.

**Specifying the default**

You can specify the default for the MAXPRIM command option in your installation options module by using the MAXPRIM installation option. UNLOAD PLUS was
shipped with a default value of 0 for this option. The command option overrides the default that is in the installation options module.

**MAXSECD**

Use MAXSECD to specify an upper limit on a secondary allocation extent by using the units (tracks or cylinders) that you specified for the SPACE command or installation option.

The resulting extent size may not exceed 65535 tracks.

Specifying a nonzero value for integer establishes an upper limit for the secondary space allocation. A value of 0 indicates no limit. If the amount of required space that UNLOAD PLUS calculates is greater than the MAXSECD limit, UNLOAD PLUS issues message BMC50166W.

**Additional considerations**

The following considerations apply to the MAXSECD option:

- UNLOAD PLUS ignores this option for referenced files.
- Specifying the MAXSECD option when UNLOAD PLUS is automatically sizing a data set might result in additional units being allocated or might cause the data set to be undersized.
- If you specify a value for the primary and secondary allocations for the SPACE command option, UNLOAD PLUS uses the SPACE value and ignores the value that you specify for MAXSECD.
- When you are using the SMS-guaranteed space allocation, UNLOAD PLUS applies MAXSECD to all of the volumes that you specified in the VOLUMES command or installation option.

**Specifying the default**

You can specify the default for the MAXSECD command option in your installation options module by using the MAXSECD installation option. UNLOAD PLUS was shipped with a default value of 0 for this option. The command option overrides the default that is in the installation options module.
**NBRSECD**

After UNLOAD PLUS calculates the primary space allocation, the remaining space is secondary space. Use NBRSECD to specify the number of extents into which UNLOAD PLUS should divide the remaining space.

You can specify AUTO, or an integer from 1 through 123.

UNLOAD PLUS ignores this option for referenced files.

**Specifying the default**

You can specify the default for the NBRSECD command option in your installation options module by using the NBRSECD installation option. UNLOAD PLUS was shipped with a default value of AUTO for this option. The command option overrides the default that is in the installation options module.

**AUTO**

If you specify AUTO, UNLOAD PLUS considers the following criteria and uses a value of either 16 or 123:

- If the allocation is non-SMS or SMS and the data set will not be in extended format, UNLOAD PLUS uses a value of 16.
- If the allocation is SMS and UNLOAD PLUS allocates the data set in extended format, UNLOAD PLUS uses a value of 123.

**integer**

If you specify an integer, the value can range from 1 through 123.

For non-SMS allocations, the maximum value that you can specify for NBRSECD is 16. If you specify a value that is greater than 16, UNLOAD PLUS uses a value of 16.

**FILESZPCT**

Use FILESZPCT to specify the amount by which you want UNLOAD PLUS to adjust the total calculated space for the data sets that this OUTPUT statement describes when automatically sizing data sets.
The value can range from 1 through 999. Values from 1 to 99 indicate a reduction in file size. Values from 101 to 999 indicate an increase in file size. A value of 100 indicates no change in the file size.

UNLOAD PLUS uses the following formula to calculate the resulting total file size:

\[
\text{resultantFileSize} = \text{ROUND} \left( \frac{\text{calculatedFileSize} \times \text{FILESZPCT}}{100} \right)
\]

**Additional considerations**

The following considerations apply to the FILESZPCT option:

- UNLOAD PLUS ignores this option for referenced files.

- If you specify a value for the primary and secondary allocations for the SPACE command option, UNLOAD PLUS uses the SPACE value and ignores the value you specify for FILESZPCT.

- UNLOAD PLUS uses the value calculated by FILESZPCT as the total size (in kilobytes) of the data set before it calculates the primary and secondary space values specified by the PCTPRIM, NBRSECD, and MAXSECD options.

**Specifying the default**

You can specify the default for the FILESZPCT command option in your installation options module by using the FILESZPCT installation option. UNLOAD PLUS was shipped with a default value of 100 for this option. The command option overrides the default that is in the installation options module.

**VOLUMES**

Specify VOLUMES to provide a list of default volumes for the named output descriptor.
UNLOAD PLUS uses the list as the default list for the VOL01 and VOL02 options for the named output descriptor. The number of entries in the list must not exceed the value that VOLCNT specifies for the named output descriptor.

If you do not want to use a value for VOLUMES and you installed one or more volumes at installation, specify VOLUMES(NONE).

For referenced files, UNLOAD PLUS functions as follows:

- For DSNTYPE HFS, UNLOAD PLUS ignores this option.
- For DSNTYPE PDS or LIBRARY, UNLOAD PLUS uses only the first volume in the list.

**Note**
To allocate the primary space that the utility requires for the output data set, UNLOAD PLUS must find sufficient space on the first specified volume.

**Specifying the default**

You can specify the default for the VOLUMES command option in your installation options module by using the VOLUMES installation option. UNLOAD PLUS was shipped without a default value for this option. The command option overrides the default that is in the installation options module.

**DISKRETN**

When you use dynamic allocation, specify DISKRETN for the named output descriptor to set a new retention period (in days) for the current disk data set.

```
DISKRETN integer
```

The value of *integer* must be from 1 through 9999.

**Restrictions**

This option is restricted for referenced files as follows:

- For DSNTYPE HFS, UNLOAD PLUS ignores this option.
- For DSNTYPE PDS or LIBRARY, UNLOAD PLUS terminates if you specify a value for this option.

**Specifying the default**
You can specify the default for the DISKRETN command option in your installation options module by using the DISKRETN installation option. If you do not specify DISKRETN, UNLOAD PLUS does not specify a retention period. When you specify the DISKEXPD command option, it supersedes DISKRETN. If you do not specify DISKEXPD on the OUTPUT descriptor statement, but instead specify DISKRETN, DISKRETN overrides DISKEXPD in the installation options module.

**DISKEXPD**

When you use dynamic allocation, specify DISKEXPD for the named output descriptor to set a new expiration date for the current disk data set.

```
DISKEXPD date
```

The value of *date* must be in the format `yyyyddd` or `yyyy/ddd` where *yyyy* is the 4-digit year and *ddd* specifies the 3-digit Julian day (000 through 365 for a nonleap year, and 000 through 366 for a leap year).

**Restrictions**

This option is restricted for referenced files as follows:

- For DSNTYPE HFS, UNLOAD PLUS ignores this option.
- For DSNTYPE PDS or LIBRARY, UNLOAD PLUS terminates if you specify a value for this option.

**Specifying the default**

You can specify the default for the DISKEXPD command option in your installation options module by using the DISKEXPD installation option. If you do not specify DISKEXPD, UNLOAD PLUS does not provide an expiration date. When you specify the DISKEXPD command option, it supersedes DISKRETN.

**Tape options for the OUTPUT command**

This section describes the options that apply only to unload data sets that UNLOAD PLUS writes to tape.

**TRTCH**

Specify TRTCH to define tape data compression for the named output descriptor.
The following table describes the values that you can specify for this option.

Table 37: Values for the TRTCH option

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>Specify TRTCH NONE if you want to use the MVS default for tape data compression.</td>
</tr>
<tr>
<td>COMP</td>
<td>TRTCH COMP specifies that UNLOAD PLUS provide tape data compression.</td>
</tr>
<tr>
<td>NOCOMP</td>
<td>TRTCH NOCOMP specifies that UNLOAD PLUS prevent data compression.</td>
</tr>
</tbody>
</table>

Specifying the default

You can specify the default for the TRTCH command option in your installation options module by using the TRTCH installation option. UNLOAD PLUS was shipped with a default value of NONE for this option. The command option overrides the default that is in the installation options module.

RETPD

Specify RETPD for the named output descriptor to set a retention period (in days) for the current unload data set.

The value of $integer$ must be 0 through 9999.

Specifying the default

You can specify the default for the RETPD command option in your installation options module by using the RETPD installation option. When you specify EXPDT on the OUTPUT descriptor statement, EXPDT supersedes RETPD. However, if you do not specify EXPDT on the OUTPUT descriptor statement, but instead specify RETPD, RETPD overrides EXPDT in the installation options module.
EXPDT

Specify the EXPDT date in the descriptor for the named output descriptor to set an expiration date for the current unload data set.

The value of date must be in the format yyyyddd or yyyy/ddd, where yyyy is the 4-digit year and ddd is the 3-digit Julian day (000 through 365 for a nonleap year, and 000 through 366 for a leap year). The number 99000 specifies that the tape data sets have no expiration date.

Specifying the default

You can specify the default for the EXPDT command option in your installation options module by using the EXPDT installation option. UNLOAD PLUS was shipped with a default value of 99000, indicating no expiration date, for this option; however, the value might have been changed during installation. When you specify the EXPDT command option, EXPDT overrides RETPD. If you specify the RETPD command option, RETPD overrides EXPDT in the installation options module.

SELECT options

This section describes options that you can specify with your SELECT statement.

SELECT

The SELECT statement provides UNLOAD PLUS with information about the object that you are unloading and the output records.
You specify the following information with the SELECT statement:

- Table that UNLOAD PLUS should unload
- Rows of the table to unload
- Partitions of the table to unload
- Definitions of fields of the output record
- Values in the output record

UNLOAD PLUS requires at least one SELECT statement unless you specify UNLOAD TABLESPACE. You may specify one or more SELECT statements for each table that you want to unload. When you specify multiple SELECT statements, you can specify multiple unload data sets in your JCL to direct the result of each SELECT statement to a different unload data set. For more information, see “Using JCL to specify multiple unload data sets” on page 289.

Place the SELECT statement as the last set of options in your command stream.

**Restrictions**

Note the following SELECT statement restrictions:

- You cannot specify an expression on the SELECT statement.
- You cannot specify a SELECT statement with the UNLOAD TABLESPACE option.
- If you also specify FORMAT BMLOAD, see “Data migration using LOADPLUS (FORMAT BMLOAD)” on page 67 for restrictions that apply to the SELECT option.
- When DIRECT YES is in effect, you cannot specify multiple SELECT statements when unloading LOB or XML data.

**Additional considerations**

The following considerations also apply to the SELECT statement:

- DIRECT NO supports the full range of DB2 SQL SELECT functionality.
- When DIRECT NO is in effect, DB2 enforces row- and column-level security, which might affect the results of your SELECT statement.
- UNLOAD PLUS supports the use of the double-byte character set (DBCS). You can use DBCS characters in DB2 identifiers such as tables, columns, or view
names, or in the constants and field names that you specify in your SELECT statement.

- Too many concurrent tasks to a single output data set can degrade performance. When you use a single SELECT statement to unload a partitioned table space, use multiple unload data sets or use the MAXSORTS option to limit the number of concurrent tasks and reduce contention.

* (asterisk)

Specify an asterisk (*) to tell UNLOAD PLUS to select all columns in the table.

When DIRECT NO is in effect and you want to unload columns in your table that are defined as IMPLICITLY HIDDEN, you must explicitly specify the columns to unload. (For DIRECT YES, UNLOAD PLUS unloads hidden columns when you specify SELECT *.)

**columnName**

If you specify a column name, it must be a column in the named table. If any selected column uses a FIELDPROC, the routine must be included in your system’s LINKLIST, or your JOBLIB or STEPLIB.

**constant**

Specify a constant to have UNLOAD PLUS put a specific value in the output record. The constant must be a valid SQL integer, decimal, graphic, or character string, with a length of up to 255 bytes.

**CURRENT DATE, CURRENT TIME, or CURRENT TIMESTAMP**

Specify CURRENT DATE, CURRENT TIME, or CURRENT TIMESTAMP to have UNLOAD PLUS put the current date, time, or timestamp in the output record. UNLOAD PLUS derives this value from the system clock value. For each record, UNLOAD PLUS obtains a new system clock value.

You cannot specify CURRENT TIMESTAMP WITH TIME ZONE. However, if you are unloading to a TIMESTAMP WITH TIME ZONE column, UNLOAD PLUS includes the time zone from the DSNHDECP IMPLICIT_TIMEZONE value.

**CURRENT RID**

*This option is valid only when DIRECT YES is in effect.*
This option tells UNLOAD PLUS to include in your output file a field that contains the RID for the corresponding row in the named table. The output RID is either a 4- or 5-byte hexadecimal value, in the same format as an internal DB2 index entry.

**Restrictions**

This option has the following restrictions:

- BMC recommends that you do not include a field specification for the RID field on your INTO statement. Instead, specify a comma as a placeholder.
  - If you choose to include a field specification, the data type must be CHAR with a length of 4 or 5.

- UNLOAD PLUS treats the RID values as if they are defined as FOR BIT DATA. As with any FOR BIT DATA values, UNLOAD PLUS does not translate these values.

**Example**

The following simple example SELECT statement uses the CURRENT RID option:

```sql
SELECT CURRENT RID, *
INTO

  EMPNAME CHAR (20),
  SALARY DECIMAL EXTERNAL (9,2),
FROM EMPL.TB1
```

Assuming 4 byte-RIDs, a sample output record would be in the following format:

<table>
<thead>
<tr>
<th>RID</th>
<th>EMPNAME CHAR (20)</th>
<th>SALARY DECIMAL EXTERNAL (9,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>JOHN SMITH</td>
<td>50000.00</td>
</tr>
<tr>
<td>003</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**INTO**

The INTO option defines the output record.
You can omit the INTO option entirely for a simple unload, in which case UNLOAD PLUS uses the default field data type, format, and length for all selected items.

record options

The following options allow you to specify output record identification information.

For more information about generating and inserting additional syntax into utility control statements, see “Inserting additional command options into utility control statements” on page 500.

NAME

The NAME option allows you to override the default table name or file that UNLOAD PLUS generates from the control statements shown in Table 38 on page 209. When using DIRECT YES, the default name is the name of the table being unloaded. When using DIRECT NO, the default name is the userID.$TABLEn. To generate control statements with a name different from the default name, specify NAME ownerName.tableName.

Restriction

UNLOAD PLUS does not support Unicode table names.

Table 38: NAME option actions based on CNTLCARDS option

<table>
<thead>
<tr>
<th>CNTLCARDS option</th>
<th>NAME option action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLCARDS DB2LOAD (the default for CNTLCARDS)</td>
<td>Sets the DB2 table name that UNLOAD PLUS generates in the LOAD utility, LOADPLUS control statements, or CREATE DDL statements, respectively</td>
</tr>
<tr>
<td>CNTLCARDS BMCLOAD</td>
<td></td>
</tr>
<tr>
<td>CNTLCARDS DB2</td>
<td></td>
</tr>
<tr>
<td>CNTLCARDS option</td>
<td>NAME option action</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CNTLCARDS EASYTRIEVE</td>
<td>Sets the file name that UNLOAD PLUS generates for Easytrieve</td>
</tr>
<tr>
<td></td>
<td>If you specify CNTLCARDS EASYTRIEVE, the NAME option does not affect the file name that UNLOAD PLUS generates as Easytrieve input.</td>
</tr>
<tr>
<td>CNTLCARDS FOCUS</td>
<td>Sets the file name that UNLOAD PLUS generates for the FOCUS external file definition statement</td>
</tr>
<tr>
<td>CNTLCARDS SAS</td>
<td>Sets the file name that UNLOAD PLUS generates for the SAS INFILE</td>
</tr>
<tr>
<td></td>
<td>If you specify CNTLCARDS SAS, the NAME option does not affect the file name that UNLOAD PLUS generates as SAS input.</td>
</tr>
<tr>
<td>CNTLCARDS SQL/DS</td>
<td>Sets the SQL/DS table name that UNLOAD PLUS generates in the DATA LOAD utility control statements, or CREATE DDL statements, respectively</td>
</tr>
<tr>
<td>CNTLCARDS SQL/DS-LOAD</td>
<td></td>
</tr>
<tr>
<td>CNTLCARDS SQL/DS-DDL</td>
<td></td>
</tr>
<tr>
<td>CNTLCARDS TERADATA</td>
<td>Sets the table name that UNLOAD PLUS generates in the load utility control statements</td>
</tr>
<tr>
<td>CNTLCARDS NOMAD</td>
<td>Sets the file name that UNLOAD PLUS generates for the MASTER NOMAD INFILE</td>
</tr>
<tr>
<td></td>
<td>If you specify a two-part name, UNLOAD PLUS uses the second part for the master name. If you do not specify a name, you can use a comma as a placeholder.</td>
</tr>
</tbody>
</table>

### 'string'

This option allows you to insert additional load syntax to follow the INTO option for the IBM DB2 LOAD or the BMC LOADPLUS generated utility control statements when you specify CNTLCARDS DB2LOAD, CNTLCARDS DB2, or CNTLCARDS BMCLOAD. You can specify multiple strings of up to 72 bytes each. Enclose each string in single quotes. To embed a quoted string (a string enclosed by single quotation marks) within a string, add a single quotation mark before and after the quoted string. The following examples show the use of this option:

```
INTO 'PART 2:4 REPLACE'
INTO 'REPLACE WHEN EMPLDATE > ''2005-01-01'''
```

When loading system-period temporal objects, you might need to add syntax to the load statements that UNLOAD PLUS generates. You can either add the syntax as a string on the UNLOAD PLUS INTO option, or you can modify the load control cards that UNLOAD PLUS generated before you run the load job. The following table describes which strings to add for which conditions:
Table 39: Strings to add to INTO statement for temporal objects

<table>
<thead>
<tr>
<th>Condition</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>You specify FORMAT BMCLOAD.</td>
<td>'PERIODOVERRIDE TRANSIDOVERRIDE'</td>
</tr>
<tr>
<td>You want to load input file data into a system-period begin or end column</td>
<td>'PERIODOVERRIDE'</td>
</tr>
<tr>
<td>You want to load input file data into a transaction-start-ID column</td>
<td>'TRANSIDOVERRIDE'</td>
</tr>
</tbody>
</table>

RECORDID 'string'

You can use the RECORDID option to specify a value for a record identifier. The RECORDID is a character string constant that can be up to 72 bytes long, and is placed at the front of each output record for the SELECT statement.

Restrictions

Note the following restrictions when specifying the RECORDID option:

- If you also specify FORMAT BMCLOAD, UNLOAD PLUS terminates.
- If you also specify FORMAT XML, UNLOAD PLUS ignores the RECORDID option.

Additional considerations

The following considerations apply to the RECORDID option:

- If UNLOAD PLUS generates CNTLCARDS for DB2 or DB2LOAD, the utility generates appropriate WHEN syntax for the DB2 load control statements.
- If you also specify AUTOTAG YES, UNLOAD PLUS ignores the AUTOTAG option for that SELECT statement.

field specification

The field specification defines the fields of the output record. See “field specification” on page 223 for a complete description, including restrictions.

FROM

FROM specifies the table or view from which UNLOAD PLUS unloads data.
You can specify only one table or view name. If you do not specify the creator name, UNLOAD PLUS uses the DB2 primary authorization ID of the user executing UNLOAD PLUS. The table name or view name can be an alias or a synonym.

**Restrictions**

Note the following restrictions when specifying FROM:

- UNLOAD PLUS does not support the following specifications:
  - Three-part object names
  - Correlation names
  - Unicode table names
- You cannot specify an auxiliary or XML table. (To unload LOB or XML data, specify the base table.)
- When you specify the FORMAT BMCLOAD option, you cannot specify a catalog table.
- UNLOAD PLUS supports unloading from clone tables. However, you cannot unload from a clone table and a base table in the same unload job step.
- With DIRECT YES, UNLOAD PLUS supports unloading from views except for the following definitions:
  - Views that you created with the keyword DISTINCT in the subselect
  - Views that you created with a WHERE clause in the subselect
  - Views that you created with a GROUP BY clause in the subselect
  - Views that you created with a HAVING clause in the subselect
  - View columns derived from functions or expressions
  - Views that you created with a join in the view subselect
  - Views that you created with a union in the view fullselect
Note
DIRECT NO supports the full range of DB2 SQL SELECT functionality and view definitions.

- Only the authorization ID that created an unqualified synonym can access it.

**OBID**

The OBID option specifies the OBID that UNLOAD PLUS can use to find the rows for the selected table when unloading using the INFILE *ddname* option. Normally, UNLOAD PLUS uses the OBID found in the DB2 catalog for the selected table to find the rows. This option allows you to override the OBID that UNLOAD PLUS finds in the DB2 catalog. This option is useful when the OBID in the DB2 catalog for the selected table no longer matches the OBID that is associated with the table as it exists in the image copy. This option is valid only when you specify the INFILE *ddname* option.

You can specify the OBID option when selecting data from tables or views. When UNLOAD PLUS uses the specified OBID with a view, the specified OBID overrides the OBID that UNLOAD PLUS found in the DB2 catalog for the "viewed" table. You can specify the OBID as an integer or as a hexadecimal string, and you must use either parentheses or braces.

To specify the OBID, you must either use this option or specify the OBID in the DDL that the DDLIN data set provides.

When the following conditions exist, you must either specify this option (with each SELECT statement) or specify the OBID in each CREATE TABLE statement in the DDLIN data set:

- You specify INFILE *ddname* with DDL (using a DDLIN data set).
- You are either specifying multiple SELECT statements or unloading a multi-table table space.

**WHERE condition**

WHERE specifies the condition that must be true before UNLOAD PLUS can select a specific row.
A condition is a combination of predicates that use AND, OR, NOT, and () operators. The following table shows the result of the WHERE condition when you use the AND, OR, or NOT operator:

Table 40: Truth table for AND, OR, NOT

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p AND q</th>
<th>p OR q</th>
<th>NOT p</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

UNLOAD PLUS evaluates predicates from left to right, with the following operator precedence:

- ()
- NOT
- AND
- OR

When UNLOAD PLUS determines the resulting value of the condition, the utility stops further evaluation of any remaining predicates. If the result is true, UNLOAD PLUS selects the row. If the result is false, UNLOAD PLUS does not select the row.

Restriction

When unloading LOB or XML data, you can specify the WHERE clause only for a NULL condition.
**predicate**

A predicate is a simple comparison of a single column value to a constant or list of constants.

The column name must be a column of the named table, and the length of the column must be less than or equal to 255. You can use a predicate on a WHERE clause or with the IF option in a field specification.

**comparison operator**

UNLOAD PLUS performs the comparison with the DB2 internal column value, which includes FIELDPROC encoding and DB2 normalization. You can use the comparison operators shown in the following table:

---

The column name must be a column of the named table, and the length of the column must be less than or equal to 255. You can use a predicate on a WHERE clause or with the IF option in a field specification.

**comparison operator**

UNLOAD PLUS performs the comparison with the DB2 internal column value, which includes FIELDPROC encoding and DB2 normalization. You can use the comparison operators shown in the following table:
Table 41: Comparison operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>¬&lt;</td>
<td>Not less than</td>
</tr>
<tr>
<td>&lt; =</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>&lt; &gt;</td>
<td>Not equal</td>
</tr>
<tr>
<td>¬ =</td>
<td>Not equal</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>¬&gt;</td>
<td>Not greater than</td>
</tr>
<tr>
<td>&gt; =</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td>Equal to any</td>
</tr>
<tr>
<td>NOT IN</td>
<td>Not equal to all</td>
</tr>
<tr>
<td>IS NULL</td>
<td>Column is null</td>
</tr>
<tr>
<td>IS NOT NULL</td>
<td>Column is not null</td>
</tr>
</tbody>
</table>
| LIKE 'string' | Valid only in conjunction with the WHERE clause | This operator is a subset of the DB2 LIKE operator. Only CHARACTER, VARCHAR, BINARY, VARBINARY, GRAPHIC, and VARGRAPHIC columns are allowed. UNLOAD PLUS treats double quotes within a string as a single quote, and interprets this single quote as a literal character.

**Restrictions**

The following restrictions apply when using a predicate with the IF option. For more considerations when using a predicate with the IF option, see the description of the IF option.

- UNLOAD PLUS does not support CURRENT DATE, CURRENT TIMESTAMP, and LIKE for the IF *predicate* option in the field specifications.

- When DIRECT NO is in effect, you can use only the equal or not equal to operators to compare the following types of columns: numeric, DATE in USA or EUR formats, or TIME in USA format.

**ESCAPE 'character'**

Use this option to designate a single character as an escape character that you can use one or more times within your LIKE string to specify the literal occurrence of an underscore (\_) or a percent sign (%). To use this character in your LIKE string, put the escape character before the underscore or
percent sign. If you do not include an ESCAPE clause, you have not specified an escape character. When you include an ESCAPE clause, UNLOAD PLUS recognizes underscores and percent signs as wildcard characters.

**constant**

The constant specifies a value to be compared to the column value. UNLOAD PLUS then evaluates the result based on the preceding relative operator. For the comparison, the associated constant must match the data type of the column (that is, numeric to numeric, string to string, and date/time to date/time).

**Additional considerations**

Table 42 on page 217 describes the requirements for constants in predicates. In addition, consider the following information:

- For the comparison, if the string is shorter than the column length, UNLOAD PLUS pads the string with blanks. However, UNLOAD PLUS processes VARCHAR comparisons like DB2 processes them.

- Although not prevented or restricted by DB2, BMC recommends that you avoid nonstandard comparisons. See rules for constants in the IBM DB2 SQL Reference.

- If you specify a list of constants, you can use ascending sequential order to improve performance. When you specify the list in this order, you save processing time because UNLOAD PLUS does not sort the list.

- For more considerations when using constants with the IF option, see the description of the IF option.

**Table 42: Rules for constants**

<table>
<thead>
<tr>
<th>Type of constant</th>
<th>Additional details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Identical to DB2 SQL integer constants</td>
</tr>
<tr>
<td></td>
<td>The constant must be within the range of the column’s data type.</td>
</tr>
<tr>
<td>Type of constant</td>
<td>Additional details</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Decimal</td>
<td>Identical to DB2 SQL decimal constants&lt;br&gt;The constant must be within the range of the column’s data type.&lt;br&gt;Note: To determine whether your site has set the decimal point to a comma, check the DB2 subsystem DSNHDECP setting. To enable UNLOAD PLUS to distinguish comma decimal points from comma separators, include a space after any comma that you use as a separator.&lt;br&gt;Example of using the comma as a decimal point:&lt;br&gt;<code>SELECT * FROM ownerName.tableName WHERE SALARY &gt; 100,00</code>&lt;br&gt;Example using the comma as both a decimal point and a separator:&lt;br&gt;<code>SELECT * FROM ownerName.tableName WHERE SALARY IN (100,00, 200,00, 300,00)</code></td>
</tr>
<tr>
<td>Character string</td>
<td>Identical to DB2 SQL strings except that the length is limited to 255 bytes&lt;br&gt;Use '' to denote an empty string.&lt;br&gt;Note: UNLOAD PLUS does not validate character data against CCSIDs.</td>
</tr>
<tr>
<td>Hexadecimal strings</td>
<td>Identical to DB2 SQL hexadecimal strings with the following exceptions:&lt;br&gt;■ The length is limited to 255 bytes.&lt;br&gt;■ You can specify hexadecimal strings for a binary column.&lt;br&gt;Note:&lt;br&gt;■ UNLOAD PLUS does not validate hexadecimal string data against CCSIDs.&lt;br&gt;■ UNLOAD PLUS does not support the hexadecimal graphic string constants UX' xxxx' and GX' xxxx'.</td>
</tr>
<tr>
<td>Binary strings</td>
<td>Identical to DB2 SQL binary string constants except that the length is limited to 255</td>
</tr>
<tr>
<td>Graphic string</td>
<td>Identical to DB2 SQL graphic string constants except that the length is limited to 255</td>
</tr>
<tr>
<td>Date/time string</td>
<td>With the following exceptions, identical to DB2 SQL date/time strings&lt;br&gt;Exceptions:&lt;br&gt;■ Use only ISO, USA, EUR, and JIS formats.&lt;br&gt;■ Precision on a timestamp constant must be less than or equal to the precision defined in the column.&lt;br&gt;Note: If you specify a time zone on a timestamp constant, this value overrides any value that you specify for the IMPLICIT_TZ option.</td>
</tr>
</tbody>
</table>
NULL

You can specify the keyword NULL as a constant for columns that are set to null. You can specify NULL with any of the allowed operators. The null value is always higher than all other values. When a column is set to NULL, a null indicator field follows each nullable field in the output. See “Example 1 - Unloading partitions to separate data sets” on page 314 for sample output.

Consideration

If a column value is NULL, a predicate (other than IS/IS NOT or where the constant is equal to NULL) does not evaluate to "unknown" as it does in DB2. Normally this produces the same result as in DB2, because UNLOAD PLUS treats an "unknown" value for a predicate as "false." However, using NOT with a predicate that evaluates to "unknown" is "true," which is different in DB2.

CURRENT DATE - labeledDuration

This option describes the current date or, optionally, the current date minus a duration in days, months, or years. The rules for date/time arithmetic using labeled durations are identical to the DB2 SQL rules. UNLOAD PLUS takes the current date from the local time-of-day clock during the UTILINIT phase.

CURRENT TIMESTAMP - labeledDuration

This option describes the current timestamp or, optionally, the current timestamp minus a duration in days, months, or years. The rules for date/time arithmetic using labeled durations are identical to the DB2 SQL rules. UNLOAD PLUS takes the current timestamp from the local time-of-day clock during the UTILINIT phase.

You cannot specify CURRENT_TIMESTAMP WITH TIME_ZONE. However, if you are comparing the current timestamp to a TIMESTAMP WITH TIME_ZONE column, UNLOAD PLUS uses the time zone from the DSNHDECP IMPLICIT_TIMEZONE value.

ORDER BY

The ORDER BY option allows you to order the rows of the unloaded table according to the specified columns.
Alternatively, if you are ordering by the data-sorting key, you can use the ORDER YES option (“ORDER” on page 139). ORDER BY overrides ORDER YES.

Restriction

UNLOAD PLUS terminates when either of the following conditions exists:

- You specify a LOB or XML column on the ORDER BY option.
- You specify SPANNED YES.

Additional considerations

Note the following considerations when using the ORDER BY option:

- The encoding scheme of the table determines the collating sequence for character data. For example, if the table is in EBCDIC, the collating sequence is EBCDIC.
- ORDER BY sets MAXSORTS to 1 and assigns all partitions to a single task when any of the following conditions exists:
  - The columns that you specify in the ORDER BY option are different from the partitioning key columns.
  - The columns that you specify in the ORDER BY option are the same or a subset of the partitioning index key columns but are in a different order.
  - You are unloading a partition-by-growth table space.
- Ensure that UNLOAD PLUS has adequate sort work space for sorting data in all partitions. For information about sizing sort work data sets, see “SORTWK data sets in UNLOAD PLUS” on page 281.
- When you specify ORDER BY and INTO, UNLOAD PLUS uses the data type of the column that is defined in the DB2 table to sort the data. UNLOAD PLUS does not sort on the data type that you specify in the INTO statement.

columnName

Identifies a column of the specified table that you want to use for ordering.
ASC

ASC is the default, and specifies that UNLOAD PLUS uses the values of the column in ascending order.

DESC

DESC specifies that UNLOAD PLUS uses the values of the column in descending order.

ESTROWS

One of the methods that UNLOAD PLUS can use to calculate the size of the data sets that are produced during dynamic allocation is an ESTROWS number that you specify on a SELECT statement.

The integer is the number of rows that you expect that UNLOAD PLUS will unload using that SELECT statement. See “ANALYZE” on page 123.

You must specify the ESTROWS, LIMIT, or SPACE command option to supply sizing information for dynamic allocation in either of the following cases:

- DIRECT NO is in effect.
- You specify INFILE.

If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

OPTIONS

OPTIONS provides support for options that apply to a particular SELECT statement.

This option is valid only when DIRECT YES is in effect.

Options specified within the SELECT OPTIONS statement override their global command option equivalents.
PART or LOGICAL PART

Specify PART followed by integer values from 1 through 4096, or a range of values. The values identify the partitions of the partitioned table space that UNLOAD PLUS will unload for the corresponding SELECT statement.

Optionally, you can specify the LOGICAL keyword with PART to indicate that you are specifying the logical partitions to unload rather than the physical partitions.

Specifying PART within a SELECT OPTIONS statement overrides the PART and FILTERPART command options for the corresponding SELECT statement.

Specifying partition numbers

Note the following information about how to specify partition numbers with the PART option:

- Individual partitions in a list can be in any order (but partitions within a range must be in ascending order).
- You can specify a mixture of individual partitions and ranges of partitions.
- If you specify a partition number more than once, UNLOAD PLUS ignores any occurrence after the first.

In the following example, specifying PART 1, 3, 4, 5 within the SELECT OPTIONS statement overrides the PART 2 global command option. Only partitions 1, 3, 4, and 5 will be unloaded.

```
UNLOAD PART 2
SELECT * FROM MY.TABLE
OPTIONS (PART 1, 3, 4, 5)
;
```

You can also specify a range of values as shown in the following example. Specifying PART 1:5 (as shown in the following example) within the SELECT OPTIONS statement overrides the PART 2 global command option and unloads partitions 1, 2, 3, 4, and 5.

```
UNLOAD PART 2
SELECT * FROM MY.TABLE
OPTIONS (PART 1:5)
;
```

Dynamic allocation

Specifying PART also determines which data sets UNLOAD PLUS dynamically allocates for the corresponding SELECT statement. In the following example, one
data set will be generated for the SELECT statement (USER.P001) and only the first partition in the partitioned table space will be unloaded.

```
UNLOAD
UNLOADDN(SYSREC) ACTIVE (YES)
OUTPUT SYSREC DSNAME 'USER.P&PART'
SELECT * FROM MY.PART3TB
OPTIONS (PART 1)
;
```

Restrictions

The following restrictions apply to the PART option:

- UNLOAD PLUS terminates if you specify both logical and physical partitions in the same job.

- If you specify INFILE ddname and supply DDL (using a DDLIN data set), UNLOAD PLUS ignores the LOGICAL keyword and considers the specified partitions to be physical partitions.

```
; (semicolon)
```

Specify the semicolon to end the SELECT statement.

For DIRECT YES, UNLOAD PLUS does not require a semicolon when you specify the UNLOAD PLUS SELECT-like syntax. For DIRECT NO with multiple SELECT statements that use options that are not available in the UNLOAD PLUS SELECT-like syntax, you must end each SELECT statement with a semicolon.

field specification

The field specification option defines a field in the output record.
UNLOAD PLUS options

field specification, part 1

fieldName

<table>
<thead>
<tr>
<th>CHAR</th>
<th>MIXED</th>
<th>FILL</th>
<th>NO</th>
<th>TRUNCATE</th>
<th>TRIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR</td>
<td></td>
<td></td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>outputDescriptorName</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOBF</td>
<td>outputDescriptorName</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>EXTERNAL</td>
<td>(length)</td>
<td></td>
<td>TRUNCATE</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>(length)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>(length)</td>
<td>ROUND</td>
<td>FILL</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>INTEGER</td>
<td>(length)</td>
<td>ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>EXTERNAL</td>
<td>(length, scale)</td>
<td></td>
<td>TRUNCATE</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>EXTERNAL</td>
<td>(length)</td>
<td>CENTURY (ccyy, ccyy)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>EXTERNAL</td>
<td>(length)</td>
<td>CENTURY (ccyy, ccyy)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>EXTERNAL</td>
<td>(length)</td>
<td>CENTURY (ccyy, ccyy)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>WITH TIME ZONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>(precision)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>EXTERNAL</td>
<td>(length)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>PARM</td>
<td></td>
<td>constant</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TRIM and TRUNCATE are not valid for BLOBF, CLOBF, or DBCLOBF

See decimal specification block detail
The field specification identifies the following information:

- The field name
- (optional) The data type, data format, and data length of the field
- (optional) The value to assign to the field under certain conditions

You do not have to specify a field for each SELECT list item, but, if you do not include a field specification for an item, you must specify a comma (,) as a placeholder. In this case, the description of the field defaults to that of the SELECT list item. If the INTO list has too many fields, UNLOAD PLUS terminates. If the INTO list has too few, the fields default to that of the SELECT list item.

**Restrictions**

Note the following restrictions when including a field specification:

- NAME and RECORDID are reserved words that you cannot use to name fields. If you want to use SQL reserved words or UNLOAD PLUS command options as object names, you must delimit them with quotation marks to prevent syntax errors.
If you are using FORMAT BMCLOAD, you cannot use the field specification list.

If you include a field specification and you also specify FORMAT CSV with the RTRIM option, UNLOAD PLUS ignores the RTRIM option for the associated column when any of the following conditions exists:

- The column is defined with a data type other than CHAR or VARCHAR.
- For a CHAR or VARCHAR column
  - You specify a length, but the specified length differs from the actual length of the data.
  - You specify any of the other keywords allowed for a CHAR or VARCHAR column.

If you specify SELECT CURRENT RID, BMC recommends that you do not include a field specification for the RID field. Instead, specify a comma as a placeholder. If you choose to include a field specification, the data type must be CHAR with a length of 4 or 5.

**fieldName**

The field name identifies a field in the output record. The field name does not have to match the corresponding column name in the SELECT list if a column name is used. If the column is nullable and you do not specify the IF NULL option, a one-byte character field follows immediately after the specified field. This one-byte field is set to X'00' if the value is not null and is set to a question mark (?) if it is null.

**dataType**

The data type option specifies the data type and format of the output value and can also indicate the length of the output value. If you do not specify the data type of the field, UNLOAD PLUS uses the data type and data length of the column or constant from the SELECT list. UNLOAD PLUS performs no conversions in this case, except for denormalization and FIELDPROC decoding.

The default data format depends on the FORMAT option that is in effect. For example, when FORMAT STANDARD (the default) is in effect, the default data format is DB2 internal, except for date, time, and timestamp columns, which default to external format.

For details about valid data types and general rules for data types, see “Data type keywords in UNLOAD PLUS” on page 229. For allowable conversions, see “Supported data type conversions in UNLOAD PLUS” on page 248.
**IF**

The IF option determines the value of the field based on the specified condition.

**Restrictions**

The following restrictions apply to the IF option:

- UNLOAD PLUS terminates when you specify the IF option and you are unloading LOB or XML data.

- UNLOAD PLUS does not support CURRENT DATE, CURRENT TIMESTAMP, and LIKE for the IF *predicate* option.

- When DIRECT NO is in effect, you can use only the equal or not equal comparison operators to compare the following types of columns:
  - Numeric
  - DATE in USA or EUR formats
  - TIME in USA format

- When DIRECT NO is in effect, date or timestamp constants must use a forward slash (/) as the date separator.

**Additional considerations**

Although UNLOAD PLUS uses the standard rules for comparison with the majority of predicate block constants, DIRECT YES and DIRECT NO have different behaviors when used with the IF predicate:

- When using DIRECT YES, IF predicate comparisons are made against the raw DB2 row data and predicate evaluation behavior is standard.

- When using DIRECT NO, IF predicate comparisons are made against the DB2 dynamic SQL host variable data.

Depending on your subsystem configuration, some IF predicate comparison results will differ between DIRECT YES and DIRECT NO, specifically those that involve the following types of comparisons:

- DATE, TIME, and TIMESTAMP fields that are subject to local formatting options

- FOR BIT DATA fields and hexadecimal constants in an environment where more than one character code set is used
NULL

The IF NULL option determines the value of the field if the column value is null. The column must be nullable; if you specify IF NULL on the field, UNLOAD PLUS does not generate a null indicator on the field. If you use the IF NULL option, you cannot use the NULL value or any related predicates.

ERROR

The IF ERROR option determines the value of the field if an error occurs when converting the column value to the field’s data type.

predicate

The IF predicate option determines the value of the field if UNLOAD PLUS determines that the predicate is true. See the description of the WHERE predicate option for rules about specifying the predicate.

VALUE

This option specifies that you want to place a constant value, null, current date, current time, or current time stamp in the field if the condition is true.

constant

You can use only integer, decimal, character string, hexadecimal string, binary string or graphic string constants. UNLOAD PLUS does not support floating-point constants. The constants must match the field’s data type (that is, numeric to numeric, string to string, and date/time to date/time). However, you can use a hexadecimal string with a binary field.

Additional considerations

Note the following considerations:

- Character, hexadecimal, binary, and graphic string constants cannot exceed 255 bytes.

- Decimal string constants require a decimal point.

- If you use CHARACTER for a numeric or date, time, or timestamp column, the value’s data type must also be CHARACTER. Conversely, if you use type EXTERNAL, the constant’s value must be a value that is compatible with the data type.
NULL

This option specifies that you want a NULL value for the field if the condition is true. You may use NULL only if the corresponding column is nullable and you did not include IF NULL on the field specification.

CURRENT DATE

This option specifies the current date. The field must be the DATE data type.

CURRENT TIME

This option specifies the current time. The field must be the TIME data type.

CURRENT TIMESTAMP

This option specifies the current timestamp. The field must be the TIMESTAMP or TIMESTAMP WITH TIME ZONE data type.

You cannot specify CURRENT TIMESTAMP WITH TIME ZONE. However, if you are unloading to a TIMESTAMP WITH TIME ZONE column, UNLOAD PLUS includes the time zone from the DSNHDECP IMPLICIT_TIMEZONE value.

Data type keywords in UNLOAD PLUS

This section describes the keywords that you can use to specify data types.

For more information about data types, see the following references:

- “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247
- “Supported data type conversions in UNLOAD PLUS” on page 248
- “Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS” on page 259
- “Additional numeric field considerations in UNLOAD PLUS” on page 260
- “Additional date, time, or timestamp field considerations in UNLOAD PLUS” on page 261
- “Additional variable field considerations in UNLOAD PLUS” on page 261
- “Additional ROWID field considerations in UNLOAD PLUS” on page 261
CHAR

CHAR (or CHARACTER) defines a fixed-length character field.

If you use CHAR on a column that is not defined as CHAR or VARCHAR, it is (almost) identical to specifying type EXTERNAL.

MIXED

MIXED defines a mixed character field.

FILL

FILL tells UNLOAD PLUS how to handle leading zeros when converting data from a source with numeric data types.

YES

FILL YES externalizes all leading zeros when converting numeric data types to their external representations. The first character in the external representation of the number is either a minus sign (-) or a plus sign (+). This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

NO

FILL NO removes all leading zeros when converting numeric data types to their external representations.

BLOBF

BLOBF defines a field that contains the name of a binary large object (BLOB) file to which you are unloading LOB data. You must specify the name of the output descriptor that you defined in your OUTPUT statement for this file. For more information, see the description of the OUTPUT option.

UNLOAD PLUS ignores this field specification if you specify SPANNED YES.

BINARYXML

This option is valid only when DIRECT YES is in effect.

BINARYXML indicates that you are unloading XML data to the BLOB file. When generating LOAD control cards, UNLOAD PLUS adds the BINARYXML option to the field specification for the XML column.
CLOBF

CLOBF defines a field that contains the name of a character large object (CLOB) file to which you are unloading LOB or XML data. You must specify the name of the output descriptor that you defined in your OUTPUT statement for this file. For more information, see the description of the OUTPUT option.

UNLOAD PLUS ignores this field specification if you specify SPANNED YES.

DBCLOBF

DBCLOBF defines a field that contains the name of a double-byte character large object (DBCLOB) file to which you are unloading LOB data. You must specify the name of the output descriptor that you defined in your OUTPUT statement for this file. For more information, see the description of the OUTPUT option.

Restrictions

The following restrictions apply to the DBCLOBF field specification:

- You cannot unload XML data to a DBCLOB file.
- UNLOAD PLUS ignores this field specification if you specify SPANNED YES.

(length)

This option specifies the number of single-byte characters.

When used with the BLOBF, CLOBF, or DBCLOBF subtype, the length specifies the size of the expanded output descriptor name in bytes. Note the following considerations with these subtypes:

- If you specify a length that is less than the size of the expanded output descriptor name, UNLOAD PLUS terminates.
- If you specify a length that is greater than the expanded output descriptor name, UNLOAD PLUS pads the remaining length with X'40' characters.
- If you do not specify a length, UNLOAD PLUS defaults to the following values:
  - 54 for DSNTYPE=PDS or DSNTYPE=LIBRARY
  - 255 for DSNTYPE=HFS
TRIM

TRIM tells UNLOAD PLUS to remove as many trailing blanks from the string as needed to make the string length match the length that you specify.

UNLOAD PLUS performs the TRIM function before attempting to assign the value to the field. If the string is still too long, a conversion error occurs. UNLOAD PLUS applies TRIM before TRUNCATE.

TRIM is useful in the following scenarios:

- When you are converting from a CHAR column to a VARCHAR field
- When you are converting from a CHAR column to a shorter CHAR field, and you want to detect (by way of conversion error messages) nonblank data beyond position \( n \)

Restrictions

UNLOAD PLUS ignores the TRIM option in either of the following cases:

- You specify FORMAT DSNTIAUL.
- You use the BLOBF, CLOBF, or DBCLOBF subtype.

TRUNCATE

TRUNCATE tells UNLOAD PLUS to truncate a string if it is longer than the field (even after the TRIM option has truncated trailing blanks). UNLOAD PLUS performs TRUNCATE after TRIM.

UNLOAD PLUS ignores TRUNCATE for BLOBF, CLOBF, and DBCLOBF subtypes.

VARCHAR

VARCHAR defines a character string field that varies in length.

For more information, see “Additional variable field considerations in UNLOAD PLUS” on page 261.

MIXED

MIXED defines a mixed character field.
FILL

FILL tells UNLOAD PLUS how to handle leading zeros when converting data from a source with numeric data types. This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

YES

When you specify FILL YES, UNLOAD PLUS externalizes all leading zeros when converting numeric data types to their external representations. The first character in the external representation of the number is either a minus sign (-) or a plus sign (+). This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

NO

When you specify FILL NO, UNLOAD PLUS removes all leading zeros when converting numeric data types to their external representations.

BLOBF

BLOBF defines a field that contains the name of a binary large object (BLOB) file to which you are unloading LOB data. You must specify the name of the output descriptor that you defined in your OUTPUT statement for this file. For more information, see the description of the OUTPUT option.

UNLOAD PLUS ignores this field specification if you specify SPANNED YES.

BINARYXML

This option is valid only when DIRECT YES is in effect.

BINARYXML indicates that you are unloading XML data to the BLOB file. When generating LOAD control cards, UNLOAD PLUS adds the BINARYXML option to the field specification for the XML column.

CLOBF

CLOBF defines a field that contains the name of a character large object (CLOB) file to which you are unloading LOB or XML data. You must specify the name of the output descriptor that you defined in your OUTPUT statement for this file. For more information, see the description of the OUTPUT option.

UNLOAD PLUS ignores this field specification if you specify SPANNED YES.
**DBCLOBF**

DBCLOBF defines a field that contains the name of a double-byte character large object (DBCLOB) file to which you are unloading LOB data. You must specify the name of the output descriptor that you defined in your OUTPUT statement for this file. For more information, see the description of the OUTPUT option.

**Restrictions**

The following restrictions apply to the DBCLOBF field specification:

- You cannot unload XML data to a DBCLOB file.
- UNLOAD PLUS ignores this field specification if you specify SPANNED YES.

***(length)***

The length indicates the number of single-byte characters. A 2-byte length field that contains the length of the character string in bytes precedes the VARCHAR string. This length does not include the two length bytes. For calculating length when converting from a numeric column, see “Supported data type conversions in UNLOAD PLUS” on page 248.

When used with the BLOBF, CLOBF, or DBCLOBF subtype, the length specifies the size of the expanded output descriptor name in bytes. Note the following considerations with these subtypes:

- If you specify a length that is less than the size of the expanded output descriptor name, UNLOAD PLUS terminates.
- If you do not specify a length, UNLOAD PLUS defaults to the following values:
  - 54 for DSNTYPE=PDS or DSNTYPE=LIBRARY
  - 255 for DSNTYPE=HFS

**TRIM**

TRIM tells UNLOAD PLUS to remove as many trailing blanks from the string as needed to make the string length match the length that you specify. If the string is still too long, a conversion error occurs. UNLOAD PLUS applies TRIM before TRUNCATE.

TRIM is useful when you are converting from a VARCHAR column to a CHAR($n$) or VARCHAR($n$) field and you want to detect (by way of conversion error messages) nonblank data beyond position $n$.

**Restrictions**
UNLOAD PLUS ignores the TRIM option in either of the following cases:

- You specify FORMAT DSNTIAUL.
- You use the BLOBF, CLOBF, or DBCLOBF subtype.

**TRUNCATE**

TRUNCATE tells UNLOAD PLUS to truncate a string if it is longer than the field (even after the TRIM option has truncated trailing blanks). UNLOAD PLUS performs TRUNCATE after TRIM.

UNLOAD PLUS ignores TRUNCATE for BLOBF, CLOBF, and DBCLOBF subtypes.

**GRAPHIC**

This keyword defines a graphic field.

**GRAPHIC(length)**

GRAPHIC indicates an internal graphic value without the shift-in and shift-out characters. Length indicates the number of double-byte characters.

**GRAPHIC EXTERNAL(length)**

GRAPHIC EXTERNAL indicates an external graphic value with the shift-out and shift-in characters. Length indicates the number of double-byte characters.

**TRUNCATE**

TRUNCATE tells UNLOAD PLUS to truncate a string if it is longer than the field.

**VARGRAPHIC (length)**

This keyword defines a variable-length graphic string field.

A 2-byte length field that contains the length of the graphic string in double-byte characters precedes the string. The length does not include the two length bytes. For more information, see “Additional variable field considerations in UNLOAD PLUS” on page 261.

**TRUNCATE**

TRUNCATE tells UNLOAD PLUS to truncate a string if it is longer than the field.
SMALLINT

SMALLINT defines a small integer numeric field.

For more information, see “Additional numeric field considerations in UNLOAD PLUS” on page 260.

SMALLINT(length, scale)

SMALLINT indicates an internal 2-byte integer value. If you specify the length, you must specify 2. Specifying scale multiplies or divides the number by a power of 10.

SMALLINT EXTERNAL(length, scale)

EXTERNAL indicates a string that contains a character representation of an integer constant. This data type is identical to INTEGER EXTERNAL except that when you use CNTLCARDS DB2 or CNTLCARDS DB2DDL, the CREATE TABLE column type is SMALLINT. For more information, see “Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS” on page 259.

ROUND

ROUND tells UNLOAD PLUS to round the value, if necessary, during conversion. If you do not specify ROUND, UNLOAD PLUS truncates any digits that it does not need for precision.

FILL

FILL tells UNLOAD PLUS how to handle leading zeros when converting data to SMALLINT EXTERNAL. Specify YES or NO:

- FILL YES externalizes all leading zeros when converting numeric data types to their external representations. The first character in the external representation of the number is either a minus sign (-) or a plus sign (+). This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

- FILL NO removes all leading zeros when converting numeric data types to their external representations.

INTEGER

This keyword defines an integer numeric field.

You can abbreviate INTEGER to INT. For more information, see “Additional numeric field considerations in UNLOAD PLUS” on page 260.
**INTEGER**(*length*, *scale*)

INTEGER indicates an internal 4-byte integer value. If you specify the length, you must specify 4. Specifying scale multiplies or divides the number by a power of 10.

**INTEGER EXTERNAL**(*length*, *scale*)

EXTERNAL indicates a string that contains a character representation of an integer constant. For more information, see “Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS” on page 259.

**ROUND**

ROUND tells UNLOAD PLUS to round the value, if necessary, during conversion. If you do not specify ROUND, UNLOAD PLUS truncates any digits that it does not need for precision.

**FILL**

FILL tells UNLOAD PLUS how to handle leading zeros when converting data to INTEGER EXTERNAL. Specify YES or NO:

- FILL YES externalizes all leading zeros when converting numeric data types to their external representations. The first character in the external representation of the number is either a minus sign (-) or a plus sign (+). This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

- FILL NO removes all leading zeros when converting numeric data types to their external representations.

**BIGINT**

BIGINT defines a big integer numeric field.

For more information, see “Additional numeric field considerations in UNLOAD PLUS” on page 260.

**BIGINT**(*length*, *scale*)

BIGINT indicates an internal 8-byte integer value. If you specify the length, you must specify 8. Specifying scale multiplies or divides the number by a power of 10.

**BIGINT EXTERNAL**(*length*, *scale*)

EXTERNAL indicates a string that contains a character representation of an integer constant. This data type is identical to INTEGER EXTERNAL except that when you
use CNTLCARDS DB2 or CNTLCARDS DB2DDL, the CREATE TABLE column type is BIGINT. For more information, see “Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS” on page 259.

ROUND

ROUND tells UNLOAD PLUS to round the value, if necessary, during conversion. If you do not specify ROUND, UNLOAD PLUS truncates any digits that it does not need for precision.

FILL

FILL tells UNLOAD PLUS how to handle leading zeros when converting data to BIGINT EXTERNAL. Specify YES or NO:

- FILL YES externalizes all leading zeros when converting numeric data types to their external representations. The first character in the external representation of the number is either a minus sign (−) or a plus sign (+). This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

- FILL NO removes all leading zeros when converting numeric data types to their external representations.

DECIMAL

This keyword defines a decimal numeric field.

You can abbreviate DECIMAL to DEC. For more information, see “Additional numeric field considerations in UNLOAD PLUS” on page 260.

DECIMAL PACKED\((\text{precision}, \text{scale})\)

DECIMAL PACKED is the default, which indicates that the output value is packed decimal with precision and scale. The precision represents the total number of digits; the scale represents the number of digits in the fractional part of the number.

DECIMAL\((\text{precision}, \text{scale})\)

This option is identical to DECIMAL PACKED.

DECIMAL ZONED\((\text{precision}, \text{scale})\)

This option specifies that the output value is decimal-zoned with precision and scale. The precision represents the total number of digits; the scale represents the number of digits in the fractional part of the number.
Note

To generate the absolute value of an output field or to ensure that all output fields can be printed, specify the installation option ZONEDDECVP=(F,F).

DECIMAL EXTERNAL(*length, *scale*)

EXTERNAL indicates a string that contains a character representation of a decimal constant. A decimal point is always placed in the string based on the scale specification. For more information, see “Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS” on page 259.

ROUND

ROUND tells UNLOAD PLUS to round the value, if necessary, during conversion. If you do not specify ROUND, UNLOAD PLUS truncates any digits that it does not need for precision.

FILL

FILL tells UNLOAD PLUS how to handle leading zeros when converting data to DECIMAL EXTERNAL. Specify YES or NO:

- FILL YES externalizes all leading zeros when converting numeric data types to their external representations. The first character in the external representation of the number is either a minus sign (-) or a plus sign (+). This capability supports applications such as COBOL that require leading zeros in order to edit and check unloaded data properly.

- FILL NO removes all leading zeros when converting numeric data types to their external representations.

FLOAT

FLOAT defines a floating-point numeric field.

For more information, see “Additional numeric field considerations in UNLOAD PLUS” on page 260.

FLOAT(*length, *scale*)

If you specify the length, the value must be in one of the following ranges:

- For a 4-byte internal floating-point number, the value must be between 1 and 21.

- For an 8-byte internal floating-point number, the value must be between 22 and 53.
FLOAT EXTERNAL\((length, scale)\)

EXTERNAL indicates a string that contains a character representation of a float constant. For more information, see “Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS” on page 259.

ROUND

ROUND tells UNLOAD PLUS to round the value, if necessary, during conversion. If you do not specify ROUND, UNLOAD PLUS truncates any digits that it does not need for precision.

BINARY

This keyword defines a fixed-length binary field.

\((length)\)

This option specifies the number of single-byte characters.

TRIM

TRIM tells UNLOAD PLUS to remove as many trailing hexadecimal zeros from the string as needed to make the string length match the length you specify.

UNLOAD PLUS performs the TRIM function before attempting to assign the value to the field. If the string is still too long, a conversion error occurs. UNLOAD PLUS applies TRIM before TRUNCATE.

UNLOAD PLUS ignores the TRIM option if you specify FORMAT DSNTIAUL.

TRUNCATE

TRUNCATE tells UNLOAD PLUS to truncate a string if it is longer than the field (even after the TRIM option has truncated trailing hexadecimal zeros). UNLOAD PLUS performs TRUNCATE after TRIM.

VARBINARY

This keyword defines a binary string field that varies in length.

You can also specify this field type as BINARY VARYING. For more information, see “Additional variable field considerations in UNLOAD PLUS” on page 261.
The length indicates the number of single-byte characters. A 2-byte length field that contains the length of the character string in bytes precedes the VARBINARY string. This length does not include the two length bytes.

**TRIM**

TRIM tells UNLOAD PLUS to remove as many trailing hexadecimal zeros from the string as needed to make the string length match the length you specify. If the string is still too long, a conversion error occurs. UNLOAD PLUS applies TRIM before TRUNCATE.

UNLOAD PLUS ignores the TRIM option if you specify FORMAT DSNTIAUL.

**TRUNCATE**

TRUNCATE tells UNLOAD PLUS to truncate a string if it is longer than the field (even after the TRIM option has truncated trailing hexadecimal zeros). UNLOAD PLUS performs TRUNCATE after TRIM.

**DATE**

This keyword defines a date field.

For more information, see “Additional date, time, or timestamp field considerations in UNLOAD PLUS” on page 261.

**DATE**

DATE specifies a DB2 internal date format.

**DATE EXTERNAL**(length)

This option specifies a DB2 date string.

**DATE-format EXTERNAL**(length)

This option specifies a DB2 date string. In addition, it indicates the format of the external date in the column value, not the format of the field. The column must be CHARACTER, DECIMAL, or INTEGER. For information about available date formats, see “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247.
When using DIRECT NO, specifying DATE-\textit{format} EXTERNAL(\textit{length}) requires extra consideration due to the various ways in which the data is returned from DB2. Often, the same DATE-\textit{format} is not appropriate when using DIRECT YES versus DIRECT NO.

\textbf{CENTURY(\textit{ccyy},\textit{ccyy})}

CENTURY specifies the 100-year range that determines the century for DATE external formats that contain two-digit year values. The first four-digit year value must be less than the second four-digit year. You must specify both values, which must span 100 years.

Any two-digit year between the first \textit{yy} specification and 99 has the first \textit{cc} value prefixed to create a four-digit year. Any two-digit year that is between 00 and the second \textit{yy} specification has the second \textit{cc} value prefixed to create a four-digit year.

For example, if you specify CENTURY(1950,2049), UNLOAD PLUS places 19 in front of each two-digit year with a value 50 through 99, and places 20 in front of each two-digit year with a value 00 through 49. The date 99/12/31 becomes 1999/12/31 and 00/12/31 becomes 2000/12/31.

\textbf{TIMESTAMP}

This keyword specifies a timestamp field.

For more information, see “\textbf{Additional date, time, or timestamp field considerations in UNLOAD PLUS}” on page 261.

\textbf{TIMESTAMP}

TIMESTAMP specifies a DB2 internal timestamp format.

\textbf{TIMESTAMP EXTERNAL(\textit{length})}

This option specifies a DB2 timestamp string value.

\textbf{TIMESTAMP-\textit{format} EXTERNAL(\textit{length})}

This option specifies a DB2 timestamp string. In addition, it specifies the format of the timestamp in the column value, not the format of the field. The column must be CHARACTER only. For information about the available timestamp formats, see “\textbf{Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS}” on page 247.
When using DIRECT NO, specifying TIMESTAMP-format EXTERNAL\((length)\) requires extra consideration due to the various ways in which the data is returned from DB2. Often, the same TIMESTAMP-format is not appropriate when using DIRECT YES versus DIRECT NO.

**CENTURY\((ccyy,ccyy)\)**

CENTURY specifies the 100-year range that determines the century for TIMESTAMP external formats that contain two-digit year values. The first four-digit year value must be less than the second four-digit year. You must specify both values, which must span 100 years.

Any two-digit year between the first yy specification and 99 has the first cc value prefixed to create a four-digit year. Any two-digit year between 00 and the second yy specification has the second cc value prefixed to create a four-digit year.

For example, if you specify CENTURY(1950,2049), UNLOAD PLUS places 19 in front of each two-digit year with a value 50 through 99, and places 20 in front of each two-digit year with a value 00 through 49. The date 99/12/31 becomes 1999/12/31 and 00/12/31 becomes 2000/12/31.

**TIMESTAMP WITH TIME ZONE**

TIMESTAMP specifies a DB2 internal timestamp with time zone format.

**TIMESTAMP WITH TIME ZONE EXTERNAL\((length)\)**

This option specifies a DB2 timestamp with time zone string value.

**TIME**

This keyword specifies a time field.

For more information, see “Additional date, time, or timestamp field considerations in UNLOAD PLUS” on page 261.

**TIME**

TIME specifies a DB2 internal time format

**TIME EXTERNAL\((length)\)**

This option specifies a DB2 time string value.
**TIME-format EXTERNAL(length)**

This option specifies a DB2 time string as well as the format of the time in the column value, not the format of the field. The column must be CHARACTER, DECIMAL, or INTEGER. For information about available time formats, see “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247.

When using DIRECT NO, specifying `TIME-format EXTERNAL(length)` requires extra consideration due to the various ways in which the data is returned from DB2. Often, the same `TIME-format` is not appropriate when using DIRECT YES versus DIRECT NO.

**BLOB**

BLOB defines a field that contains binary large object (BLOB) data.

A four-byte length field precedes the string. This length does not include the four length bytes.

Specifying this option indicates that you are unloading the data directly to the unload data set rather than to a file that is referenced in the unload data set.

**CLOB**

CLOB defines a field that contains character large object (CLOB) data.

A four-byte length field precedes the string. This length does not include the four length bytes.

Specifying this option indicates that you are unloading the data directly to the unload data set rather than to a file that is referenced in the unload data set.

**DBCLOB**

DBCLOB defines a field that contains double-byte character large object (DBCLOB) data.

A four-byte length field precedes the string. This length does not include the four length bytes.

Specifying this option indicates that you are unloading the data directly to the unload data set rather than to a file that is referenced in the unload data set.
DECFLOAT

This option defines a decimal floating-point numeric field.

For more information, see “Additional numeric field considerations in UNLOAD PLUS” on page 260.

\( \text{precision} \)

The precision represents the total number of digits. You can specify 16 (an 8-byte number) or 34 (a 16-byte number). If you do not specify a precision, UNLOAD PLUS uses the following defaults:

- If the source column is defined as data type DECFLOAT, UNLOAD PLUS uses the precision attribute of the source column.
- If the source column is not defined as data type DECFLOAT, UNLOAD PLUS uses a precision of 34 bytes.

EXTERNAL

EXTERNAL indicates a string that contains a character representation of a decimal floating-point constant.

\( \text{length} \)

The length indicates the number of single-byte characters. Valid values are 1 through 42. If you do not specify a value, UNLOAD PLUS defaults to one of the following values:

- 23 if the source column is defined as data type DECFLOAT with a length of 8 bytes
- 42 in the following cases:
  - If the source column is defined as data type DECFLOAT with a length of 16 bytes
  - If the source column is not defined as data type DECFLOAT

When converting from DECFLOAT to DECFLOAT EXTERNAL, the length that you specify must be long enough to contain the significant digits of the input value without rounding. Otherwise, UNLOAD PLUS discards the row.

XML

This keyword defines a field that contains XML data.
Specifying this option indicates that you are unloading the data directly to the unload data set rather than to a file that is referenced in the unload data set.

**EXIT programName PARM (parmList) (length)**

Use the EXIT data type to request your own data conversion.

You must specify the program name, program type, and length. Optionally, you can specify a list of parameters to pass to the EXIT routine.

**programName**

This option specifies the name of your conversion exit routine. The library in which the program you name resides must be in your system LINKLIST, your JOBLIB, or STEPLIB.

The following table shows the return codes that UNLOAD PLUS expects in register 15.

**Table 43: User exit return codes**

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Conversion complete</td>
</tr>
<tr>
<td>4</td>
<td>Conversion error UNLOAD PLUS issues message BMC51632E and rejects this record but continues processing other records. If you specified the DISCARDS integer option, records that UNLOAD PLUS discarded because of a conversion error count toward the discards limit.</td>
</tr>
<tr>
<td>8</td>
<td>Parameter error UNLOAD PLUS issues message BMC51632E and does not process other records. The utility return code is set to 8.</td>
</tr>
</tbody>
</table>

If the exit returns an invalid return code, UNLOAD PLUS issues message BMC51632E and ends. For more information about exit routines, see “UNLOAD PLUS user exits” on page 531.

**PARM (constant1,constant2,...)**

You can specify an optional parameter list that contains integer, decimal, string, or hexadecimal constants. UNLOAD PLUS does not support floating-point and graphic constants for parameter values.

The parameter list that UNLOAD PLUS passes to a conversion exit routine is identical to the parameter list that UNLOAD PLUS passes to a DB2 FIELDPROC exit routine; that is, the FPBFCODE in the FPIB is set to FBBFDEC (= 4), the CVD is the
value for the output record, the FVD is the column value, and the FPPVL is the PARM.

Coding requirements for the UNLOAD PLUS conversion exit are the same as those for a DB2 FIELDPROC. For more help with exit requirements, see the IBM DB2 documentation.

**Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS**

The tables in this topic list input date, time, and timestamp formats that UNLOAD PLUS supports in addition to the internal DB2 date, time, and timestamp formats.

These formats support the conversion of CHAR, INTEGER and DECIMAL data types to DATE-format, TIMESTAMP-format, and TIME-format EXTERNAL formats.

When using DIRECT NO, DATE-format, TIMESTAMP-format, or TIME-format formats require extra consideration due to the various ways in which the data is returned from DB2. Often, the same DATE-format, TIMESTAMP-format, or TIME-format formats is not appropriate when using DIRECT YES versus DIRECT NO.

<table>
<thead>
<tr>
<th>Table 44: Date formats</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format number</strong></td>
</tr>
<tr>
<td><strong>Char/Int/Dec</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>1E</td>
</tr>
<tr>
<td>2E</td>
</tr>
<tr>
<td>3E</td>
</tr>
<tr>
<td>4E</td>
</tr>
<tr>
<td>5E</td>
</tr>
</tbody>
</table>

*a ‘/’ can be any character.*
Table 45: Timestamp formats

<table>
<thead>
<tr>
<th>Format number</th>
<th>Nondelimited format&lt;sup&gt;a&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>YYMMDDHHMMSS</td>
</tr>
<tr>
<td>2</td>
<td>YYMMDDHHMMSSNNNNNN&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>3</td>
<td>Store clock 8-byte binary</td>
</tr>
<tr>
<td>1E</td>
<td>YYYYMMDDHHMMSS</td>
</tr>
<tr>
<td>2E</td>
<td>YYYYMMDDHHMMSSNNNNNN&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

<sup>a</sup> Formats are CHAR only except 3.

<sup>b</sup> The number of microseconds (N) can be from 0 through 12.

Table 46: Time formats

<table>
<thead>
<tr>
<th>Format number</th>
<th>Char/Int/Dec</th>
<th>Nondelimited format</th>
<th>Format number</th>
<th>Char</th>
<th>Delimited format&lt;sup&gt;a&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HHMMSS</td>
<td>D1</td>
<td>HH.MM.SS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>HHMM</td>
<td>D2</td>
<td>HH.MM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D3</td>
<td>HH.MM xM&lt;sup&gt;b&lt;/sup&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup> ‘.’ can be any character.

<sup>b</sup> xM is either AM or PM.

### Supported data type conversions in UNLOAD PLUS

The following sections describe the allowable data conversions and default output lengths.

A blank cell in a table indicates that the data type conversion is not allowed.

**Numeric output**

The following table lists the conversions that UNLOAD PLUS supports to numeric output. Table 48 on page 250 lists the default output lengths for these conversions.
Table 47: Allowable data type conversions for numeric output fields

<table>
<thead>
<tr>
<th>Input column</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>SMALLINT, INTEGER, or BIGINT</th>
<th>DECIMAL</th>
<th>DECIMAL ZONED</th>
<th>FLOAT</th>
<th>DECFLOAT</th>
<th>DECIMAL, FLOAT, or DECFLOAT EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X</td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td>INTEGER</td>
<td>X</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X</td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td>BIGINT</td>
<td>X</td>
<td>X</td>
<td>D</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X</td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>D</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X</td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td>FLOAT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>D</td>
<td></td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>D</td>
<td></td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X</td>
<td></td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X</td>
<td></td>
<td>X&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The column values are:

**X** = Allowable conversion

**D** = Allowable conversion and indicates the default output data type, which is the data type of the column

**N** = Allowable conversion but not supported with DIRECT NO
For integer and decimal columns, you can assign positive or negative overpunch values to decimal-zoned numeric output.

For integer and decimal columns, conversion to FLOAT is inexact, and the value in the floating-point output might not be identical to the value in the input column.

When making an allowable conversion from a numeric column to a CHAR, VARCHAR, DECIMAL EXTERNAL, FLOAT EXTERNAL, or DECFLOAT EXTERNAL field, you must allow space for punctuation. Punctuation includes the minus sign, decimal point, exponent marker, exponent sign, and exponent value.

When converting from DECFLOAT to DECFLOAT EXTERNAL, if the length that you specify is not long enough to contain the significant digits of the input value without rounding, UNLOAD PLUS discards the row.

The following considerations apply to the default output length information for these fields:

- With FORMAT EXTERNAL, the length defaults to the following value unless EXTERNAL is specified on the field:
  - SMALLINT defaults to 6.
  - INTEGER defaults to 11.
  - BIGINT defaults to 20.
  - DECIMAL defaults to 17.

- For DECFLOAT columns converting to DECFLOAT EXTERNAL output fields, the default length depends on the source column:
  - If the source column is defined with a length of 8 bytes, the default is 23.
  - If the source column is defined with a length of 16 bytes, the default is 42.

### Table 48: Default lengths for numeric output fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>SMALLINT, INTEGER, or BIGINT EXTERNAL</th>
<th>DECIMAL</th>
<th>DECIMAL ZONED</th>
<th>FLOAT</th>
<th>DECFLOAT</th>
<th>DECFLOAT or FLOAT EXTERNAL</th>
<th>DECFLOAT</th>
<th>DECFLOAT EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>42</td>
</tr>
<tr>
<td>INTEGER</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>42</td>
</tr>
<tr>
<td>BIGINT</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>42</td>
</tr>
<tr>
<td>Output field</td>
<td>SMALLINT</td>
<td>INTEGER</td>
<td>BIGINT</td>
<td>SMALLINT, INTEGER, or BIGINT</td>
<td>DECIMAL</td>
<td>DECIMAL ZONED</td>
<td>FLOAT</td>
<td>DECIMAL or FLOAT EXTERNAL</td>
<td>DECFLOAT</td>
<td>DECFLOAT EXTERNAL</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>---------</td>
<td>--------</td>
<td>-------------------------------</td>
<td>---------</td>
<td>----------------</td>
<td>-------</td>
<td>-------------------------</td>
<td>----------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>C</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>C</td>
<td>C</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Graphic, graphic, and binary output**

The following table lists the conversions that UNLOAD PLUS supports to character, graphic, and binary output. Table 50 on page 253 lists the default output lengths for these conversions.

- **X** = Explicit length is required.
- **C** = The default length is defined by the column. For DECIMAL and DECFLOAT, the precision is defined by the column.
- **N** = The length is defined by the input value.
- **L** = If your DSNHDEC format is LOCAL, the length is the local DATE and TIME length. If your DSNHDEC format is ISO, USA, EUR, or JIS, the default length for DATE is 10 and the default length for TIME is 8.
- **T** = See “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247 for length information.
When you specify FORMAT CSV or FORMAT XML, UNLOAD PLUS does not support conversion to VARCHAR, VARGRAPHIC, BINARY, VARBINARY, or ROWID data types.

### Table 49: Allowable data type conversions for character, graphic, and binary output fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>CHAR or VARCHAR</th>
<th>CHAR or VARCHAR</th>
<th>CHAR or VARCHAR</th>
<th>GRAPHIC</th>
<th>GRAPHIC EXTERNAL</th>
<th>VARGRAPHIC</th>
<th>BINARY</th>
<th>VARBINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td>BLOBF</td>
<td>BLOBF</td>
<td>DBCLOBF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>X a</td>
<td>X a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>X a</td>
<td>X a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>X a</td>
<td>X a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>X a</td>
<td>X a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>X a</td>
<td>X a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFL0AT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>D</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>X</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>X</td>
<td>X</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>XML</td>
<td>X b</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The column values are:

- X = Allowable conversion
- D = Allowable conversion and indicates the default output data type, which is the data type of the column
- N = Allowable conversion but not supported with DIRECT NO
When making an allowable conversion from a numeric column to a CHAR, VARCHAR, DECIMAL EXTERNAL, FLOAT EXTERNAL, or DECFLOAT EXTERNAL field, you must allow space for punctuation. Punctuation includes the minus sign, decimal point, exponent marker, exponent sign, and exponent value.

When unloading XML data to a CHAR BLOBF or VARCHAR BLOBF field, you must specify BINARYXML.

### Table 50: Default lengths for character, graphic, and binary output fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>GRAPHIC</th>
<th>GRAPHIC EXTERNAL</th>
<th>VARGRAPHIC</th>
<th>BINARY</th>
<th>VARBINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>X</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>X</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>X</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>X</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>X</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>C&lt;sup&gt;a&lt;/sup&gt;</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>C</td>
<td>N&lt;sup&gt;a&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td>C</td>
<td>C</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td>C</td>
<td>C</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td>N</td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td>N</td>
</tr>
<tr>
<td>DATE</td>
<td>10</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>8</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>26&lt;sup&gt;b&lt;/sup&gt;</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>26b</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Numeric values in a column indicate the required output length.

X = Explicit length is required.

C = The default length is defined by the column. For DECIMAL and DECIMAL, the precision is defined by the column.

N = The length is defined by the input value.

L = If your DSNHDEC format is LOCAL, the length is the local DATE and TIME length. If your DSNHDEC format is ISO, USA, EUR, or JIS, the default length for DATE is 10 and the default length for TIME is 8.

T = See “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247 for length information.

---

**Date, time, and timestamp output**

Table 51 on page 254 lists the conversions that UNLOAD PLUS supports to date, time, and timestamp output. Table 52 on page 255 lists the default output lengths for these conversions.

The following considerations apply to these data type conversions:

- The default data format depends on the FORMAT option that is in effect. For example, when FORMAT STANDARD (the default) is in effect, the default data format for date/time columns is external format.

- For output date/time EXTERNAL data types with a format number, the format number describes the representation of the column date/time value, not the format of the output field.

- When using DIRECT NO, specifying formats for EXTERNAL data types (for example, DATE- format EXTERNAL) might require extra consideration due to the various ways in which the data is returned from DB2. Often, you should not specify the same format when using DIRECT NO versus DIRECT YES.

**Table 51: Allowable data type conversions for date, time, and timestamp output fields**

<table>
<thead>
<tr>
<th>Output field</th>
<th>DATE</th>
<th>DATE- format EXTERNAL</th>
<th>TIME</th>
<th>TIME- format EXTERNAL</th>
<th>TIMESTAMP or TIMESTAMP WITH TIME ZONE</th>
<th>TIMESTAMP EXTERNAL</th>
<th>TIMESTAMP -format EXTERNAL</th>
<th>TIMESTAMP WITH TIME ZONE EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 52: Default lengths for date, time, and timestamp output fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>DATE</th>
<th>DATE EXTERNAL</th>
<th>DATE-EXTERNAL</th>
<th>TIME</th>
<th>TIME EXTERNAL</th>
<th>TIME-EXTERNAL</th>
<th>TIMESTAMP</th>
<th>TIMESTAMP WITH TIME ZONE</th>
<th>TIMESTAMP</th>
<th>TIMESTAMP -EXTERNAL</th>
<th>TIMESTAMP WITH TIME ZONE EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMALFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>X</td>
<td>N</td>
<td>X</td>
<td></td>
<td>N</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>X</td>
<td>N</td>
<td>X</td>
<td></td>
<td>N</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The column values are:
- X = Allowable conversion
- D = Allowable conversion and indicates the default output data type, which is the data type of the column
- N = Allowable conversion but not supported with DIRECT NO
### UNLOAD PLUS options

<table>
<thead>
<tr>
<th>Output field</th>
<th>DATE</th>
<th>DATE - format</th>
<th>TIME</th>
<th>TIME - format</th>
<th>TIMESTAMP or TIMESTAMP WITH TIME ZONE</th>
<th>TIMESTAMP</th>
<th>TIMESTAMP - format</th>
<th>TIMESTAMP WITH TIME ZONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td>DATE EXTERNAL</td>
<td>DATE - format EXTERNAL</td>
<td>TIME EXTERNAL</td>
<td>TIME - format EXTERNAL</td>
<td>TIMESTAMP or TIMESTAMP WITH TIME ZONE</td>
<td>TIMESTAMP EXTERNAL</td>
<td>TIMESTAMP - format EXTERNAL</td>
<td>TIMESTAMP WITH TIME ZONE EXTERNAL</td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td>T</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td>T</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>4</td>
<td>T</td>
<td>3</td>
<td>T</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>4</td>
<td></td>
<td>3</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>4</td>
<td>L&lt;sup&gt;a&lt;/sup&gt;</td>
<td>3</td>
<td>L&lt;sup&gt;a&lt;/sup&gt;</td>
<td>10&lt;sup&gt;b&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td>3</td>
<td>L&lt;sup&gt;a&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>4</td>
<td>L&lt;sup&gt;a&lt;/sup&gt;</td>
<td>3</td>
<td>L&lt;sup&gt;a&lt;/sup&gt;</td>
<td>10&lt;sup&gt;b&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>4</td>
<td>3</td>
<td>10&lt;sup&gt;b&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Numeric values in a column indicate the required output length.

X = Explicit length is required.

C = The default length is defined by the column. For DECIMAL and DECFLOAT, the precision is defined by the column.

N = The length is defined by the input value.

L = If your DSNHDECP format is LOCAL, the length is the local DATE and TIME length. If your DSNHDECP format is ISO, USA, EUR, or JIS, the default length for DATE is 10 and the default length for TIME is 8.

T = See “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247 for length information.

1 For EXTERNAL date/time fields, if you also specified DATEFMT, TIMEFMT, or TSFMT, the length defaults to the format element string length.

2 For timestamp columns that are defined with precision, the default length of the output field is defined by the input value.
# LOB and XML output

The following table lists the conversions that UNLOAD PLUS supports to LOB and XML output. Table 54 on page 258 lists the default output lengths for these conversions.

**Table 53: Allowable data type conversions for LOB and XML output fields**

<table>
<thead>
<tr>
<th>Output field</th>
<th>BLOB</th>
<th>CLOB</th>
<th>DBCLOB</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input column</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

The column values are:

- X = Allowable conversion
- D = Allowable conversion and indicates the default output data type, which is the data type of the column
- N = Allowable conversion but not supported with DIRECT NO
### Table 54: Default lengths for LOB and XML output fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>BLOB</th>
<th>CLOB</th>
<th>DBCLOB</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITH TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Numeric values in a column indicate the required output length.

X = Explicit length is required.

C = The default length is defined by the column. For DECIMAL and DECFLOAT, the precision is defined by the column.

N = The length is defined by the input value.

L = If your DSNHDECPC format is LOCAL, the length is the local DATE and TIME length. If your DSNHDECPC format is ISO, USA, EUR, or JIS, the default length for DATE is 10 and the default length for TIME is 8.

T = See “Supported DATE, TIME, and TIMESTAMP formats in UNLOAD PLUS” on page 247 for length information.
Additional numeric EXTERNAL or CHARACTER field considerations in UNLOAD PLUS

This topic describes additional considerations that apply to numeric EXTERNAL fields and CHARACTER fields.

A numeric EXTERNAL or CHARACTER field for a numeric column produces an EBCDIC character representation of the numeric value. The major difference between the two is that numeric EXTERNAL is a numeric type, and CHARACTER is a character type. This difference is important when using IF condition VALUE(), because the data type of the value must match the data type of the field. For example, INTEGER EXTERNAL(9) requires that you use a numeric value like VALUE(0), while CHARACTER(9) would require you to use a character value such as VALUE('    0') to get the same results.

The following table illustrates what you can expect when you use numeric EXTERNAL:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT EXTERNAL</td>
<td>Character representation of the integer value with leading zeros removed, unless you also specify FILL=YES or specify the FILL installation option</td>
</tr>
<tr>
<td>INTEGER EXTERNAL</td>
<td>Character representation of the integer value with leading zeros removed, unless you also specify FILL=YES or specify the FILL installation option</td>
</tr>
<tr>
<td>BIGINT EXTERNAL</td>
<td>Character representation of the integer value with leading zeros removed, unless you also specify FILL=YES or specify the FILL installation option</td>
</tr>
<tr>
<td></td>
<td>Leading '-' sign if negative</td>
</tr>
<tr>
<td></td>
<td>No decimal point, and right-justified in an 11-byte maximum field (for SMALLINT and INTEGER) or 20-byte maximum field (for BIGINT)</td>
</tr>
<tr>
<td></td>
<td>UNLOAD PLUS uses blanks to pad the value if the field is longer than 11 (SMALLINT and INTEGER) or 20 (BIGINT).</td>
</tr>
<tr>
<td>DECIMAL EXTERNAL</td>
<td>Character representation of the decimal value with leading zeros removed, unless you also specify FILL=YES or specify the FILL installation option</td>
</tr>
<tr>
<td></td>
<td>Leading '-' sign if negative</td>
</tr>
<tr>
<td></td>
<td>Decimal point as indicated by the scale and right justified in a 33-byte maximum field</td>
</tr>
<tr>
<td></td>
<td>UNLOAD PLUS uses blanks to pad the value if the field is longer than 33.</td>
</tr>
<tr>
<td>FLOAT EXTERNAL</td>
<td>Character representation of the floating-point value in the format of +-.dddddddddE+-ee</td>
</tr>
<tr>
<td></td>
<td>The maximum field width is 15 for single precision and 24 for double precision.</td>
</tr>
</tbody>
</table>
### Additional numeric field considerations in UNLOAD PLUS

This topic describes additional considerations for numeric fields.

#### Scale

With the exception of DECFLOAT fields, you can specify scale on any numeric (internal, EXTERNAL, or other format) fields with or without specifying a length. Specifying a scale multiplies or divides the number by a power of 10 or, for DECIMAL EXTERNAL, indicates where you want the decimal point. For example, `INTEGER(,2)` multiplies the number by 100, `INTEGER(-3)` divides the number by 1000, and `DECIMAL EXTERNAL(5,2)` gives you two digits to the right of the decimal point, no matter what the scale of the source value is.

If you do not specify scale, 0 is the default unless the column is DECIMAL and the field is DECIMAL(PACKED) (explicit or defaulted), or DECIMAL ZONED. In these cases, the scale defaults to the scale of the column.

#### Restrictions

The following restrictions apply to the scale specification:

- You cannot specify scale on a DECFLOAT field.
- With the exception of a decimal field, UNLOAD PLUS does not support the scale specification on an output field if you are converting from a DECFLOAT column.

#### Rounding

For all numeric fields except DECFLOAT, you can specify ROUND to tell UNLOAD PLUS to round values during conversion to make them fit into the target field. Otherwise, UNLOAD PLUS truncates any additional digits that it does not need for precision. Truncation is not considered a conversion error.

For DECFLOAT columns, the value of the DECFLOAT_ROUNDMODE option determines the rounding method.
Additional date, time, or timestamp field considerations in UNLOAD PLUS

This topic describes additional considerations for date, time, and timestamp fields.

A date, time, or timestamp EXTERNAL field produces a character representation of the date, time, or timestamp value, producing an EUR, ISO, JIS, USA, or LOCAL format depending on the default date/time formats and lengths found in DB2 module DSNHDECP.

The default data format depends on the FORMAT option that is in effect. For example, when FORMAT STANDARD (the default) is in effect, EXTERNAL is the default for the field data type if you do not specify the field data type for a date, time, or timestamp column.

Additional variable field considerations in UNLOAD PLUS

This topic describes additional considerations for variable-length fields.

For VARCHAR, VARGRAPHIC, and VARBINARY data types, a two-byte length precedes the data. For VARCHAR and VARBINARY, the length is the number of single-byte characters. For VARGRAPHIC, the length is the number of double-byte characters. The length does not include the two length bytes.

Additional ROWID field considerations in UNLOAD PLUS

This topic describes additional considerations for ROWID fields.

For this data type, UNLOAD PLUS supports the data only in internal format.

Dynamic SQL processing cannot identify ROWID columns that are defined as GENERATED ALWAYS. When you specify DIRECT NO, UNLOAD PLUS generates the LOAD control statements as if the ROWID column were defined as GENERATED BY DEFAULT.

Data translation in UNLOAD PLUS

Before writing the output record to the output data set, UNLOAD PLUS translates the data to the specified output encoding scheme. How UNLOAD PLUS performs this translation depends on the value of your DIRECT option.
**DIRECT YES processing**

For DIRECT YES processing, UNLOAD PLUS translates data to the specified output encoding scheme using the ASCII, EBCDIC, or UNICODE option and the CCSID option. During translation processing, UNLOAD PLUS searches the SYSIBM.SYSSTRINGS catalog table for a row that identifies how to translate the data.

Translation processing is handled in the follow manner:

1. Based on the CCSID that you specify or that UNLOAD PLUS retrieves from DSNHDECP, UNLOAD PLUS searches the SYSIBM.SYSSTRINGS catalog table for a row that identifies how to translate the data. (To review the values that are assigned to the CCSID, see the UNLOAD PLUS parameter listing under message number BMC50471I.)

2. If a row exists, UNLOAD PLUS searches the row for a value, first in the TRANSPROC field and then in the TRANSTAB field. UNLOAD PLUS terminates when either of the following conditions exists:
   - The TRANSPROC field contains DSNXVJPC or DSNXVTWC. UNLOAD PLUS does not support translations that require a translation procedure.
   - The TRANSTAB field is empty, and the TRANSPROC field contains a value.

In all other cases, UNLOAD PLUS performs the data translation providing that the combination is supported. For a list of supported translations, see Table 56 on page 263.

3. If a row does not exist in SYSIBM.SYSSTRINGS, UNLOAD PLUS uses the z/OS Unicode Conversion Services to translate the data.

**DIRECT NO processing**

For DIRECT NO processing, DB2 performs the data translation. Using DIRECT NO, you can translate your encoded data with full SELECT capabilities.

**Supported character conversions**

The following table identifies the character conversions that UNLOAD PLUS supports for translation processing:
Table 56: Supported translations

<table>
<thead>
<tr>
<th>Source</th>
<th>ASCII SBCS</th>
<th>ASCII MIXED</th>
<th>ASCII DBCS</th>
<th>EBCDIC SBCS</th>
<th>EBCDIC MIXED</th>
<th>EBCDIC DBCS</th>
<th>UNICODE SBCS</th>
<th>UNICODE MIXED</th>
<th>UNICODE DBCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII SBCS</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCII MIXED</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCII DBCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBCDIC SBCS</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBCDIC MIXED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBCDIC DBCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNICODE SBCS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNICODE MIXED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNICODE DBCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

This table indicates the supported translations with the following identifiers:

- X: allowable translation for either DIRECT YES or DIRECT NO
- Y: allowable translation for DIRECT YES only

Expansion and contraction

The translation process might cause the length of the resultant data to change based on the CCSID type that you specified. To determine the resultant length of the data based on each CCSID conversion, see the IBM DB2 SQL Reference.

Restrictions

UNLOAD PLUS has the following restrictions on data translation:

- Because UNLOAD PLUS does not convert data when you specify FORMAT BMCLOAD, the utility ignores special options for date and time, ASCII, EBCDIC, and UNICODE. The CCSID of the target table must be identical to the CCSID of the source table, however.
UNLOAD PLUS does not support the UNICODE option with the FORMAT XML option.

UNLOAD PLUS does not translate the following data:

— Binary data

— Columns that are defined as FOR BIT DATA, including RID values when you specify SELECT CURRENT RID

**Mixed data**

Translation is not supported from Unicode MBCS (UTF-8) to EBCDIC or ASCII MIXED on a DB2 subsystem that is defined as MIXED=NO, because the target CCSID value is undefined. However, UNLOAD PLUS and other SQL applications such as SPUFI and DSNTIAUL, extract MIXED columns as SBCS data. During the translation process, character substitutions might occur between the source CCSID and target CCSID; a loss of data integrity can result, due to SUBBYTE characters in the translated data. For more details, see “SUBBYTE and ERRORBYTE” on page 265.

**WARNING**

If you specify the NOSUBS option with DIRECT NO, UNLOAD PLUS ignores the NOSUBS option for translations that DB2 handles. In cases in which UNLOAD PLUS handles row-level data translation (such as INTO field specifications or reformatting of DATE, TIME, or TIMESTAMP columns), specifying NOSUBS might result in UNLOAD PLUS discarding rows or terminating.

**Command constants**

For comparisons between command constants and row data, UNLOAD PLUS must translate certain command constants from EBCDIC (using the DB2 installation default EBCDIC SBCS CCSID) to the encoding scheme of the table. UNLOAD PLUS translates the following character constants for comparison:

- Predicate block constants
- LIKE constants
- IN constants

For additional information about these constants, see “predicate” on page 215.

UNLOAD PLUS translates output data from EBCDIC (using the DB2 installation default EBCDIC SBCS CCSID) to the output encoding scheme. UNLOAD PLUS translates the following character constants for output:
AUTOTAG values

SELECT constants

FORMAT CSV constants

FORMAT XML tags

IF VALUE constants

NULLCHAR

For additional information about these constants, see the description for the individual option in this chapter.

**SUBBYTE and ERRORBYTE**

UNLOAD PLUS uses the SUBBYTE and ERRORBYTE fields in a similar way. If you specify NOSUBS to prevent character substitutions when a string is converted from one CCSID to another, and during conversion a substitution character is placed in the output string because a character in the source CCSID does not exist in the target, the attempted character substitution causes a conversion error.

In UNLOAD PLUS, the DISCARDS option defines the limit on the number of records that UNLOAD PLUS ignores during the character conversion. UNLOAD PLUS ends abnormally if it reaches the discard limit, and does not write the discarded records to any data set.

**Order of data type conversion and data translation**

UNLOAD PLUS supports data type conversions from one encoding scheme to another. The following tables detail when translation occurs with respect to the data type conversion that UNLOAD PLUS is performing.

### Table 57: Conversion and translation processing for numeric output

<table>
<thead>
<tr>
<th>Output field</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL</th>
<th>DECIMAL ZONED</th>
<th>FLOT</th>
<th>All numeric EXTERNAL output except DECFLOAT</th>
<th>DECFLOAT</th>
<th>DECFLOAT EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>cpt 2</td>
<td>x</td>
</tr>
<tr>
<td>INTEGER</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>ccp 2</td>
<td>x</td>
</tr>
<tr>
<td>BIGINT</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>ccp 2</td>
<td>x</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>ccp 2</td>
<td>x</td>
</tr>
<tr>
<td>FLOAT</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>ccp 2</td>
<td>x</td>
</tr>
</tbody>
</table>
### Table 58: Conversion and translation processing for character, graphic, and binary output

<table>
<thead>
<tr>
<th>Output field</th>
<th>CHARACTER</th>
<th>VARCHAR</th>
<th>GRAPHIC, GRAPHIC EXTERNAL</th>
<th>VARGRAPHIC</th>
<th>BINARY</th>
<th>VARBINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output field</td>
<td>CHARACTER</td>
<td>VARCHAR</td>
<td>GRAPHIC, GRAPHIC EXTERNAL</td>
<td>VARGRAPHIC</td>
<td>BINARY</td>
<td>VARBINARY</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>---------</td>
<td>---------------------------</td>
<td>------------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>cpt 2</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>cpt 2</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>cpt 2</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>cpt 3</td>
<td>cpt 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>cpt 3</td>
<td>cpt 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>cpt 2</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>cpt 2</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>cpt 2</td>
<td>cpt 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>cpt2</td>
<td>cpt2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The character combinations in this table indicate the order in which data type conversions (c), data translations (t), and character padding (p) occur. The numbers (1,2,3) represent the type of data translation that occurs when converting from one data type to another.

- x = Support conversion, no translation required
- c = Convert
- t = Translate
- p = Pad with blank
- 1 = Translate input CCSID to EBCDIC SBCS (system default)
- 2 = Translate EBCDIC SBCS (system default) to output CCSID
- 3 = Translate input CCSID to output SBCS
Table 59: Conversion and translation processing for date, time, and timestamp output

<table>
<thead>
<tr>
<th>Output field</th>
<th>DATE</th>
<th>DATE - format</th>
<th>TIME</th>
<th>TIME - format</th>
<th>TIMESTAMP, TIMESTAMP WITH TIME ZONE</th>
<th>TIMESTAMP</th>
<th>TIMESTAMP - format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td>DATE EXTERNAL</td>
<td>DATE EXTERNAL</td>
<td>TIME EXTERNAL</td>
<td>TIME EXTERNAL</td>
<td>TIMESTAMP, TIMESTAMP WITH TIME ZONE</td>
<td>TIMESTAMP EXTERNAL</td>
<td>TIMESTAMP - format EXTERNAL</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>x</td>
<td>a</td>
<td>x</td>
<td>a</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>x a</td>
<td></td>
<td>x a</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>tc1</td>
<td>tctp1,2 a</td>
<td>tc1</td>
<td>tctp1,2 a</td>
<td>tc1</td>
<td>tctp1,2 a</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>tc1</td>
<td></td>
<td>tc1</td>
<td></td>
<td>tc1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The character combinations in this table indicate the order in which data type conversions (c), data translations (t), and character padding (p) occur. The numbers (1,2,3) represent the type of data translation that occurs when converting from one data type to another.

x = Support conversion, no translation required

c = Convert

t = Translate

p = Pad with blank

1 = Translate input CCSID to EBCDIC SBCS (system default)

2 = Translate EBCDIC SBCS (system default) to output CCSID

3 = Translate input CCSID to output SBCS
When using DIRECT NO, specifying DATE-format, TIMESTAMP-format, or TIME-format formats might require extra consideration due to the various ways in which the data is returned from DB2. Often, you should not specify the same format when using DIRECT NO versus DIRECT YES.

Table 60: Conversion and translation processing for LOB and XML output

<table>
<thead>
<tr>
<th>Output field</th>
<th>BLOB</th>
<th>CLOB</th>
<th>DBCLOB</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input column</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
<td>tc3</td>
</tr>
</tbody>
</table>
The character combinations in this table indicate the order in which data type conversions (c), data translations (t), and character padding (p) occur. The numbers (1,2,3) represent the type of data translation that occurs when converting from one data type to another.

- x = Support conversion, no translation required
- c = Convert
- t = Translate
- p = Pad with blank

1 = Translate input CCSID to EBCDIC SBCS (system default)
2 = Translate EBCDIC SBCS (system default) to output CCSID
3 = Translate input CCSID to output SBCS
Building and executing UNLOAD PLUS jobs

This chapter describes the elements of an UNLOAD PLUS job, including the data sets that UNLOAD PLUS uses.

Building the UNLOAD PLUS job

Building a job for the UNLOAD PLUS for DB2 product involves creating a set of JCL, described in this section.

The JCL must include the following elements:

- A JOB statement
- An EXEC statement with the appropriate utility parameters
- STEPLIB or JOBLIB DD statements
- DD statements for the appropriate number and size of data sets
- UNLOAD PLUS control statements using the appropriate command syntax

For examples of UNLOAD PLUS jobs, see “Examples of UNLOAD PLUS jobs” on page 311.

JOB statement

Include a JOB statement that conforms to your site’s standards.

You can include the REGION parameter on either your JOB statement or your EXEC statement. For recommendations, see “REGION parameter” on page 272.
UNLOAD PLUS EXEC statement

The UNLOAD PLUS EXEC statement specifies the module to be executed for the UNLOAD PLUS utility.

The UNLOAD PLUS module name is ADUUMAIN. The EXEC statement also specifies the utility parameters, which are described in “Utility parameters on the UNLOAD PLUS EXEC statement” on page 272.

You can use the REGION parameter to specify the region size on either your EXEC statement or your JOB statement. See “REGION parameter” on page 272 for recommendations.

REGION parameter

Include the REGION parameter on either your JOB statement or your EXEC statement to specify the region size (the amount of virtual storage used by the utility).

For the best performance, BMC recommends that you specify REGION=0M to allocate the optimal amount available virtual storage to the utility job. If your data center does not permit you to specify REGION=0M, specify the amount that allows the most virtual storage both above and below the 16-megabyte line. Using a region size that is less than optimal risks the following potential issues:

- Running less efficiently, which could result in additional CPU and elapsed time
- Encountering memory failures or jobs that fail when new versions implement changes that require additional memory

Note

If you specify a value for REGION other than 0M, ensure that you have an appropriate value set for the MEMLIMIT parameter, either as your site’s default SMF option or on your JOB statement or EXEC statement.

- Specify NOLIMIT to allow unlimited above-the-bar memory.
- If you are unable to specify NOLIMIT, specify at least 4 GB; if you are operating on LOB or XML data, specify at least 32 GB.

Utility parameters on the UNLOAD PLUS EXEC statement

The UNLOAD PLUS EXEC statement includes the following utility parameters:
- DB2 subsystem ID or group attachment name
- Utility ID
- Restart parameter
- TSO user ID for notification of progress made on the unload job
- Message level
- Installation options module

The following illustration shows the format of the EXEC statement:

```
//stepName EXEC PGM=ADUUMAIN,
//  PARM='ssid,utilityID,restartParameter,userID,MSGLEVEL(n),optionsModule'
```

The UNLOAD PLUS utility parameters are positional. If you do not specify a value for a parameter (thus allowing the default value), you must substitute a comma for that parameter if additional parameters follow. The comma indicates that a parameter was omitted.

**DB2 subsystem identifier (SSID)**

This parameter specifies the four-character DB2 subsystem ID where the object resides.

If you do not specify the SSID, the utility uses the DB2 installation default from the DSNHDECP module. The utility depends on the application defaults module being named DSNHDECP. If you do not specify an SSID and the utility cannot find a module named DSNHDECP in your LINKLIST or STEPLIB, the utility terminates.

Note the following considerations when running the utility in a data sharing environment:

- The utility supports the DB2 group attachment name capability. When you supply a group attachment name as the SSID, the utility uses it to connect all plans. The utility then determines the actual DB2 SSID from within that group to use for the current reorganization.

- When you are restarting in a data sharing environment, the utility can use either the same member that was chosen in the original reorganization or any other member in the specified group.

- When both of the following conditions exist, specify a member SSID for your reorganization job instead of a group attachment name. Ensure that the DB2 version of the SSID that you specify corresponds to the version of the DB2 load library that you specify in your STEPLIB.
You are using table space compression.

The subsystems in your data sharing group are not all at the same DB2 version level.

**Utility identifier (utility ID)**

This parameter specifies the 1- to 16-character utility ID that gives a unique name to a utility job. If you omit this parameter, the utility uses the default, `userID.jobName`. Each BMC utility job should have a unique ID.

Utility IDs that include special characters might cause the utility to generate invalid data set names when using dynamic allocation. For more information, see the DSNPAT or DSNNAME option description.

**Restart parameter**

The restart parameter can have one of the following values:

- Blank or not specified
- NEW
- TERM
- MAINT

*Note*

Although UNLOAD PLUS accepts the restart values RESTART, RESTART(PHASE), NEW/RESTART, or NEW/RESTART(PHASE) if you specify one of them, the utility executes as if you specified NEW.

**Blank or not specified**

By not specifying a restart parameter, UNLOAD PLUS initiates a new BMC utility job. The utility ID that you specify cannot currently exist in the BMCUTIL table.

**NEW**

Specifying this value initiates a new BMC utility job or replaces an existing utility ID. Specifying this value allows you to start a utility without having to end the utility ID separately.

If you specify NEW and the utility ID has a status of X (executing), UNLOAD PLUS issues error message BMC50012E and ends with return code 8. For
message explanations, access the BMC Documentation Center from the BMC Support Central site http://www.bmc.com/support.

TERM

Specifying this value terminates an existing utility, removes the utility ID from the BMCUUTIL table, and removes the corresponding rows from the BMCSYNC table. After removing sync point and restart information, UNLOAD PLUS terminates without unloading. UNLOAD PLUS terminates with return code 0 regardless of whether the utility ID exists.

When specifying TERM, you need only minimal JCL. Your JCL must include at least a SYSPRINT DD statement and STEPLIB to the UNLOAD PLUS load library.

MAINT

Specifying this value forces MSGLEVEL(1) and causes UNLOAD PLUS to print the following information:

- An options module report that lists the values in the installation options module that you are using
- The values in the DSNHDECP module that UNLOAD PLUS uses
- A summary report of all of the product fixes that you have applied

When you specify the MAINT parameter, the job ends without affecting any utility that is running.

For this parameter, you need only minimal JCL. Your JCL must include at least a SYSPRINT DD statement and STEPLIB to the UNLOAD PLUS and DB2 load libraries.

User identifier (user ID)

This parameter specifies the TSO user ID that the utility notifies after it completes each phase and after it completes the command execution.

Message level (MSGLEVEL)

This parameter controls which messages UNLOAD PLUS returns in the SYSPRINT and SYSPRIN2 data sets. MSGLEVEL(0) returns minimal messages. MSGLEVEL(1) returns additional messages to help you diagnose problems and fine-tune performance.

You can use the MSGLEVEL installation option to change the default value of this parameter.
**Installation options module**

This parameter allows you to identify which installation options module to use. If you include this parameter, you must specify the full name of the options module. If you omit this parameter, UNLOAD PLUS uses the default installation options module, ADU$OPTS.

For more information about installation options, see “UNLOAD PLUS installation options” on page 417. For information about how to create multiple installation options modules, see the Installation System documentation.

**UNLOAD PLUS STEPLIB DD statement**

The UNLOAD PLUS STEPLIB DD statement must specify the following libraries, unless they are included in your system’s LINKLIST or in a JOBLIB statement:

- Load libraries that contain the files (including the options modules) for the following BMC products and components:
  - UNLOAD PLUS
  - BMCSORT (AUP)
  - DB2 Utilities Common Code (D2U)
  - DB2 Solution Common Code (SCC)

- Libraries that contain any DB2 user exits (EDITPROCs, VALIDPROCs, FIELDPROCs, and user-written conversion routines)

- DB2 load library

All load libraries in the STEPLIB or JOBLIB concatenation must be APF authorized.

**UNLOAD PLUS DD statements**

UNLOAD PLUS uses data sets that are specified by ddnames.

The use of these data sets is optional unless specified otherwise. This section provides specification guidelines, allocation information, and usage notes for each of the data sets that UNLOAD PLUS uses. Use the following table to quickly find the data set for which you want more information.
### Table 61: ddname descriptions

<table>
<thead>
<tr>
<th>Data set type</th>
<th>Description reference</th>
<th>Default ddname</th>
<th>ddname option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command input</td>
<td>“SYSIN data sets” on page 285</td>
<td>SYSIN</td>
<td>Not applicable</td>
</tr>
<tr>
<td>DDL definition (for input object)</td>
<td>“DDLIN data sets in UNLOAD PLUS” on page 278</td>
<td>DDLIN</td>
<td>DDLDDN</td>
</tr>
<tr>
<td>Input copy</td>
<td>“UNLOAD PLUS input copy data set” on page 279</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Message output</td>
<td>“SYSPRINT data sets” on page 285</td>
<td>SYSPRINT</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Message output</td>
<td>“SYSPRIN2 data sets” on page 285</td>
<td>SYSPRIN2</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Other</td>
<td>“Other data sets” on page 308</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Output control</td>
<td>“SYSCNTL data sets in UNLOAD PLUS” on page 284</td>
<td>SYSCNTL</td>
<td>CNTLDDN</td>
</tr>
<tr>
<td>Output data</td>
<td>“SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286</td>
<td>SYSREC</td>
<td>UNLOADDDN</td>
</tr>
<tr>
<td>Secondary output data</td>
<td>“SYSREC and SYSRED data sets in UNLOAD PLUS” on page 286</td>
<td>SYSRED</td>
<td>UNLOADDDN</td>
</tr>
<tr>
<td>Sort message output indicator</td>
<td>“UTPRINT data sets” on page 307</td>
<td>UTPRINT</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Sort work</td>
<td>“SORTWK data sets in UNLOAD PLUS” on page 281</td>
<td>SORTWK</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Thread cancelation report output</td>
<td>“BMCFORCE data sets” on page 278</td>
<td>BMCFORCE</td>
<td>Not applicable</td>
</tr>
<tr>
<td>VSAM linear input</td>
<td>“VSAMDD data sets in UNLOAD PLUS” on page 281</td>
<td>VSAMDD</td>
<td>INFILE VSAMDDPREFIX</td>
</tr>
</tbody>
</table>

Instead of designating DD statements in your JCL for output data sets, you can use output descriptors in the SYSIN command stream to dynamically allocate these data sets. For more information about dynamic allocation, see the following references:

- “Dynamic data set allocation in UNLOAD PLUS” on page 52
- “OUTPUT” on page 182
**BMCFORCE data sets**

You can specify a BMCFORCE DD statement in your JCL to allocate a data set to contain the thread cancelation report output that is generated when you specify one of the following options:
- `FORCE REPORTONLY`
- `FORCE_RPT YES with FORCE READERS or FORCE ALL`

If you do not specify a BMCFORCE DD statement in your JCL, the utility sends the report to your SYSPRINT.

**DDLIN data sets in UNLOAD PLUS**

This optional data set, which you may use with the INFILE `ddname` option, contains the DDL that defines all of the objects that are related to the table from which UNLOAD PLUS unloads rows.

UNLOAD PLUS uses the DDL in the DDLIN data set for DB2 object definitions instead of using object definitions in the DB2 catalog.

**Requirements**

Note the following requirements when using a DDLIN data set:

- You must specify the attributes of this data set as fixed length (RECFM is F, FB, or FBS), and the record length must be 80 bytes (the logical record length or `LRECL=80`). UNLOAD PLUS uses only columns 1 through 72.

- The data set must contain the following DDL for all objects that are related to the table from which UNLOAD PLUS is unloading rows:
  - `CREATE DATABASE`
  - `CREATE TABLESPACE`
  - `CREATE TABLE`
  - `CREATE INDEX`

- The DDL in the DDLIN data set must match the definition of the object that you are unloading.

- If either of the following conditions exists, you must specify the OBID of the table, either on each `CREATE TABLE` statement in the DDLIN data set or with each `SELECT` statement:
  - You specify multiple `SELECT` statements.
—You are unloading a multi-table table space.

Restrictions

The following restrictions apply to using a DDLIN data set:

■ If you specify LOGICAL PART on your UNLOAD command, UNLOAD PLUS ignores the LOGICAL keyword and considers the specified partitions to be physical partitions.

■ DDL in the DDLIN data set has the following restrictions:

—UNLOAD PLUS does not support the following syntax and ignores these statements in the DDLIN data set:

—Syntax to create materialized query tables (MQTs)
—CREATE VIEW statements
—ALTER DDL statements
—Syntax to create clone tables
—Character, hexadecimal, binary, and graphic string constants are limited to a length of 255 bytes.

UNLOAD PLUS input copy data set

UNLOAD PLUS requires this data set if you specify the INFILE option to unload from one of the following types of copy data set instead of a DB2 table space:

■ A full or incremental image copy
■ A DSN1COPY sequential data set
■ An inline copy

WARNING
Unpredictable results can occur if the data sets do not match the options that you specify in the INFILE command option.

For information about the data set needed when you use the INFILE option to unload from a VSAM linear data set, see “VSAMDD data sets in UNLOAD PLUS” on page 281.
Unloading from a single data set

If you specify the INFILE option and you are unloading from a single data set that contains an entire table space or one partition of a partitioned table space, you must have one DD statement that uses the ddname that you specified in the INFILE option.

Unloading an entire partitioned table space

If you are unloading an entire partitioned table space from multiple image copy data sets (one image copy data set for each partition of the table space), you must have one DD statement for each partition. Each DD statement must use the ddname that you specified in the INFILE option as a prefix and the partition number of the copy as a suffix, even if multiple image copy data sets are stacked as multiple files on a tape.

*Note*
Concatenating multiple image copy data sets, particularly when mixing copy data sets from tape with copy data sets from DASD, produces unpredictable results.

Unloading only selected partitions

If you are unloading only selected partitions, you must have one DD statement for each selected partition. Use the ddname that you specified in the INFILE option as a prefix and the partition number of the copy as a suffix.

Unloading from multiple table spaces

If you are unloading from multiple table spaces, you must have one DD statement for each selected table space. Use the ddname that you specified in the INFILE option as a prefix and the sequence in which the table space is referenced using the SELECT statements as a suffix.

For example, if the first SELECT statement specifies INFILE COPY, a DD statement with a ddname of COPY1 must represent the table space. If the second statement references a table in a different table space, you must have a DD statement with COPY2 for the ddname. If the second SELECT statement references a table in the same table space as the first, you do not need an additional DD statement for the copy.

*Note*
Concatenating multiple image copy data sets, particularly when mixing copy data sets from tape with copy data sets from DASD, produces unpredictable results.
VSAMDD data sets in UNLOAD PLUS

UNLOAD PLUS requires this data set if you specify the INFILE option to unload from a VSAM linear data set instead of a DB2 table space.

You can use either the default ddname (VSAMDD) or a ddname that you specify by using the INFILE VSAMDDPREFIX command option. Use the VSAMDDPREFIX option if you are unloading a partitioned object with more than 99 partitions.

For multiple partitions, add the partition number as a suffix to the ddname that you specify in your JCL.

**WARNING**

Unpredictable results can occur if the data sets do not match the options that you specify in the INFILE command option.

SORTWK data sets in UNLOAD PLUS

SORTWK data sets are the work files that BMCSORT uses.

When you specify DIRECT YES, BMCSORT uses the SORTWK data sets in the unload phase for processing.

For any job in which UNLOAD PLUS performs a sort, you must allocate SORTWK files in one of the following ways:

- (recommended) Have BMCSORT dynamically allocate SORTWK data sets.

- Explicitly specify SORTWK DD statements in your JCL. Use this option when you want to control the allocation of your SORTWK data sets.

**Dynamically allocating SORTWK data sets**

BMCSORT dynamically allocates SORTWK data sets under either of the following circumstances:

- You specifically request dynamic allocation through command or installation options as described in Table 62 on page 282.

- BMCSORT determines that it needs more sort work space than you have allocated in your JCL, and dynamic allocation is enabled through command or installation options.

Several factors affect this dynamic allocation. The following table describes the results of combining these factors.
### Table 62: Factors that affect dynamic allocation of sort work data sets

<table>
<thead>
<tr>
<th>SORTDEV or SORTNUM</th>
<th>Third parameter of BMCSORT DYNALOC</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTDEV specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORTNUM n specified (where n is greater than 0)</td>
<td>ON or OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the value is OFF, specifying a value greater than 0 for SORTDEV or specifying SORTNUM changes this value to ON.</td>
<td>If you specify a SORTNUM value that is greater than 32, BMCSORT allocates the number of data sets that it determines it needs, up to the specified number of data sets per task. Otherwise, BMCSORT allocates the number of data sets that it determines it needs up to 32 per task.</td>
</tr>
<tr>
<td>SORTDEV not specified and SORTNUM 0</td>
<td>ON</td>
<td>BMCSORT allocates the number of data sets that it determines it needs, up to 32 per task.</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>BMCSORT does not allocate any sort work data sets and attempts to perform sort processing in memory.</td>
</tr>
</tbody>
</table>

To ensure that BMCSORT has enough information to allocate SORTWK space accurately and efficiently, specify the ENUMROWS command option and a value greater than 0 for the SORTNUM command option.

#### Allocating SORTWK data sets in your JCL

If you cannot dynamically allocate SORTWK data sets, you must specify SORTWK DD statements in your JCL. The number of SORTWK files should be evenly divisible by the number of tasks, with a minimum of two files per task. Available resources might limit the number of concurrent tasks to four; therefore, if you allocate twelve SORTWK files, UNLOAD PLUS can run up to four tasks without wasting sort space.

**Note**

When determining the number of concurrent tasks to run, UNLOAD PLUS checks the amount of allocated SORTWK space. In these calculations, UNLOAD PLUS uses only the primary allocation because the secondary allocation is not guaranteed.

#### Calculating partitioned sort work space

If the table space is partitioned, you can use the formulas in this section to calculate the minimum amount of sort work space that UNLOAD PLUS requires. This strategy allows enough sort work space to perform the sort. If the resulting data set size is not optimal, the elapsed time for the unload increases, but the utility can complete processing, despite the space constraint.
To determine the data set size when you specify the partitions to unload (by using the PART option), calculate the following value, and divide it by the number of SORTWK data sets that you are allocating. Use values from your largest single partition.

\[
(\# \text{ rows selected} \times (\text{avg. output record length} + \text{longest key length} + 12 \text{ bytes})) \times \text{number of specified partitions}
\]

To determine the data set size when you are not specifying any partitions, calculate the following value and divide it by the number of SORTWK data sets that you are allocating. Use values from your largest single partition. MAXP is the maximum number of partitions that you want to unload concurrently with a single task.

\[
(\# \text{ rows selected} \times (\text{avg. output record length} + \text{longest key length} + 12 \text{ bytes})) \times \text{MAXP}
\]

**Calculating nonpartitioned sort work space**

To determine the data set size for a nonpartitioned table space, use the following formula to calculate the value, and divide it by the number of SORTWK data sets that you are allocating.

\[
\# \text{ rows selected} \times (\text{avg. output record length} + \text{longest key length} + 12 \text{ bytes})
\]

**Restrictions**

You cannot allocate a SORTWK data set as any of the following data set types:

- A VIO data set
- A tape data set
- A multi-volume data set
- A data set in an SMS storage group that specifies EXTENDED FORMAT YES

BMCSORT does not support SORTWK data sets that extend beyond 65535 tracks on a single volume.

**Consideration**

When unloading from multiple table spaces, UNLOAD PLUS attempts to use multiple tasks to unload partitioned table spaces first. After unloading the partitioned table spaces, UNLOAD PLUS uses multiple tasks to unload all remaining table spaces, assigning one table space per task. You must ensure that you allocate enough sort work space to the sort work data sets so that there is sufficient space to sort the largest participating table space. If you are using UNLOAD PLUS to unload from multiple table spaces, BMC recommends that you allocate enough sort work space to each set of sort work data sets so that there is sufficient space to sort the largest participating table space.
**SYSCNTL data sets in UNLOAD PLUS**

UNLOAD PLUS always requires a SYSCNTL data set if you specify the CNTLCARDS or CNTLDDN option.

This output data set contains the control statements that UNLOAD PLUS generates for the unloaded data.

**Considerations**

Consider the following information about using SYSCNTL data sets:

- BMC recommends that you do not use a partitioned data set (PDS).

- If you specify multiple data sets, you must use the SYSCNTLn form of the ddname.

- To override the default ddname, use the CNTLDDN option.

- If you specify data control block (DCB) attributes, you must specify one of the following sets of attributes:
  
  — For fixed-length data sets, RECFM can be F, FB, or FBS. The logical record length (LRECL) must be a minimum of 80 bytes.

  — For variable-length data sets, RECFM can be V or VB. LRECL must be a minimum of 84 bytes.

- If you do not specify DCB attributes, UNLOAD PLUS uses DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120).

*Using multiple SYSCNTL data sets with multiple SELECT statements*

When you use multiple SELECT statements, you can also specify multiple SYSCNTL DD statements. Specify DD statements for some or all of the SELECT statements.

When you specify a SYSCNTL data set for one or more SELECT statements, the SYSCNTL data set contains the control cards to create or load the objects that those SELECT statements name.

SELECT statements are numbered sequentially beginning with the number 1. Specify a ddname with the format SYSCNTLn, where n is the numerical position of the SELECT statement. When unloading a SELECT statement that does not have a corresponding data set in the JCL, and you have specified CNTLCARDS, UNLOAD PLUS puts the corresponding control cards in the default data set, SYSCNTL.
SYSIN data sets

The SYSIN data set is an input data set that contains the utility command. The utility always requires the SYSIN data set.

The UTILINIT phase reads, parses, and verifies the utility command that is provided in this data set. The data set’s attributes must be specified as fixed length (RECFM is F, FB, or FBS), and the record length must be 80 columns (LRECL=80). The utility uses only columns 1 through 72.

SYSPRINT data sets

The SYSPRINT data set is the output data set that contains the utility’s messages. The utility always requires the SYSPRINT data set.

You should not depend on the content and format of this data set (for example, as input to user-defined processes). Message content and format are subject to change without notice.

The utility overrides any data control block attributes that you specify in your JCL with DCB=(RECFM=VBA,LRECL=137,BLKSIZE=141).

Note the following considerations if you direct SYSPRINT to a tape or disk data set:

- The DSN messages from DB2 are lost.
- BMC does not recommend using the BUFNO parameter on the DD statement for this data set.

For information about the level of messages that the utility displays and how to change the message level, see the description for the MSGLEVEL utility parameter.

SYSPRIN2 data sets

The SYSPRIN2 data set is an optional output data set that contains the utility’s messages.

With the exceptions noted in this section, the content is identical to the content of the SYSPRINT data set.

SYSPRIN2 is not a substitute for SYSPRINT. If you include a SYSPRIN2 DD statement in your JCL, you must still include a SYSPRINT DD statement.
You should not depend on the content and format of this data set (for example, as input to user-defined processes). Message content and format are subject to change without notice.

The utility overrides any data control block attributes that you specify in your JCL with DCB=(RECFM=VBA,LRECL=137,BLKSIZE=141).

In a worklist environment, you can specify SYSPRIN2 DD SYSOUT=* to view output in real time from any of the BMC Utility products that run in that worklist.

Note the following considerations if you direct SYSPRIN2 to a tape or disk data set:

- The DSN messages from DB2 are lost.
- BMC does not recommend using the BUFNO parameter on the DD statement for this data set.

For information about the level of messages that the utility displays and how to change the message level, see the MSGLEVEL parameter description.

**SYSREC and SYSRED data sets in UNLOAD PLUS**

UNLOAD PLUS always requires a SYSREC data set.

This output data set contains the unloaded rows. If you specify multiple data sets, you must specify the SYSRECCnn form of the data set name. You may optionally specify a second data set (SYSRED) or second group of data sets (SYSREDnn).

SYSRED data sets must have a one-to-one correspondence with your SYSREC data sets.

For information about using multiple output data sets, see “Using JCL to specify multiple unload data sets” on page 289.

If you are unloading LOB or XML data to referenced files, SYSREC and SYSRED data sets contain the unloaded rows from the base table. For more information about referenced files, see “Using referenced files” on page 287.

**Overriding the default ddname**

To change the installation default ddname or ddname prefix SYSREC or SYSRED, use the UNLOADDN installation or command option.

If you specify more than 99 data sets, use the UNLOADDN command option to override the default data set name of SYSREC or SYSRED, specifying a ddname.
prefix that results in eight characters or less after UNLOAD PLUS appends the highest data set number.

**Allocating SYSREC and SYSRED data sets**

You can allocate SYSREC and SYSRED files in one of the following ways:

- Have UNLOAD PLUS dynamically allocate the data sets. For more information, see “Using dynamically allocated unload data sets” on page 294.

- Explicitly specify SYSREC and SYSRED DD statements in your JCL. Use this option when you want to control the allocation of these data sets.

BMC recommends that you do not combine dynamically allocated SYSREC data sets with SYSRED data sets that are allocated in your JCL.

**Using referenced files**

Instead of unloading LOB or XML data directly to unload data sets (SYSREC and SYSRED), you can unload it to files that are referenced in your unload data sets. These referenced files can be partitioned data sets (PDSs), extended partitioned data sets (PDSEs), or hierarchical file systems (HFSs). In this case, your unload data sets contain the unloaded rows only from the base table.

UNLOAD PLUS always allocates PDS or PDSE referenced files dynamically. UNLOAD PLUS incorporates the OUTPUT statement and its associated options, as well as the corresponding installation options, to support dynamic allocation of these data sets to disk. (These data sets cannot be allocated to tape.) You cannot allocate these data sets in your JCL, but you can use the output descriptors to control the allocation. You must ensure that you specify the SPACE and DIR options on the OUTPUT statement to provide UNLOAD PLUS with data set sizing information. ANALYZE processing is not available for these data sets.

For HFS referenced files, UNLOAD PLUS generates the files dynamically, but the file system must be preallocated.

You must use OUTPUT statements to enable UNLOAD PLUS to dynamically allocate PDS or PDSE data sets or HFS files. For more information about specifying options to allocate these files, see “Considerations when unloading LOB or XML data to referenced files” on page 60.

**Using multiple referenced output files**

When specifying DIRECT YES, you can unload LOB and XML data to multiple referenced files. When determining the number of referenced files to use, consider the following information:

- You can define any one of the following configurations of referenced files:
— One file for each LOB and XML column
— One file per base table space partition for each LOB and XML column
— One file per subset of base table space partitions for each LOB and XML column

■ A referenced file can contain data for only one LOB or XML column.

Allocating unload data sets in your JCL

Use the following information when allocating unload data sets in your JCL.

Calculating SYSREC and SYSRED size

If you are allocating a single SYSREC and SYSRED data set, you can determine the data set size if you calculate the PRIQTY as A, and the SECQTY as B - A where

A = sum for each table or view:
   (# rows selected * (avg. output record length + 4 bytes if variable length))
B = sum for each table or view:
   (# rows selected * (max. output record length + 4 bytes if variable length))

Considerations

The following considerations apply to allocating SYSREC and SYSRED data sets:

■ If you specify SPANNED YES to unload LOB or XML data to a VBS data set, do not specify SYSOUT=* on your SYSREC DD statement.

■ When you use a single SELECT statement on a partitioned table space, BMC recommends that you specify an unload data set for each partition to obtain optimal performance.

■ You cannot stack multiple SYSREC or SYSRED data sets on a single tape; UNLOAD PLUS opens all SYSREC and SYSRED data sets at the start of unload processing.

■ UNLOAD PLUS issues an informational message if you allocated the SYSREC or SYSRED data set with DISP=MOD, but the utility continues processing. UNLOAD PLUS honors the DISP=MOD specification, and appends the unloaded data to existing data in SYSREC.

■ UNLOAD PLUS terminates if you allocate a block size that is greater than MVS allows.

■ The BLKSIZE calculations for the SYSREC and SYSRED data sets must be equal.
Unless you specify one of the following command options, UNLOAD PLUS calculates the optimal record format, record size, and block size and overrides any data control block attributes that you specify in your JCL:

- USELRECL, which overrides the record format and record size
- SPANNED YES, which overrides the record format and record size
- RECFM VB, which overrides the record format
- MAXBLKSIZE, which overrides the block size

**Note**
If the calculated record size exceeds the optimal block size for the unload data set, UNLOAD PLUS creates a VBS file. To produce a VB file, you can specify the MAXBLKSIZE option to override the optimal block size.

---

**Using JCL to specify multiple unload data sets**

Creating multiple unload data sets depends on the following factors:

- The type of table space (partitioned or nonpartitioned) that UNLOAD PLUS is unloading
- The number of SELECT statements in the UNLOAD command
- The number of SYSREC DD statements that you specify in the JCL

**Unloading partitions of a partitioned table space to separate data sets**

To unload each partition of a partitioned table space to a separate data set, perform the following steps:

1. Specify a single SELECT statement in the UNLOAD command to unload the partitioned table space.
2. Specify a separate DD statement in the JCL for each partition.

**Note**
When using DIRECT NO, UNLOAD PLUS does not unload data from individual partitions to separate data sets. To unload data from individual partitions to separate data sets when using DIRECT NO, specify a SELECT statement that isolates each partition with multiple unload data sets.

The following table describes examples (shown in Figure 18 on page 291) of configurations for unloading partitions to separate data sets.
Table 63: Examples of unloading partitions to separate data sets

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The ddname that you specify for each partition must follow the format SYSRECNn, where SYSREC is the default prefix for the unload data set and nn is the partition number. The first example illustrates unloading a partitioned table space with each partition being unloaded to a separate data set. UNLOAD PLUS unloads any partitions for which you do not specify SYSREC DD statements in the JCL to the default data set, SYSREC.</td>
</tr>
<tr>
<td>2</td>
<td>The second example illustrates overriding the default name, unloading two partitions to the default data set (MYDD), and writing the remaining two partitions to separate data sets.</td>
</tr>
<tr>
<td>3</td>
<td>The third example illustrates unloading all partitions of a partitioned table space to a single data set. In this case, you supply a single DD statement that matches the default ddname. You may use a ddname of SYSREC or SYSREC1.</td>
</tr>
<tr>
<td>4</td>
<td>The fourth example illustrates unloading all partitions of a partitioned table space to a single SYSREC that is directed to tape.</td>
</tr>
</tbody>
</table>
Figure 18: Unloading a partitioned table space (part 1 of 2)

Directing each partition to a separate data set

```
//SYSREC1
//SYSREC2
//SYSREC3
//SYSREC4
UNLOAD
SELECT *
FROM
TABLE1
```

```
Partition 1
---
|         |
|         |
SYSSREC1
---
Partition 2
---
|         |
|         |
SYSSREC2
---
Partition 3
---
|         |
|         |
SYSSREC3
---
Partition 4
---
|         |
|         |
SYSSREC4
---
```

Directing selected partitions to separate data sets

```
//MYDD
//MYDD3
//MYDD4
UNLOAD
UNLOADDDN MYDD
SELECT *
FROM TABLE1
```

```
Partition 1
---
|         |
|         |
MYDD
---
Partition 2
---
|         |
|         |
MYDD3
---
Partition 3
---
|         |
|         |
MYDD4
---
```

Directing all partitions to a single data set

```
//SYSREC
UNLOAD
SELECT *
FROM
TABLE1
```

```
Partition 1
---
|         |
|         |
SYSREC
---
Partition 2
---
|         |
|         |
---
Partition 3
---
|         |
|         |
---
Partition 4
---
|         |
|         |
```
Using multiple SELECT statements

You can use one or multiple unload data sets when you specify multiple SELECT statements in an UNLOAD command. Specify DD statements for some or all of the SELECT statements. SELECT statements are numbered sequentially, beginning with the number 1. Specify a ddname with the format SYSRECnn, where nn is the numerical position of the SELECT statement.

The following table describes examples (shown in Figure 20 on page 293) of configurations for specifying multiple SELECT statements.

Table 64: Examples of multiple SELECT statements

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The first example illustrates unloading several tables within a single table space with each SELECT statement unloaded to a separate data set. UNLOAD PLUS unloads a SELECT statement for which you do not specify SYSREC DD statements in the JCL to the default data set, SYSREC.</td>
</tr>
<tr>
<td>2</td>
<td>The second example illustrates unloading two SELECT statements to a single data set, and unloading two SELECT statements to separate data sets. To unload all SELECT statements to a single data set, specify a single DD statement that matches the default ddname. You may use a ddname of SYSREC.</td>
</tr>
<tr>
<td>3</td>
<td>The third example illustrates unloading several tables to a single data set.</td>
</tr>
</tbody>
</table>
Figure 20: Directing multiple SELECT statements to one or more data sets

**Directing all SELECT statements to separate data sets**

```
SELECT * FROM TABLE1
```

```
TABLE 1 → SELECT 1 → SYSREC1
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 2 → SYSREC2
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 3 → SYSREC3
```

```
SELECT * FROM TABLE3
```

```
TABLE 3 → SELECT 4 → SYSREC4
```

**Directing some SELECT statements to separate data sets**

```
SELECT * FROM TABLE1
```

```
TABLE 1 → SELECT 1 → SYSREC1
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 2 → SYSREC
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 3 → SYSREC
```

```
SELECT * FROM TABLE3
```

```
TABLE 3 → SELECT 4 → SYSREC4
```

**Directing all SELECT statements to a single data set**

```
SELECT * FROM TABLE1
```

```
TABLE 1 → SELECT 1
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 2 → SYSREC
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 3 → SYSREC
```

```
SELECT * FROM TABLE3
```

```
TABLE 3 → SELECT 4
```

---

Chapter 4: Building and executing UNLOAD PLUS jobs
Using dynamically allocated unload data sets

Identical rules govern data sets that you allocate using JCL DD statements and those that UNLOAD PLUS allocates using dynamic allocation and output descriptors. UNLOAD PLUS can allocate or use a secondary unload data set only when it has a corresponding primary unload data set. For example, if a default primary unload data set exists and UNLOAD PLUS uses it for all SELECT statements, UNLOAD PLUS dynamically allocates only a single default secondary unload data set. If you specify a primary unload data set for each SELECT statement, UNLOAD PLUS can dynamically allocate a secondary unload data set for each SELECT statement.

When you use the UNLOADDN and OUTPUT commands and assign an output descriptor to a SELECT statement for the primary and secondary unload data sets, the data set name determines how many data sets UNLOAD PLUS creates for that SELECT statement. The primary and secondary unload data sets must resolve to the same number of data sets.

How to activate and configure dynamic allocation for unload data sets

You can activate dynamic allocation of unload data sets in the installation options module or in the command syntax:

1. Use one of the following methods to activate dynamic allocation:
   - To set the default for activating dynamic allocation of your primary and secondary unload data sets, specify UNLOADDN_ACTIVE=(YES,YES) in your installation options.
   - To activate dynamic allocation for primary and secondary unload data sets by using command options, specify UNLOADDN(primary,secondary) ACTIVE(YES,YES).

2. If either of the following conditions exists, specify either the ESTROWS, LIMIT, or SPACE command option to provide UNLOAD PLUS with data set sizing information:
   - DIRECT NO is in effect.
   - You specify the INFILE option.

   **Note**

   The SPACE installation option does not provide this information.

In all other cases, UNLOAD PLUS uses ANALYZE processing to obtain data set sizing information for dynamic allocation.
3 Ensure that the installation options in your $ADUOPTS macro are appropriate for your unload data sets, or use the OUTPUT command option overrides. For example, using symbolic variables can simplify construction of data set names for dynamically allocated data sets.

**Generating load control cards for LOADPLUS input data sets**

If you specify dynamic allocation, the unload data sets are not associated with a DD statement that allows a subsequent LOAD step to use traditional load control cards. LOADPLUS accepts actual data set names in load control cards. When you specify CNTLCARDS BMCLOAD, UNLOAD PLUS generates INDSN(DSN,...,DSN).

If the dynamically allocated unload data set is a generation data group (GDG), UNLOAD PLUS assigns the explicit cataloged data set name.

**Constructing data set names using symbolic variables**

When you use dynamic allocation with UNLOAD PLUS, symbolic variables can simplify the task of constructing data set names, as the following example shows:

```
UNLOAD UNLOADDN (DATAOUT)
OUTPUT DATAOUT UNIT CART
  DSNAME 'BMC.&DB.&TS.&UTIL(+1)'
SELECT *
  FROM BMC.EMPLS
```

When you specify a primary or secondary unload data set name in an output descriptor, you can use symbolic variables to specify any or all nodes of a data set name. For a list of variables that you can use, see the description of the DSNAME command option.

The previous example showed combining a real node name with symbolic variables. The following example shows using symbolic variables for all nodes:

```
DSNAME '&USERID.&TS.&TYPE'
```

In the following example, UNLOAD PLUS writes the data in each partition to a separate data set ('USER.P001', 'USER.P002', 'USER.P003'):

```
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
PART 1,2,3
SELECT * FROM TABLE1
OUTPUT SYSREC DSNAME '&USERID.P&PART'
```

You must prefix symbols for numeric variables with an alphabetic character. In the following example, the first statement is incorrect and the second statement is correct.
Incorrect:

```plaintext
UNLOAD  UNLOADDN (DATAOUT)
OUTPUT DATAOUT UNIT SYSDA
   DSNNAME 'BMC.&DB.&TS.&JDAY'
SELECT *
   FROM BMC.EMPLS
```

Correct:

```plaintext
UNLOAD  UNLOADDN (DATAOUT)
OUTPUT DATAOUT UNIT SYSDA
   DSNNAME 'BMC.&DB.&TS.&JDAY'
SELECT *
   FROM BMC.EMPLS
```

Although you can prefix a symbolic variable with an alphabetic character, you cannot append characters. For example, XX&TS is valid, but &TSXX is invalid. &TS.XX is valid.

To use symbolic variables to specify GDG data set names, append the generation number in parentheses. For example, you can use &TS(+1).

**Matching output descriptors to SELECT statements**

Substitution variables in the DSNNAME keyword can cause UNLOAD PLUS to generate multiple files for an OUTPUT descriptor name, especially when you specify the &PART substitution variable in the DSNNAME keyword for SELECT statements that unload a partitioned table.

If multiple SELECT statements share the same output descriptor that results in a single data set name, the statements share the same unload data set. Only SELECT statements that share the same dynamic allocation specification can resolve to the same data set name. If two SELECT statements each use a different output descriptor and their data set names resolve to the same real value, UNLOAD PLUS issues an error message and terminates.

**Note**

If you specify the nonsuffixed SYSREC DD in the JCL, UNLOAD PLUS does not dynamically allocate data sets. All SELECT statements that do not match a suffixed DD match the nonsuffixed DD.

**Matching output descriptors to multiple SELECT statements**

You can use multiple output descriptors when you specify multiple SELECT statements in an UNLOAD PLUS SYSIN command stream. Specify output descriptors for some or all of the SELECT statements, which are numbered sequentially, beginning with the number 1. Specify output descriptors with the format SYSRECNn, where nn is the number of the SELECT statement that you want to match. If you do not specify a SYSRECNn, UNLOAD PLUS uses the output...
descriptor SYSREC if you provide it in an OUTPUT statement, or UNLOAD PLUS uses the defaults that the installation options provide.

See the following section for examples and illustrations that show how to match SELECT statements and unloaded data sets and how to use the &PART and &SELNUM substitution variables in output descriptors.

**Using dynamic allocation to specify unload data sets: examples**

The following examples 1 through 5 describe various methods for specifying JCL when UNLOAD PLUS uses dynamic allocation.

**Figure 21: Example 1: No JCL DD statements: multiple SELECT statements with different output descriptors**

```plaintext
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM TABLE1
SELECT * FROM TABLE2
SELECT * FROM TABLE3
OUTPUT SYSREC DSNAME '&USERID.SYSREC'
OUTPUT SYSREC03 DSNAME '&USERID.SYSREC03'
```

This example includes multiple SELECT statements that have different output descriptors. The JCL does not include DD statements. Example 1 in Figure 28 on page 300 illustrates the results.

UNLOAD PLUS writes SELECT statement 1 from TABLE1 and SELECT statement 2 from TABLE2 to the dynamically allocated data set 'USER.SYSREC' that OUTPUT SYSREC specifies. UNLOAD PLUS writes the third SELECT statement to the dynamically allocated data set 'USER.SYSREC03' that OUTPUT SYSREC03 specifies.

**Figure 22: Example 2a: No JCL DD statements: multiple SELECT statements with identical output descriptor**

```plaintext
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM TABLE1
SELECT * FROM TABLE2
SELECT * FROM TABLE3
OUTPUT SYSREC DSNAME '&USERID.SYSREC'
```

This example includes multiple SELECT statements that have an identical output descriptor. The JCL does not include DD statements. This example demonstrates how to combine all of the SELECT statements into a single primary unload data set when UNLOAD PLUS is using dynamic allocation. UNLOAD PLUS writes all SELECT statements to the dynamically allocated data set 'USER.SYSREC'. Example 2a in Figure 28 on page 300 illustrates this scenario.

Furthermore, UNLOAD PLUS can write each SELECT statement to a separate dynamically allocated primary unload data set. In the following example, UNLOAD PLUS writes each SELECT statement to separate data sets named 'USER.S001','USER.S002', and 'USER.S003'. The substitution variable &SELNUM
(SELECT number) allows UNLOAD PLUS to create multiple unload data sets. Example 2b in Figure 28 on page 300 illustrates this scenario.

Figure 23: Example 2b: No JCL DD statements: multiple SELECT statements using the &SELNUM variable

```plaintext
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM TABLE1
SELECT * FROM TABLE2
SELECT * FROM TABLE3
OUTPUT SYSREC DSNAME '&USERID.S&SELNUM'
```

Figure 24: Example 3: No JCL DD statements: single SELECT statement and using the &PART variable

```plaintext
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
PART 1,2,3
SELECT * FROM TABLE1
OUTPUT SYSREC DSNAME '&USERID.P&PART'
```

This example has only one SELECT statement and includes the &PART symbolic variable in the data set name specified on the OUTPUT statement. The JCL does not include DD statements. UNLOAD PLUS writes the data in each partition to a separate data set ('USER.P001', 'USER.P002', 'USER.P003'). Example 3 in Figure 29 on page 301 illustrates this scenario.

Figure 25: Example 4a: Mixture of JCL and dynamic allocation: multiple SELECT statements and using the &PART variable

```plaintext
//SYSREC01 DD DISP=(,CATLG)
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM PART3TB
SELECT * FROM NONPARTTB
OUTPUT SYSREC DSNAME '&USERID.P&PART'
```

This example includes a combination of JCL and dynamic allocation. Multiple SELECT statements are combined with the &PART variable. Example 4a in Figure 29 on page 301 illustrates this scenario.

UNLOAD PLUS assigns SELECT statement 1 from PART3TB to SYSREC01, which has a JCL-allocated DD statement. UNLOAD PLUS writes SELECT statement 2 from NONPARTTB to data set 'USER.P000', which is dynamically allocated.

An additional example reverses the order of the SELECT statements to demonstrate that SELECT statement 2 produces more than one dynamically allocated data set. Example 4b in Figure 29 on page 301 illustrates this scenario.

Figure 26: Example 4b: Mixture of JCL and dynamic allocation: multiple SELECT statements and using the &PART variable

```plaintext
//SYSREC01 DD DISP=(,CATLG)
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
```
UNLOAD PLUS assigns SELECT statement 1 from NONPARTTB to SYSREC01, which has a JCL-allocated DD statement. No dynamic allocation is necessary for this SELECT statement.

UNLOAD PLUS writes SELECT statement 2 from PART3TB to dynamically allocated data sets USER.P001, USER.P002, USER.P003 (one for each partition that UNLOAD PLUS is unloading).

Figure 27: Example 5: Mixture of JCL and dynamic allocation: multiple SELECT statements with the &SELNUM variable

This example includes a combination of JCL and dynamic allocation using multiple SELECT statements. The output descriptor includes the &SELNUM variable. Example 5 in Figure 30 on page 302 illustrates this scenario.

UNLOAD PLUS writes SELECT statement 1 from TABLE1 to dynamically allocated data set USER.SYSREC.S001, SELECT statement 2 from TABLE2 to dynamically allocated data set USER.SYSREC.S002, and SELECT statement 3 to the data set to which the SYSREC03 DD in the JCL points.
Figure 28: Dynamic allocation examples (part 1 of 3)

**Example 1: Multiple SELECT statements with different output descriptors**

```
SELECT * FROM TABLE1
```

```
TABLE 1 → SELECT 1 → USER. SYSREC
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 2
```

```
SELECT * FROM TABLE3
```

```
TABLE 3 → SELECT 3 → USER. SYSREC03
```

**Example 2a: Multiple SELECT statements with identical output descriptors**

```
SELECT * FROM TABLE1
```

```
TABLE 1 → SELECT 1
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 2 → USER. SYSREC
```

```
SELECT * FROM TABLE3
```

```
TABLE 3 → SELECT 3
```

**Example 2b: All SELECT statements to separate primary unload data sets (using the &SELNUM variable)**

```
SELECT * FROM TABLE1
```

```
TABLE 1 → SELECT 1 → USER. S001
```

```
SELECT * FROM TABLE2
```

```
TABLE 2 → SELECT 2 → USER. S002
```

```
SELECT * FROM TABLE3
```

```
TABLE 3 → SELECT 3 → USER. S003
```
Figure 29: Dynamic allocation examples (part 2 of 3)

Example 3: Single SELECT statement (using the &PART variable)

Example 4a: Mixing JCL and dynamic allocation
(multiple SELECT statements with &PART variable)

Example 4b: Mixing JCL and dynamic allocation
(reversed multiple SELECT statements with &PART variable)

Unload data sets that UNLOAD PLUS produces from a DD statement specified in the JCL are shaded in this figure.
During dynamic allocation, UNLOAD PLUS uses your installation options and UNLOAD command options to choose a method for calculating the data set size of the unload data sets. UNLOAD PLUS uses the following priorities when determining the method:

1. If you specify SPACE on the OUTPUT statement, UNLOAD PLUS uses the SPACE parameters as you have specified them.

2. If you specify LIMIT, UNLOAD PLUS derives the total number of records per data set and calculates the data set size and space that it needs for allocating the data sets.

3. If you specify ESTROWS on the SELECT statement, UNLOAD PLUS uses your estimate to calculate the data set size.

4. If you specify ANALYZE DB2STATS or if you have specified the ANALYZE installation option, UNLOAD PLUS uses information about the table or partition that is available from the DB2 catalog to calculate the size of the unload data sets.

5. If you have not specified the SPACE, LIMIT, ESTROWS, or ANALYZE DB2STATS options, UNLOAD PLUS uses the table size or partition high-used RBA (HURBA) to allocate the unload data sets. For more information, see the description for the ANALYZE HURBA option.
6  If you do not specify the SPACE option, you can use the FILESZPCT command or installation option to adjust the total calculated space. See “FILESZPCT” on page 200 for more details.

The following examples show how this hierarchy works. The examples use the following table information:

<table>
<thead>
<tr>
<th>Table</th>
<th>Number of records</th>
<th>Statistics age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>20</td>
</tr>
</tbody>
</table>

**Table 65: Dynamic allocation examples data**

**Figure 31: Dynamic allocation example 1**

```
UNLOAD SYSREC
OUTPUT SYSREC01 SPACE (10,1)
OUTPUT SYSREC02
OUTPUT SYSREC03
OUTPUT SYSREC04
ANALYZE DB2STATS AGE 10
SELECT * FROM OWNER.TABLE1;
SELECT * FROM OWNER.TABLE2
ESTROWS 50;
SELECT * FROM OWNER.TABLE3;
SELECT * FROM OWNER.TABLE4;
```

In example 1, UNLOAD PLUS allocates

- 10 cylinders of space to SYSREC01 because the OUTPUT command has a specified value for SPACE

- Space for 50 records to SYSREC02 because an estimate for the number of rows is specified, but SYSREC02 does not have an assigned value for SPACE and LIMIT is not specified in the UNLOAD command

- Space for 100 records (obtained from the DB2 catalog information) to SYSREC03 because SYSREC03 does not specify a value for SPACE, LIMIT, or an estimated number of rows, and the catalog information is current (age less than the specified 10 days)

- Space based on the table space size (HURBA) to SYSREC04 because the statistics information for Table 4 in this example is 20 days old, and the catalog information is not current

**Figure 32: Dynamic allocation example 2**

```
UNLOAD
OUTPUT SYSREC
```
In example 2, the output from the first SELECT statement will be in the SYSREC01 data set, but the output from the other SELECT statements will be in the SYSREC data set. UNLOAD PLUS allocates

- 10 cylinders of space to SYSREC01 because a value is specified for SPACE
- Space for 60 records to SYSREC because the LIMIT value is 20 and SYSREC contains the unloaded output for the other three tables

**Figure 33: Dynamic allocation example 3**

<table>
<thead>
<tr>
<th>UNLOAD</th>
<th>ANALYZE HURBA</th>
<th>OUTPUT SYSREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM OWNER.TABLE4;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In example 3, UNLOAD PLUS uses the table space size to allocate space for SYSREC, because the JCL does not specify a value for SPACE, LIMIT, or ESTROWS and ANALYZE HURBA was specified.

**How UNLOAD PLUS calculates space for SMS-managed data sets**

UNLOAD PLUS can detect when the storage management subsystem (SMS) will manage a data set, and adjusts the primary and secondary allocations to make optimum use of the available DASD space to reduce abends that space limitations can cause. The SMS-managed space calculations and allocation parameters differ from those that SMS does not manage. These differences are as follows:

- UNLOAD PLUS always allocates SMS-managed data sets in tracks, regardless of the allocation type that you specify on the OUTPUT statement or in the installation options module.

- SMS routines in UNLOAD PLUS do not handle striped data sets. UNLOAD PLUS uses the non-SMS routines for allocating those data sets.

- UNLOAD PLUS allocates space in megabyte increments when the required space is greater than 1 megabyte, or in kilobyte increments when the required space is less than 1 megabyte. The smallest allocation that UNLOAD PLUS requests for tracks is 1 kilobyte.

- You can specify the value of MAXPRIM in either tracks or cylinders. UNLOAD PLUS converts that value to tracks, using the default geometry specified in the SMS base configuration.
**Determining the number of tracks to allocate**: When determining the number of tracks to allocate, UNLOAD PLUS uses the following method and priorities for the maximum record size and the number of required records for the data set allocation:

1. If you specify LIMIT, UNLOAD PLUS sums the limit value for each table space, partition, and SELECT statement that UNLOAD PLUS will direct to the data set.

2. If you specify ESTROWS on a SELECT statement and the table space is partitioned, UNLOAD PLUS proportions the value of ESTROWS to each partition that is unloaded and sums the proportioned values for each partition that is directed to this data set. If the input is from an image copy (that is, you specified INFILE in the command options), UNLOAD PLUS evenly distributes the value of ESTROWS among the partitions that it is unloading.

3. If the table is partitioned, UNLOAD PLUS sums the number of rows for each partition that is being unloaded for each SELECT statement that is directed to this data set.

4. UNLOAD PLUS sums the total table value for each SELECT statement that is directed to the data set.

5. UNLOAD PLUS uses the total number of records and the maximum record size to calculate the total required space in kilobytes.

6. If you do not specify the SPACE option, UNLOAD PLUS uses the value for the FILESZPCT command or installation option to adjust the total calculated space. See “FILESZPCT” on page 200 for more details.

**Calculating the SMS space**: After UNLOAD PLUS determines that SMS will manage a data set, UNLOAD PLUS obtains space information from the storage groups to calculate the space parameters. Because the Automatic Class Selection (ACS) routines could return more than one storage group for a specific data set, UNLOAD PLUS searches the storage groups for the first DASD group that contains sufficient space to hold the entire data set. If none of the storage groups has sufficient space, UNLOAD PLUS uses the first available storage group that is enabled.

**Note**

When UNLOAD PLUS requires guaranteed space because you supplied volume serial numbers on the OUTPUT statement, and the storage class specifies guaranteed space, UNLOAD PLUS considers only the volumes that you specify on the OUTPUT statement.

After determining whether it requires guaranteed space, UNLOAD PLUS selects either the standard SMS algorithm or the guaranteed space algorithm to calculate the space:
- **Standard SMS algorithm**

1. UNLOAD PLUS applies the value of PCTPRIM to calculate primary and secondary allocation values in tracks. If the value of PCTPRIM is AUTO, UNLOAD PLUS initially applies a value of 100%.

2. If you specify MAXPRIM, UNLOAD PLUS adds the difference between the calculated primary allocation and MAXPRIM to the secondary allocation value. UNLOAD PLUS sets the value of the primary allocation to the MAXPRIM value.

3. When the required primary allocation is greater than the largest free extent and the value of PCTPRIM is AUTO, UNLOAD PLUS sets the value of the primary allocation to the value of the largest free extent, and adds the difference to the secondary allocation value. If the value of PCTPRIM is not AUTO, UNLOAD PLUS generates an error message and terminates the job.

4. If the secondary allocation is zero, UNLOAD PLUS calculates 10% of the primary allocation to use as the secondary allocation value. If the secondary allocation value is less than the average free space per volume, UNLOAD PLUS sets the unit count to 1 and continues allocating with the next step. If the secondary allocation value is greater than the average free space per volume, UNLOAD PLUS divides the secondary allocation value by the average free space per volume to get the required number of units to meet the secondary allocation. If the required primary allocation is greater than the average free space per volume, UNLOAD PLUS adds 1 to the unit count. If you specify the UNITCNT option, UNLOAD PLUS uses that value. If volume serial numbers are supplied, UNLOAD PLUS uses the number of volume serial numbers. If either of these two values is less than the calculated number of units, UNLOAD PLUS issues a warning message but continues allocation.

5. If the value of NBRSECD is AUTO, UNLOAD PLUS sets NBRSECD to 123 if the SMS DATACLAS contains extended format. If the DATACLAS does not specify extended format, UNLOAD PLUS sets the value of NBRSECD to 16. If the calculated unit count is 1, UNLOAD PLUS reduces the value of NBRSECD by 1.

6. UNLOAD PLUS divides the secondary allocation value by the value for NBRSECD to get the track value for the secondary allocation.

7. The secondary allocation value is checked against the value specified for the MAXSECD option. If the secondary allocation value is less than or equal to the MAXSECD value, processing continues. Otherwise, the secondary value is set to the value of MAXSECD and the number of units required is recalculated.

- **Guaranteed space algorithm**
1 UNLOAD PLUS applies the value of PCTPRIM to calculate primary and secondary allocation values in tracks. If the value of PCTPRIM is AUTO, UNLOAD PLUS initially applies a value of 100%.

2 UNLOAD PLUS divides the value of the calculated primary allocation by the number of volume serial numbers that were supplied.

3 If you specify MAXPRIM, UNLOAD PLUS multiplies the difference between the calculated primary allocation and MAXPRIM by the number of volumes, and adds that value to the secondary value. The value of the primary allocation is set to the MAXPRIM value.

4 If the secondary allocation is zero, UNLOAD PLUS calculates 10% of the primary allocation to use as the secondary allocation value. When the required primary allocation is greater than the smallest free extent and the value of PCTPRIM is AUTO, UNLOAD PLUS sets the value of the primary allocation to the value of the smallest free extent and adds the difference multiplied by the number of volumes to the secondary allocation value. If the value of PCTPRIM is not AUTO, UNLOAD PLUS generates an error message and stops processing.

5 UNLOAD PLUS divides the value of the secondary allocation by the number of volume serial numbers that were supplied. If the secondary allocation value is greater than average free space per volume, UNLOAD PLUS issues a warning message but continues allocation.

6 If the value of NBRSECD is AUTO, UNLOAD PLUS sets NBRSECD to 123 if the SMS DATACLAS contains extended format. If the DATACLAS does not specify extended format, UNLOAD PLUS sets the value of NBRSECD to 16. If the calculated unit count is 1, UNLOAD PLUS reduces the value of NBRSECD by 1.

7 UNLOAD PLUS divides the secondary allocation value by the value for NBRSECD to get the tracks value for the secondary allocation.

8 The secondary allocation value is checked against the value specified for the MAXSECD option. If the secondary allocation value is less than or equal to the MAXSECD value, processing continues. Otherwise, the secondary value is set to the value of MAXSECD and UNLOAD PLUS issues message BMC50166W.

**UTPRINT data sets**

The utility always requires the UTPRINT data set if sorting is necessary.

The presence of this data set tells the utility to report sort messages. However, the actual messages for each sort process appear in separate SYSnnnnnn data sets, where nnnnnn is a system-assigned sequential number. You cannot specify a sequential or partitioned data set for UTPRINT. UTPRINT supports only SYSOUT data sets.
WARNING
JES3 users should be aware of a limitation within JES3 that does not allow concurrent tasks to share SYSOUT data sets. This limitation means that you cannot use additional sort routine reporting DDs (other than UTPRINT) if they are defined as JES3 SYSOUT data sets and when the utility is multitasking its sort activity. If you attempt to use an unsupported DD, you risk S1FB abends when concurrent sort tasks are running.

Other data sets
Because it uses BMCSORT, the utility ignores any traditional sort routine DD statements (such as $ORTPARM and DFSPARM) that you specify.

Running UNLOAD PLUS jobs
After you have built your UNLOAD PLUS job, the next step is to run the job. This section describes how to invoke a job, how to restart it, and how to terminate or cancel it.

Invoking UNLOAD PLUS
You normally invoke UNLOAD PLUS as a batch job by specifying execution of the module ADUUMAIN on the EXEC statement of your JCL and including the required EXEC statement parameters.

You must also specify any DD statements that UNLOAD PLUS requires, as described in preceding sections.

Ensure that all required libraries are available and APF-authorized as described in “UNLOAD PLUS STEPLIB DD statement” on page 276.

Restarting UNLOAD PLUS
Although UNLOAD PLUS accepts the values RESTART, RESTART(PHASE), NEW/RESTART, or NEW/RESTART(PHASE) if you specify one of them, the utility executes as though you specified the NEW value.

For more information, see “Restart parameter” on page 274.
Terminating or canceling an UNLOAD PLUS job

You can terminate an UNLOAD PLUS job by deleting the corresponding rows from the BMCUTIL and BMCSYNC tables or by specifying TERM on the restart parameter of the utility. If the job is currently running, it is terminated at the next sync point.

If you want to end the utility immediately, use the MVS or TSO CANCEL command to cancel the job.
Examples of UNLOAD PLUS jobs

This chapter presents several examples of jobs that were run by using the UNLOAD PLUS product.

Overview of UNLOAD PLUS examples

This chapter presents several examples of jobs for the UNLOAD PLUS product.

Each example includes the following information:

■ A description of the unload job

■ The UNLOAD PLUS job stream

■ The SYSPRINT from the job

Some examples show additional information that might be useful to understand a particular feature or to see the results of the unload job.

All of these examples have the following common properties:

■ These examples use a single BMC product load library, assuming that this library contains the UNLOAD PLUS load files and the common component load files that UNLOAD PLUS requires.

■ The examples use minimal data set space allocations in the JCL.

You can find copies of the JCL for these examples in members ADUXMnn (where nn is the example number) in the HLQ.LLQCNTL installation data set (where HLQ is the high-level qualifier specified during installation and LLQ is the low-level qualifier or prefix set during installation). The HLQ.LLQCNTL data set also includes the member ADUXM00 that contains the following elements:

■ DDL for the table spaces

■ Sample data to load into the table spaces
- Load jobs to load the data

For syntax details, see “Syntax of the UNLOAD command” on page 79. For information about JCL statement requirements, see “Building the UNLOAD PLUS job” on page 271.

Use the following table to locate an example of a specific unload type or option.

Table 66: Cross-reference of examples by function

<table>
<thead>
<tr>
<th>Functions</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unload features</strong></td>
<td></td>
</tr>
<tr>
<td>Change overpunch values during unload</td>
<td>17</td>
</tr>
<tr>
<td>Convert data during unload</td>
<td>2, 17, 18</td>
</tr>
<tr>
<td>Translate data during unload</td>
<td>18</td>
</tr>
<tr>
<td>Create a new field during unload</td>
<td>3</td>
</tr>
<tr>
<td>Use multiple SELECT statements</td>
<td>3, 4, 8, 9, 11, 13</td>
</tr>
<tr>
<td>Unload from prior image copy</td>
<td>10</td>
</tr>
<tr>
<td>Unload an entire table space</td>
<td>18</td>
</tr>
<tr>
<td>Use a DDLIN file</td>
<td>11</td>
</tr>
<tr>
<td>Unload to a CSV file</td>
<td>12</td>
</tr>
<tr>
<td>Unload to Unicode format</td>
<td>18</td>
</tr>
<tr>
<td>Unload selected columns</td>
<td>2, 4, 8, 13, 15</td>
</tr>
<tr>
<td>Unload selected partitions</td>
<td>5, 6, 10</td>
</tr>
<tr>
<td>Unload selected rows</td>
<td>4-7, 13, 14</td>
</tr>
<tr>
<td>Use a subselect to unload selected rows</td>
<td>13</td>
</tr>
<tr>
<td>Use a join to unload selected rows</td>
<td>13</td>
</tr>
<tr>
<td>Use columnar functions during unload</td>
<td>13</td>
</tr>
<tr>
<td>Unload using DB2 dynamic SQL</td>
<td>13-15</td>
</tr>
<tr>
<td>Handle an abnormal termination</td>
<td>14</td>
</tr>
<tr>
<td>Specify a nonstandard null indicator</td>
<td>15</td>
</tr>
<tr>
<td>Unload zoned decimal data</td>
<td>17</td>
</tr>
<tr>
<td>Use high-speed data migration with LOADPLUS</td>
<td>19</td>
</tr>
<tr>
<td>Unload XML data</td>
<td>15</td>
</tr>
<tr>
<td>Unload LOB data</td>
<td>15</td>
</tr>
<tr>
<td>Write to referenced files</td>
<td>15</td>
</tr>
<tr>
<td>Functions</td>
<td>Relevant examples</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>Object types</strong></td>
<td></td>
</tr>
<tr>
<td>Nonpartitioned table space</td>
<td>3, 8, 9, 11, 13-15, 17, 18</td>
</tr>
<tr>
<td>Partitioned table space</td>
<td>1, 2, 4-7, 9, 10, 12, 16, 19</td>
</tr>
<tr>
<td>DSN1COPY</td>
<td>5</td>
</tr>
<tr>
<td>Full image copy</td>
<td>6</td>
</tr>
<tr>
<td>Incremental image copy</td>
<td>10</td>
</tr>
<tr>
<td><strong>Command options</strong></td>
<td></td>
</tr>
<tr>
<td>ACTIVE NO</td>
<td>10, 12-14, 17, 18</td>
</tr>
<tr>
<td>ACTIVE YES</td>
<td>16</td>
</tr>
<tr>
<td>ANALYZE DB2STATS</td>
<td>16</td>
</tr>
<tr>
<td>AUTOTAG</td>
<td>18</td>
</tr>
<tr>
<td>CNTLCARDS</td>
<td>3, 5, 7, 8, 13, 15, 17-19</td>
</tr>
<tr>
<td>CNTLDDN</td>
<td>8</td>
</tr>
<tr>
<td>CONSISTENT YES</td>
<td>4</td>
</tr>
<tr>
<td>CURRENTDEGREE</td>
<td>13</td>
</tr>
<tr>
<td>DATEFMT</td>
<td>6</td>
</tr>
<tr>
<td>DELETENFILES</td>
<td>14</td>
</tr>
<tr>
<td>DIR</td>
<td>15</td>
</tr>
<tr>
<td>DIRECT NO</td>
<td>13-15</td>
</tr>
<tr>
<td>DIRECT YES</td>
<td>15</td>
</tr>
<tr>
<td>DSNNAME</td>
<td>7, 9, 15, 16, 19</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>15</td>
</tr>
<tr>
<td>ENUMROWS</td>
<td>9</td>
</tr>
<tr>
<td>ESTROWS</td>
<td>7</td>
</tr>
<tr>
<td>FILTERPART</td>
<td>7</td>
</tr>
<tr>
<td>FORMAT</td>
<td>12, 15, 17, 19</td>
</tr>
<tr>
<td>FIXEDVARCHAR</td>
<td>2, 17</td>
</tr>
<tr>
<td>IF NULL VALUE or IF predicate VALUE</td>
<td>2</td>
</tr>
<tr>
<td>IMAGECOPY</td>
<td>6, 10</td>
</tr>
<tr>
<td>INCREMENTAL</td>
<td>10</td>
</tr>
<tr>
<td>INFILE</td>
<td>5, 6, 10, 11</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>4</td>
</tr>
<tr>
<td>Functions</td>
<td>Relevant examples</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>INTO</td>
<td>1, 2, 3, 8, 15-17, 19</td>
</tr>
<tr>
<td>LIMIT</td>
<td>2, 4</td>
</tr>
<tr>
<td>NULLCHAR</td>
<td>15</td>
</tr>
<tr>
<td>NULLTYPE</td>
<td>15</td>
</tr>
<tr>
<td>OBID</td>
<td>5, 11</td>
</tr>
<tr>
<td>ON FAILURE</td>
<td>14</td>
</tr>
<tr>
<td>ORDER YES</td>
<td>2, 3, 19</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>6, 9</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>7, 9, 15, 16, 19</td>
</tr>
<tr>
<td>PART</td>
<td>5, 6, 10</td>
</tr>
<tr>
<td>RECORDID</td>
<td>8</td>
</tr>
<tr>
<td>RETCODE</td>
<td>14</td>
</tr>
<tr>
<td>SELECT</td>
<td>all examples except 18</td>
</tr>
<tr>
<td>SHRLEVEL CHANGE</td>
<td>4</td>
</tr>
<tr>
<td>SORTDEVT</td>
<td>3</td>
</tr>
<tr>
<td>SPACE</td>
<td>15, 16, 19</td>
</tr>
<tr>
<td>SUBSETS</td>
<td>15</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>18</td>
</tr>
<tr>
<td>TSFMT</td>
<td>6</td>
</tr>
<tr>
<td>UNLOADDN</td>
<td>8, 10, 12-14, 16-18</td>
</tr>
<tr>
<td>UNICODE</td>
<td>18</td>
</tr>
<tr>
<td>UNIT</td>
<td>15, 16, 19</td>
</tr>
<tr>
<td>WHERE clause</td>
<td>4-7, 14</td>
</tr>
<tr>
<td>XBMID</td>
<td>4</td>
</tr>
<tr>
<td>ZONEDDECOPV</td>
<td>17</td>
</tr>
</tbody>
</table>

**Example 1 - Unloading partitions to separate data sets**

In this example, UNLOAD PLUS unloads all rows from a partitioned table space with four partitions and unloads each partition to a separate data set.
The SYSPRINT shows that UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT *</td>
<td>Unloads all columns in the table</td>
</tr>
<tr>
<td>INTO</td>
<td>Converts columns to fields in the output table that have the specified data types</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 1:

**Figure 34: JCL for example 1**

```plaintext
// JOB
//UNLOAD01 EXEC PGM=ADUUMAIN,REGION=OM,COND=EVEN,
// PARM=(DEHJ,'ADUXMO1','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *

* UNLOAD SELECT *
  INTO
    EMPL_ID DECIMAL EXTERNAL (5,0),
      JOB_CODE SMALLINT EXTERNAL (2,0),
    SALARY DECIMAL EXTERNAL (9,2),
    ...
  FROM BMC.EMPLS

* //SYSCNTL DD DSN=ADU.EXAMPLE1.SYSCNTL,
  DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSCNTL DD DSN=ADU.EXAMPLE1.SYSCNTL02,
  DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSCNTL DD DSN=ADU.EXAMPLE1.SYSCNTL03,
  DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSCNTL DD DSN=ADU.EXAMPLE1.SYSCNTL04,
  DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
```
The following tables show the records that the job unloads in example 1. A NULL indicator field (NULLn in the table row headings) follows each nullable field. The numbers given for POS indicate the start and end positions for each field.

Table 68: Sample output for SYSREC01

<table>
<thead>
<tr>
<th>Field name and position</th>
<th>Record 1</th>
<th>Record 2</th>
<th>Record 3</th>
<th>Record 4</th>
<th>Record 5</th>
<th>Record 6</th>
<th>Record 7</th>
<th>Record 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPL_ID POS(1,5)</td>
<td>1.</td>
<td>17.</td>
<td>13.</td>
<td>19.</td>
<td>10.</td>
<td>4.</td>
<td>16.</td>
<td>12.</td>
</tr>
<tr>
<td>SSN POS(6,14)</td>
<td>490287572</td>
<td>387628543</td>
<td>763647562</td>
<td>574736232</td>
<td>487376242</td>
<td>337848646</td>
<td>382859832</td>
<td>363526475</td>
</tr>
<tr>
<td>BMC_NULL 1 POS(25)</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT_NO POS(26,29)</td>
<td>1100</td>
<td>1300</td>
<td>1300</td>
<td>2100</td>
<td>2400</td>
<td>2500</td>
<td>2500</td>
<td>2700</td>
</tr>
<tr>
<td>JOB_CODE POS(30,31)</td>
<td>13</td>
<td>13</td>
<td>93</td>
<td>44</td>
<td>44</td>
<td>93</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>BMC_NULL 2 POS(32)</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARY_CODE POS(33)</td>
<td>H</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>S</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>SALARY POS(34,42)</td>
<td>9.00</td>
<td>44000.00</td>
<td>4.75</td>
<td>76000.00</td>
<td>92000.00</td>
<td>8.10</td>
<td>24000.00</td>
<td></td>
</tr>
<tr>
<td>BMC_NULL 3 POS(43)</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PART401K POS(44)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC_NULL 4 POS(45)</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_LNAME POS(46,*)</td>
<td>Johnson</td>
<td>Broderick</td>
<td>Baker</td>
<td>Adkinson</td>
<td>Jackson</td>
<td>Jackson</td>
<td>Billingsly</td>
<td>Brown</td>
</tr>
<tr>
<td>EMPL_FNAME POS(<em>,</em>)</td>
<td>Raymond</td>
<td>Samuel</td>
<td>Kathy</td>
<td>William</td>
<td>William</td>
<td>David</td>
<td>Randy</td>
<td>Ellen</td>
</tr>
<tr>
<td>EMPL_MNAME POS(<em>,</em>)</td>
<td>James</td>
<td>Gerald</td>
<td>Lynn</td>
<td>Henry</td>
<td>Henry</td>
<td>Allan</td>
<td>George</td>
<td>Rebecca</td>
</tr>
</tbody>
</table>

Example 1 - Unloading partitions to separate data sets
Table 69: Sample output for SYSREC02

<table>
<thead>
<tr>
<th>Field name and position</th>
<th>Record 1</th>
<th>Record 2</th>
<th>Record 3</th>
<th>Record 4</th>
<th>Record 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPL_ID POS(1,5)</td>
<td>15.</td>
<td>13.</td>
<td>9.</td>
<td>2.</td>
<td>7.</td>
</tr>
<tr>
<td>SSN POS(6,14)</td>
<td>837625104</td>
<td>463762574</td>
<td>737626464</td>
<td>288738465</td>
<td>958473673</td>
</tr>
<tr>
<td>HIRE_DATE POS(15,24)</td>
<td>03/05/1990</td>
<td>08/26/1993</td>
<td>12/24/1994</td>
<td>04/18/1990</td>
<td></td>
</tr>
<tr>
<td>DEPT_NO POS(26,29)</td>
<td>3200</td>
<td>3300</td>
<td>3600</td>
<td>4200</td>
<td>4300</td>
</tr>
<tr>
<td>JOB_CODE POS(30,31)</td>
<td>44</td>
<td>84</td>
<td>93</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>SALARY_CODE POS(33)</td>
<td>H</td>
<td>S</td>
<td>S</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>SALARY POS(34,42)</td>
<td>6.25</td>
<td>32000.00</td>
<td>44500.00</td>
<td>32000.00</td>
<td></td>
</tr>
<tr>
<td>BMC_NULL3 POS(43)</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>PART401K POS(44)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BMC_NULL4 POS(45)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_LNAME POS(46,*)</td>
<td>Bronson</td>
<td>Doe</td>
<td>Welch</td>
<td>Britain</td>
<td>Franklin</td>
</tr>
<tr>
<td>EMPL_FNAME POS(*)</td>
<td>Parker</td>
<td>John</td>
<td>John</td>
<td>Michelle</td>
<td>Erica</td>
</tr>
<tr>
<td>EMPL_MNAME POS(*)</td>
<td>James</td>
<td>Frank</td>
<td>Elaine</td>
<td>Jean</td>
<td></td>
</tr>
</tbody>
</table>
Table 70: Sample output for SYSREC03

<table>
<thead>
<tr>
<th>Field name and position</th>
<th>Record 1</th>
<th>Record 2</th>
<th>Record 3</th>
<th>Record 4</th>
<th>Record 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPL_ID POS(1,5)</td>
<td>3.</td>
<td>18.</td>
<td>14.</td>
<td>6.</td>
<td>7</td>
</tr>
<tr>
<td>SSN POS(6,14)</td>
<td>828382664</td>
<td>182864652</td>
<td>376275478</td>
<td>868474632</td>
<td>463984856</td>
</tr>
<tr>
<td>HIRE_DATE POS(15,24)</td>
<td>04/03/1992</td>
<td>05/19/1989</td>
<td>07/31/1991</td>
<td>12/19/1989</td>
<td></td>
</tr>
<tr>
<td>BMC_NULL1 POS(25)</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT_NO POS(26,29)</td>
<td>5100</td>
<td>5100</td>
<td>5200</td>
<td>6300</td>
<td>6700</td>
</tr>
<tr>
<td>JOB_CODE POS(30,31)</td>
<td>13</td>
<td>24</td>
<td>43</td>
<td>13</td>
<td>73</td>
</tr>
<tr>
<td>BMC_NULL2 POS(32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAL_CODE POS(33)</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>SALARY POS(34,42)</td>
<td>5.75</td>
<td>7.50</td>
<td>6.25</td>
<td></td>
<td>6.90</td>
</tr>
<tr>
<td>BMC_NULL3 POS(43)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>PART401K POS(44)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>BMC_NULL4 POS(45)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>EMPL_LNAME POS(46,*)</td>
<td>Williamson George Slaughter Johnson Peterson</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_FNAME POS(<em>,</em>)</td>
<td>Floyd Hugh Johnathan Robert Barbara</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_MNAME POS(<em>,</em>)</td>
<td>Robert Gary Lawrence Ann</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 71: Sample output for SYSREC04

<table>
<thead>
<tr>
<th>Field name and position</th>
<th>Record 1</th>
<th>Record 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPL_ID POS(1,5)</td>
<td>20.</td>
<td>5.</td>
</tr>
<tr>
<td>SSN POS(6,14)</td>
<td>583872386</td>
<td>283846563</td>
</tr>
<tr>
<td>HIRE_DATE POS(15,24)</td>
<td>03/31/1993</td>
<td>04/25/1992</td>
</tr>
<tr>
<td>BMC_NULL1 POS(25)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SYSPRINT for example 1

The following figure shows the SYSPRINT for example 1:

#### Figure 35: SYSPRINT for example 1
Example 1 - Unloading partitions to separate data sets

UNLOAD PLUS for DB2 Reference Manual
Example 2 - Unloading columns in clustering order

This example unloads from the same partitioned table space that was used in example 1 but unloads the data in clustering order.

This job achieves optimum performance through multitasking.

The SYSPRINT shows the following information:

- Message BMC51701I reports that four tasks can run concurrently, and that each task can use up to 32 sort work data sets.

- UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).

- Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

Table 72: Command options for example 2

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER YES</td>
<td>Sorts output records in data-sorting key sequence</td>
</tr>
<tr>
<td>LIMIT 2</td>
<td>Unloads two records from each partition</td>
</tr>
<tr>
<td>Command options used in JCL</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>FIXEDVARCHAR YES</td>
<td>Tells UNLOAD PLUS to convert output records that would normally be variable-length to fixed length by padding variable-length columns with spaces</td>
</tr>
<tr>
<td>SELECT</td>
<td>Specifies columns to unload and the order in which to unload them</td>
</tr>
<tr>
<td>INTO</td>
<td>Specifies names and characteristics of output fields</td>
</tr>
<tr>
<td>IF NULL VALUE ('xx')</td>
<td>Eliminates NULL indicator fields and replaces nulls with the value that you specify</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 2:

**Figure 36: JCL for example 2**

```plaintext
// JOB //UNLOAD02 EXEC PGM=ADUUMAIN,REGION=OM, // PARM=(DEHJ,'ADUXM02','NEW ',,'MSGLEVEL(1)') //***************************************************************************** //STEPLIB DD DISP=SHR,DSN=product.libraries //      DD DISP=SHR,DSN=DB2.DSNEXIT //      DD DISP=SHR,DSN=DB2.DSNLOAD //SYSIN DD * *
* UNLOAD ORDER YES LIMIT 2 FIXEDVARCHAR YES SELECT DEPT_NO,EMPL_LNAME,SSN,HIRE_DATE,JOB_CODE,SALARY_HRLY,PART_401K INTO . . HIRE_DATE IF NULL VALUE('01/01/1991'), JOB_CODE CHARACTER(2) IF NULL VALUE(''), SALARY_HRLY DECIMAL EXTERNAL(9,2) IF NULL VALUE(999.99), PART_401K IF PART_401K = '0' VALUE('N') IF PART_401K = '1' VALUE('Y') IF PART_401K = NULL VALUE ('N') FROM BMC.EMPLS *
//SYSREC DD DSN=ADU.EXAMPLE2.SYSREC, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA //SYSCNTL DD DSN=ADU.EXAMPLE2.SYSCNTL, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA //UPRINT DD SYSOUT=* //SYSPRINT DD SYSOUT=*```

Table 73 on page 323 shows the records that the job stream in example 2 produced. UNLOAD PLUS sorts output records by the clustering key, DEPT_NO. The numbers given for POS indicate the start and end positions for each field.
Although UNLOAD PLUS unloads the same table as in example 1, this output does not include NULL indicator fields because the UNLOAD command includes IF NULL clauses for all nullable fields. Also, because FIXEDVARCHAR YES is specified, EMPL_LNAME in the output is a VARCHAR field with a length indicator, but the character part of the field is always 20 bytes long. UNLOAD PLUS pads the character part of this field with blanks.

### Table 73: Output record formats for example 2

<table>
<thead>
<tr>
<th>Record type</th>
<th>Record 1</th>
<th>Record 2</th>
<th>Record 3</th>
<th>Record 4</th>
<th>Record 5</th>
<th>Record 6</th>
<th>Record 7</th>
<th>Record 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT_NO</td>
<td>1100</td>
<td>1300</td>
<td>3200</td>
<td>3300</td>
<td>5100</td>
<td>5100</td>
<td>8200</td>
<td>8200</td>
</tr>
<tr>
<td>EMPL_LNAME</td>
<td>Johnson</td>
<td>Broderick</td>
<td>Bronson</td>
<td>Doe</td>
<td>Williamson</td>
<td>George</td>
<td>Kennedy</td>
<td>Wilson</td>
</tr>
<tr>
<td>SSN</td>
<td>490287572</td>
<td>387628543</td>
<td>837625104</td>
<td>463762574</td>
<td>828382664</td>
<td>182864652</td>
<td>583872386</td>
<td>283846563</td>
</tr>
<tr>
<td>HIRE_DATE</td>
<td>01/16/1993</td>
<td>08/23/1992</td>
<td>03/05/1990</td>
<td>01/01/1991</td>
<td>04/03/1992</td>
<td>05/19/1989</td>
<td>03/31/1993</td>
<td>04/25/1992</td>
</tr>
<tr>
<td>JOB_CODE</td>
<td>13</td>
<td>13</td>
<td>44</td>
<td>84</td>
<td>13</td>
<td>24</td>
<td>82</td>
<td>84</td>
</tr>
<tr>
<td>SALARY_H</td>
<td>9.00</td>
<td>44000.00</td>
<td>6.25</td>
<td>32000.00</td>
<td>5.75</td>
<td>7.50</td>
<td>36000.00</td>
<td>45000.00</td>
</tr>
<tr>
<td>PART_401K</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

**SYSPRINT for example 2**

The following figure shows the SYSPRINT for example 2:

**Figure 37: SYSPRINT for example 2**
Example 3 - Unloading multiple tables to a single data set

This job unloads multiple tables from a multi-table table space into one DB2 table with one UNLOAD command.

There is a clustering index on the table space, which UNLOAD PLUS uses to sort the output records. The job also inserts a constant value of SALARIED, HRLY, or RETIRED into the new field, ALL_EMPLS_STATUS, on the output record. Table 74 on page 325 describes the key command options for this job.

The JCL does not contain SORTWK DD statements, so BMCSORT dynamically allocates the sort work files. As specified by the SORTDEVT option, UNLOAD PLUS will use the SYSDA device type for any sort work files that BMCSORT dynamically allocates.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

Table 74: Command options for example 3

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLCARDS DB2</td>
<td>Writes DB2 CREATE TABLE DDL and DB2 LOAD control statements to the SYSCNTL data set</td>
</tr>
</tbody>
</table>
### Command options used in JCL

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER YES</td>
</tr>
<tr>
<td>Sorts output records in data-sorting key sequence</td>
</tr>
<tr>
<td>SORTDEVT SYSDA</td>
</tr>
<tr>
<td>Specifies device SYSDA for dynamically allocated sort work files</td>
</tr>
<tr>
<td>INTO</td>
</tr>
<tr>
<td>Specifies the table in the LOAD control statements that UNLOAD PLUS generates in SYSCNTL</td>
</tr>
<tr>
<td>FROM</td>
</tr>
<tr>
<td>Names tables from which to unload data</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 3:

**Figure 38: JCL for example 3**

```
// JOB
//UNLOAD03 EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ,'ADUXM03','NEW ',...'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
* UNLOAD CNTLCARDS DB2 ORDER YES SORTDEVT SYSDA *
* SELECT *, 'SALARIED'
  INTO NAME BMC.ALL_EMPLS
  ALL_EMPLS_LNAME
  ALL_EMPLS_FNAME
  ALL_EMPLS_SALCDE
  ALL_EMPLS_SALARY
  ALL_EMPLS_DEPTNO
  ALL_EMPLS_HIREDT
  ALL_EMPLS_CRTDATE
  ALL_EMPLS_STATUS CHARACTER(8)
  FROM BMC.SAL_EMPLS *
* SELECT *, 'HRLY'
  INTO NAME BMC.ALL_EMPLS
  ALL_EMPLS_LNAME
  ALL_EMPLS_FNAME
  ALL_EMPLS_SALCDE
  ALL_EMPLS_SALARY
  ALL_EMPLS_DEPTNO
  ALL_EMPLS_HIREDT
  ALL_EMPLS_CRTDATE
  ALL_EMPLS_STATUS CHARACTER(8)
  FROM BMC.HRLY_EMPLS *
* SELECT *, 'RETIRED'
  INTO NAME BMC.ALL_EMPLS
  ALL_EMPLS_LNAME
  ALL_EMPLS_FNAME
  ALL_EMPLS_SALCDE
  ALL_EMPLS_SALARY
  ALL_EMPLS_DEPTNO
  ALL_EMPLS_HIREDT
  ALL_EMPLS_CRTDATE
  ALL_EMPLS_STATUS CHARACTER(8)
  FROM BMC.RET_EMPLS *
* //SYSREC DD DSN=ADU.EXAMPLE3.SYSREC,
// DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
```
SYSPRINT for example 3

The following figure shows the SYSPRINT for example 3:

Figure 39: SYSPRINT for example 3

Example 3 - Unloading multiple tables to a single data set

SYSPRINT for example 3

The following figure shows the SYSPRINT for example 3:
Example 3 - Unloading multiple tables to a single data set

UNLOAD PLUS for DB2 Reference Manual
Example 4 - Unloading while allowing read write access to the table space

This example unloads a sample of rows from example tables while giving users read/write access to the table space.

By specifying SHRLEVEL CHANGE CONSISTENT YES, UNLOAD PLUS briefly restricts access to the object only while externalizing the pages of the table space. When operating on partitioned objects (as in this example), UNLOAD PLUS restricts access of only the participating objects.

Message 500411 indicates the status of zIIP processing. For this example, an XBM subsystem ID was specified for SHRLEVEL CHANGE processing. UNLOAD PLUS uses the same XBM subsystem ID for zIIP processing.

The following table describes key command options that this job uses:
Table 75: Command options for example 4

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIMIT 10</td>
<td>Unloads up to 10 records</td>
</tr>
<tr>
<td>INTERVAL 2</td>
<td>Unloads every other row</td>
</tr>
<tr>
<td>SHRLEVEL CHANGE CONSISTENT YES</td>
<td>Invokes XBM or SUF to allow concurrent read and write access during the unload</td>
</tr>
<tr>
<td>XBMID XBMB</td>
<td>Identifies the XBM subsystem for using XBM or SUF with UNLOAD PLUS</td>
</tr>
<tr>
<td>SELECT</td>
<td>Names columns to unload</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
<tr>
<td>WHERE</td>
<td>Qualifies the data to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 4:

**Figure 40: JCL for example 4**

```
// JOB
//UNLOAD04 EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ,'ADUXM04','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB  DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *

UNLOAD LIMIT 10 INTERVAL 2
SHRLEVEL CHANGE CONSISTENT YES XBMID XBMB

SELECT EMPL_LNAME, HIRE_DATE, DEPT_NO, SALARY_CODE
FROM BMC.EMPLS
WHERE SALARY_CODE = 'H'

SELECT EMPL_LNAME,EMPL_HIRE_DATE,EMPL_DEPT_NO,EMPL_SALARY_CODE
FROM BMC.RET_EMPLS
WHERE EMPL_SALARY_CODE = 'S'

//SYSREC01 DD DSN=ADU.EXAMPLE4.SYSREC01,
// DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSREC02 DD DSN=ADU.EXAMPLE4.SYSREC02,
// DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//UTPRINT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*
```

**SYSPRINT for example 4**

The following figure shows the SYSPRINT for example 4:

**Figure 41: SYSPRINT for example 4**

```text
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(c) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.

BMC50001I UTILITY EXECUTION STARTING 10/28/2013 13:02:42 ...
BMC50002I UTILITY ID = 'ADUXM04'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'ADU$OPTS'.
BMC50471I z/OS 2.1.0,PID=HBB7790,DFSMS FOR Z/OS=2.1.0,DB2=11.1.0
BMC50471I REGION=0M,BELOW 16M=8844K,ABOVE 16M=1412360K,IEFUSI=NO,CPUS=3
BMC50471I MEMLIMIT=17592186040320M,AVAILABLE=17592186040311M,MEMLIMIT SET BY:REGION=0

Example 4 - Unloading while allowing read write access to the table space

330 UNLOAD PLUS for DB2 Reference Manual
Example 4 - Unloading while allowing read write access to the table space

BMC50471I UNLOAD PLUS FOR DB2 - V11.01.00
BMC50471I NO MAINTENANCE TO REPORT
BMC50471I SOLUTION COMMON CODE - V11.01.00
BMC50471I MAINT: BP0061 BP0067 BP00671 BP00674 BP00675 BP00676 BP00682 BP00686 BP00689 BP00697
BMC50471I BMCSCONF ENGINES - V02.04.01
BMC50471I MAINT: BP0041
BMC50471I BMC STATS API - V11.01.00
BMC50471I MAINT: BP0049 BP00524 BP00674
BMC50471I EXTENDED BUFFER MANAGER - V06.01.00
BMC50471I MAINT: BP00401 BP00401 BP00405 BP00407 BP00410 BP00412 BP00416
BMC50471I ANALYZE(DB2STATS,NOLIMIT) FORCE-RPT-NO SMAX=16
BMC50471I CENTURY=(1900,2049) FORCE-NONE SMCDEF=(OK,OK)
BMC50471I CHANGE_*CONSISTENT-NO HISTORY=YES SORTNUM=22
BMC50471I CHANGE_QUIESCENCE-NO INTERVALS=5 SILENT=3
BMC50471I CMX=14 INTERVAL=3 SILENDTRY=100
BMC50471I CMRATIO=50 LOADSCALE=NO TAPEKEEP=DELETE
BMC50471I CONSTRUCTS=PMC LOCKSCALE=YES TASKMAX=200
BMC50471I CURRENTDEGREE=NONE MAP=7 UBUFS=250
BMC50471I DFILES=50 MSSMAX=1 UBUFS=250
BMC50471I DRNDelay=2 NUCLOSE=46 SYSDATE=(YES,YES)
BMC50471I DRNDelay=2 NULLOUT=9 SYSDATE=(YES,SYSDATE)
BMC50471I DRMWAIT=NONE OPDB2ID=NO USELRECL=NO
BMC50471I DSLOCSTR=DSLOCSTR PLAN=AUX000 USTATE=SUP
BMC50471I EBCDICENC=EXE CS1_USERID=SYSTEM RECP=RAFT WORKUNIT=SYSLABA
BMC50471I FILTERPN=NONE SYSPTEZ=100 ZIP=DISABLED
BMC50471I FREE AT=(START,3) SHLEVEL=REFERENCE

BMC50471I TAPES=NONE
BMC50471I OUTPUT = SYSREC SYSREC SYSREC
BMC50471I UNIT = SYSLABA SYSLABA SYSLABA
BMC50471I VOLCAT = 25 25 25
BMC50471I GDLIMIT = 5 5 5
BMC50471I EDGEEMPTY = NO NO NO
BMC50471I GDSCTRL = NO NO NO
BMC50471I STORCLAS = NONE NONE NONE
BMC50471I DATACLAS = NONE NONE NONE
BMC50471I MUPDATCLAS = NONE NONE NONE
BMC50471I UNICAT = 0 0 0
BMC50471I SPACE = CYL CYL CYL
BMC50471I PICPRM = AUTO AUTO AUTO
BMC50471I MAPEX = 0 0 0
BMC50471I MAPSEQ = 0 0 0
BMC50471I FILEZCT = 100 100 100
BMC50471I NMAPRC = AUTO AUTO AUTO
BMC50471I DISKTRAN = NONE NONE NONE
BMC50471I DISKEX = NONE NONE NONE
BMC50471I DISKRETN = NONE NONE NONE
BMC50471I NBRSECZ = AUTO AUTO AUTO
BMC50471I MAXPRIM = 0 0 0
BMC50471I PCTPRIM = AUTO AUTO AUTO
BMC50471I SPACE = CYL CYL CYL
BMC50471I UNIT = SYSLABA SYSLABA SYSLABA
BMC50471I VOLCAT = 25 25 25
BMC50471I GDLIMIT = 5 5 5
BMC50471I EDGEEMPTY = NO NO NO
BMC50471I GDSCTRL = NO NO NO
BMC50471I STORCLAS = NONE NONE NONE
BMC50471I DATACLAS = NONE NONE NONE
BMC50471I MUPDATCLAS = NONE NONE NONE
BMC50471I UNICAT = 0 0 0
BMC50471I SYSID = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSREC = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSPR = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSAUX = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSPAC = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSCS = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSDTA = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSTMP = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSREC = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSPR = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSCS = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSDTA = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I SYSTMP = VOLUME-NONE VOLUME-NONE VOLUME-NONE
BMC50471I TAPES=NONE
BMC50471I FILL=NONE SUMP=NO ZONEDECOPV=(C,0)
BMC50471I FREE AT=(START,3) SHLEVEL=REFERENCE

BMC50471I DB2 ONSHDEEP MODULE SETTINGS:
BMC50471I VERSION = 11.00.00
BMC50471I SUBSYSTEM DEFAULT = DEHJ
BMC50471I CHARACTER SET = ALPHANUM
BMC50471I DDATE FORMAT = USA
BMC50471I TIME FORMAT = USA
BMC50471I LOCAL DATE LENGTH = 0
BMC50471I LOCAL TIME LENGTH = 0
BMC50471I DECMAL POINT = PERIOD
BMC50471I DECIMAL ARITHMETIC = IN
BMC50471I DELLIMITER = DEFAULT
BMC50471I SOLLIMITER = DEFAULT
BMC50471I ENCODING SCHEME = EBCDIC
BMC50471I APPL. ENCODING SCHEME = EBCDIC
BMC50471I MIXED = NO
BMC50471I EBCDIC CCSID = (37,65534,65534)
BMC50471I ASCII CCSID = (819,65534,65534)
BMC50471I UNICOD CCSID = (387,1108,1108)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)
BMC50471I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL = "BMCUTIL.BMC_BMCUTIL"
BMC50471I BMC_BMCSYNC = "BMCUTIL.BMC_BMCSYNC"
BMC50471I BMC_BMCCHUNK = "BMCUTIL.BMC_BMCCHUNK"
BMC50471I BMC_BMCCOPY = "BMCUTIL.BMC_BMCCOPY"

BMC50471I * UNLOAD LIMIT 10 INTERVAL 2
BMC50471I * SHLEVEL CHANGE CONSISTENT YES XMID XMID
BMC50471I * SELECTEMPL_NAME, HIRE_DATE, DEPT_NO, SALARY_CODE
BMC50471I * FROM BMC_EMPLOY
BMC50471I * WHERE SALARY_CODE = 'H'
BMC50471I * SELECTEMPL_NAME,EMPL_HIRE_DATE,EMPL_DEPT_NO,EMPL_SALARY_CODE
BMC50471I * FROM BMC_EMPLOY
BMC50471I * WHERE EMP_SALARY_CODE = 'S'

BMC515641I DIRECT YES IN EFFECT

BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCDB0.BMCDB0 PART 1 IS 8 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCDB0.BMCDB0 PART 2 IS 5 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCDB0.BMCDB0 PART 3 IS 5 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCDB0.BMCDB0 PART 4 IS 2 FROM DB2STATS
BMC51688I ESTIMATED ROWS FOR TABLE BMC.EMPL IS 20 FROM DB2STATS
BMC51688I ESTIMATED ROWS FOR TABLE BMC.EMPL IS 20 FROM BMCDB0.BMCDB0
BMC51688I ESTIMATED ROWS FOR TABLE BMC.REMEMPL IS 6240 FROM TABLESPACE
BMC51688I ESTIMATED ROWS FOR TABLE BMC.REMEMPL IS 6240 FROM BMCDB0.BMCDB0
BMC51688I ESTIMATED ROWS FOR TABLE BMC.SALEMPL IS 6240 FROM TABLESPACE
BMC51688I ESTIMATED ROWS FOR TABLE BMC.SALEMPL IS 6240 FROM BMCDB0.BMCDB0

Chapter 5  Examples of UNLOAD PLUS jobs 331
**Example 5 - Unloading data from a DSN1COPY**

This example unloads four partitions of a four-partition table space. In this case, each partition has a separate DSN1COPY.

To accomplish this unload, the integer in the DSN1COPY ddname must match the partition. **Table 76 on page 332** describes the key command options for this job.

In the **SYSPRINT**, the echoed SAS statements show that some of the DB2 column names in this example exceed the SAS allowable length of 8 characters. For more information, see “Generating control statements for DB2 or other software products” on page 499.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

**Table 76: Command options for example 5**

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFILE DSN1CPY</td>
<td>Unloads data from a sequential data set rather than a DB2 table space data set</td>
</tr>
</tbody>
</table>
Command options used in JCL | Description
--- | ---
CNTLCARDS SAS | Generates SAS external file INPUT statements in the SYSCNTL data set
PART 4,1:3 | Designates the individual partitions and range of partitions that will be unloaded
While you can specify a mixture of individual partitions and ranges of partitions in any order, UNLOAD PLUS unloads all partitions in ascending order.
SELECT * | Unloads all columns
FROM | Names the table to unload
OBID | Specifies the OBID that will be associated with the selected table
WHERE | Qualifies the data to unload

The following figure shows the JCL for example 5:

**Figure 42: JCL for example 5**

```bash
// JOB
//UNLOAD05 EXEC PGM=ADUUMAIN,REGION=OM,
//     PARM=(DEHJ,'ADUXM05','NEW ','MSGLEVEL(1)')
//*********************************************************************
//STEPLIB  DD DISP=SHR,DSN=product.libraries
//     DD DISP=SHR,DSN=DB2.DSNEXIT
//     DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
* UNLOAD INFILE DSN1CPY
  CNTLCARDS SAS
  PART 4,1:3
  SELECT *
     FROM BMC.EMPLS {OBID 3}
     WHERE SSN > '200000000'
* 
//DSN1CPY1 DD DISP=SHR,DSN=ADU.QA.DSN1COPY.ADUEX05.P1
//DSN1CPY2 DD DISP=SHR,DSN=ADU.QA.DSN1COPY.ADUEX05.P2
//DSN1CPY3 DD DISP=SHR,DSN=ADU.QA.DSN1COPY.ADUEX05.P3
//DSN1CPY4 DD DISP=SHR,DSN=ADU.QA.DSN1COPY.ADUEX05.P4
//SYSREC01 DD DSN=ADU.EXMPELE05.SYSREC01,
//     DISP=(,CATLG),SPACE=(TRK,(1,1))
//SYSREC02 DD DSN=ADU.EXMPELE05.SYSREC02,
//     DISP=(,CATLG),SPACE=(TRK,(1,1))
//SYSREC03 DD DSN=ADU.EXMPELE05.SYSREC03,
//     DISP=(,CATLG),SPACE=(TRK,(1,1))
//SYSREC04 DD DSN=ADU.EXMPELE05.SYSREC04,
//     DISP=(,CATLG),SPACE=(TRK,(1,1))
//SYSCNTL DD DSN=ADU.EXMPELE05.SYSCNTL,
//     DISP=(,CATLG),SPACE=(TRK,(1,1))
//UTPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
```
The following figure shows the SYSPRINT for example 5:

Figure 43: SYSPRINT for example 5

nnenippopl for 08a 691.101.10

**** BMC UNLOAD PLUS FOR DB2 V11R1.00 ****

Example 5 - Unloading data from a DSN1COPY
Example 6 - Unloading a full image copy

This example unloads from a full image copy data set that is registered in the DB2 catalog.

The SYSPRINT shows that UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).
Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

**Note**
For an example of unloading an image copy other than the most recent one, see “Example 10 - Unloading from incremental image copies” on page 350. Although example 10 illustrates unloading an incremental image copy, the negative integer in the command option applies to either type of image copy.

The following table describes the key command options for this job:

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFILE IMAGECOPY</td>
<td>Unloads the most recent full image copy data set that UNLOAD PLUS finds in the SYSIBM.SYSCOPY table</td>
</tr>
<tr>
<td>TSFMT</td>
<td>Overrides the default DB2 external timestamp format and is useful for porting unloaded data to other relational databases</td>
</tr>
<tr>
<td>DATEFMT</td>
<td>Overrides the default DB2 external data format and also overrides your DB2 date exit routine. You can use this option to port unloaded data to other relational databases</td>
</tr>
<tr>
<td>PART 3</td>
<td>Designates the partitions to unload</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
<tr>
<td>WHERE</td>
<td>Qualifies the data to unload</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>sorts output records by the specified fields</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> To order by data-sorting key, use the ORDER YES statement as shown in example 2.</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 6:

**Figure 44: JCL for example 6**

```plaintext
// JOB
//UNLOAD06 EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ,'ADUXM06','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB  DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN    DD   *
*\nUNLOAD INFILE IMAGECOPY
TSFMT('MM.DD.YY/HH:MM:SS.NN')
DATEFMT('MM.DD.YY')
```
### SYSPRINT for example 6

The following figure shows the SYSPRINT for example 6:

#### Figure 45: SYSPRINT for example 6

```
FROM BMC.EMPLS WHERE SSN > '200000000'
ORDER BY HIRE_DATE DESC,EMPL_NAME
  
//SYSREC DD DSN=ADU.EXAMPLE6.SYSREC,
//     DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSCNTL DD DSN=ADU.EXAMPLE6.SYSCNTL,
//     DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//UTPRINT DD SYSPUT=*  
//SYSPRINT DD SYSPUT=*  
//SORTWK01 DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK02 DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK03 DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK04 DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(CYL,(1,1))
```
Example 7 - Using FILTERPART

This example unloads a partitioned table space with four partitions into two dynamically allocated output data sets.

In the SYSPRINT, message BMC50651I indicates which partitions are participating in the unload job.
Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SYSREC DSNAME</td>
<td>Specifies a default data set name for the output data set. This OUTPUT statement provides the dynamic allocation options for the output data set or sets.</td>
</tr>
<tr>
<td>FILTERPART</td>
<td>Filters out partitions that do not meet the selection criteria.</td>
</tr>
<tr>
<td>CNTLCARDS</td>
<td>Generates DB2 CREATE TABLE DDL and DB2 LOAD utility control statements in the SYSCNTL data set.</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns.</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload.</td>
</tr>
<tr>
<td>WHERE</td>
<td>Qualifies the data to unload.</td>
</tr>
<tr>
<td>ESTROWS</td>
<td>Specifies the estimated number of rows that you expect to unload and is used to calculate the size of the data sets that are produced during dynamic allocation.</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 7:

**Figure 46: JCL for example 7**

```plaintext
//        JOB
//UNLOAD07 EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM=(DEHJ,'ADUXM07','NEW ', 'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB  DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN    DD   *
*  UNLOAD
   OUTPUT SYSREC DSNAME 'ADU.EXAMPLE7.SYSCNTL'
   FILTERPART YES CNTLCARDS DB2
   SELECT *
      FROM BMC.EMPLS WHERE
         ( DEPT_NO > '5000')
            AND
            ( ( SALARY_HRLY > 1000.00 AND PART_401K = '0')
                OR
                (PART_401K = '1' AND HIRE_DATE > '01.01.1960'))
   ESTROWS 10
*  SYSCNTL DD  DSN=ADU.EXAMPLE7.SYSCNTL,
//     DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//UTPRINT  DD   SYSOUT=*  //SYSPRINT DD SYSOUT=*  
```
The following figure shows the SYSPRINT for example 7:

**Figure 47: SYSPRINT for example 7**

The following command is shown:

```
**Example 7 - Using FILTERPART**
```

```sql
BMC50471I 1/11/00

--- SYSPRINT for example 7 ---
```

```sql
BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FORCE_RPT=NO
```

```sql
BMC50471I   MAINT: BPE0401  BPE0403  BPE0405  BPE0407  BPE0410  BPE0412  BPE0416
```

--- SYSPRINT for example 7 ---
Example 7 - Using FILTERPART

Chapter 5  Examples of UNLOAD PLUS jobs  341
Example 8 - Using multiple SELECT statements

This example illustrates unloading data and generating control card statements to multiple output data sets.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options and DD statements for this job:

Table 79: Command options and DD statements for example 8

<table>
<thead>
<tr>
<th>Command options and DD statements used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLCARDS DB2 'ENUMROWS' 'REPLACE'</td>
<td>Tells UNLOAD PLUS to write DB2 CREATE TABLE DDL and DB2 LOAD control statements to the DD statements that CNTLDDN indicates. 'ENUMROWS (.n)' is added to the DB2 LOAD control statement, where n is the number of rows that UNLOAD PLUS writes to the output data sets. 'REPLACE' is added to the DB2 LOAD control statement.</td>
</tr>
<tr>
<td>UNLOADDDN DATA</td>
<td>Overrides the default ddname prefix of the unload data sets</td>
</tr>
<tr>
<td>CNTLDDN CNTL</td>
<td>Overrides the default ddname or default ddname prefix of the data set to which UNLOAD PLUS writes control statements</td>
</tr>
<tr>
<td>SELECT</td>
<td>Specifies columns to unload and the order in which to unload them</td>
</tr>
<tr>
<td>INTO</td>
<td>Specifies names and characteristics of output fields</td>
</tr>
<tr>
<td>RECORDID</td>
<td>Tells UNLOAD PLUS to include the specified string in the first field of each record unloaded by this SELECT statement. Because CNTLCARDS DB2 is also specified in this example, UNLOAD PLUS generates appropriate WHEN syntax for the DB2 load control statements.</td>
</tr>
<tr>
<td>FROM</td>
<td>Specifies a table to unload</td>
</tr>
<tr>
<td>//CNTL</td>
<td>Receives DB2 LOAD control statements that correspond to SELECT statements 1, 3, and 5 because DD statements are not specified for those SELECT statements. The CNTLDDN option specifies this ddname prefix.</td>
</tr>
</tbody>
</table>
Command options and DD statements used in JCL | Description
---|---
//CNTL2  
//CNTL4  
//CNTL6  | Receives DB2 LOAD control statements for SELECT statements 2, 4, and 6
//DATA  | Receives data unloaded by SELECT statements 1, 3, and 5
The UNLOADDN option specifies this ddname prefix.
//DATA2  
//DATA4  
//DATA6  | Receives data unloaded by SELECT statements 2, 4, and 6

The following figure shows the JCL for example 8:

**Figure 48: JCL for example 8**

```
// JOB
//UNLOAD08 EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM=(DEHJ,'ADUXMO8','NEW ','MSGLEVEL(1)')
//*********************************************************************
//STEPLIB  DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN   DD *
*
UNLOAD CNTLCARDS DB2 'ENUMROWS' 'REPLACE'
UNLOADDN DATA
CNTLDDN CNTL
SELECT EMPL_LNAME, EMPL_DEPT_NO, EMPL_HIRE_DATE
   INTO RECORDID 'S'
   FROM BMC.SAL_EMPLS
SELECT *
   FROM BMC.SAL_EMPLS
SELECT EMPL_LNAME, EMPL_DEPT_NO, EMPL_HIRE_DATE
   INTO RECORDID 'R'
   FROM BMC.RET_EMPLS
SELECT *
   FROM BMC.RET_EMPLS
SELECT EMPL_LNAME, EMPL_DEPT_NO, EMPL_HIRE_DATE
   INTO RECORDID 'H'
   FROM BMC.HRLY_EMPLS
SELECT *
   FROM BMC.HRLY_EMPLS
*
//CNTL   DD DSN=ADU.EXAMPLE8.CNTL,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//CNTL2  DD DSN=ADU.EXAMPLE8.CNTL2,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//CNTL4  DD DSN=ADU.EXAMPLE8.CNTL4,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//CNTL6  DD DSN=ADU.EXAMPLE8.CNTL6,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA   DD DSN=ADU.EXAMPLE8.DATA,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA2  DD DSN=ADU.EXAMPLE8.DATA2,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA4  DD DSN=ADU.EXAMPLE8.DATA4,
//         DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA6  DD DSN=ADU.EXAMPLE8.DATA6,
```
SYSPRINT for example 8

The following figure shows the SYSPRINT for example 8:

Figure 49: SYSPRINT for example 8
Example 8 - Using multiple SELECT statements

Chapter 5  Examples of UNLOAD PLUS jobs    345
Example 8 - Using multiple SELECT statements

346 UNLOAD PLUS for DB2 Reference Manual
Example 9 - Unloading from multiple table spaces

This example illustrates unloading data from multiple tables in multiple table spaces.

The SYSPRINT shows that UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options that this job uses:

Table 80: Command options for example 9

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SYSREC DSNAME</td>
<td>Specifies a default data set name for the output data set. This OUTPUT statement provides the dynamic allocation options for the output data set or sets.</td>
</tr>
<tr>
<td>ENUMROWS</td>
<td>Dynamically allocates sort work space based on this estimate of the number of rows that BMCSORT processes. This number applies to each table space.</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns. This example shows two SELECT statements, one from each of two tables. Each table is in a separate table space.</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Sorts output records by the fields specified. Note: To order by data-sorting key, use the ORDER YES statement as shown in example 2.</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 9:

Figure 50: JCL for example 9

```plaintext
// JOB
//UNLOAD09 EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ,'ADUXM09','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.Libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
UNLOAD
```
**SYSPRINT for example 9**

The following figure shows the SYSPRINT for example 9:

Figure 51: SYSPRINT for example 9

```sql
//SYSCNTL DD DSN=ADU.EXAMPL09.SYSCNTL,DISP=(NEW,CATLG),
//                    FROM     BMC.RET_EMPLS
//                    SELECT   *
//                    FROM     BMC.EMPLS
//                    SELECT   *
//                    ENUMROWS 100
//                    ORDER BY HIRE_DATE
//                    FROM     BMC.RET_EMPLS
//                    //SYSPRINT DD SYSDUMP=* //UTPRINT DD SYSDUMP=* //SYSDUMP DD SYSDUMP=* 
```

---

**Example 9 - Unloading from multiple table spaces**

The output of the above SQL queries will be used for unloading data from the table spaces. Example 9 demonstrates how to unload data from multiple table spaces using the `UNLOAD PLUS` utility, which is part of Unload Plus for DB2. This utility allows for efficient data movement from DB2 databases to various target systems, such as flat files, tapes, or other RDBMS systems.

The `UNLOAD PLUS` utility is particularly useful for data warehousing and data migration scenarios, where large volumes of data need to be transferred. It supports various options for customization, such as compression, parallel processing, and more, to optimize the data transfer process.

---

**Key Points**

- **Unloading from Multiple Table Spaces**: Example 9 illustrates how to unload data from multiple table spaces using SQL queries and the `UNLOAD PLUS` utility.
- **SYSPRINT Output**: The SYSPRINT output for example 9 is shown in the figure, illustrating the progress and results of the unload operation.
- **Data Movement Efficiency**: `UNLOAD PLUS` is designed to enhance the efficiency of data movement, making it a preferred choice for large-scale data migration projects.
Example 10 - Unloading from incremental image copies

This example shows how to unload from an incremental image copy data set that is registered in the DB2 catalog and illustrates how to unload from an image copy that is not the most recent one.

For an example of unloading a full image copy, see “Example 6 - Unloading a full image copy” on page 335.

The SYSPRINT shows that UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:
Table 81: Command options for example 10

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFILE IMAGECOPY INCREMENTAL -1</td>
<td>Unloads an incremental image copy data set that is registered in the SYSIBM.SYSCOPY table. The negative integer designates the number of incremental image copies that UNLOAD PLUS must count back from the current copy to find the incremental image copy to unload. UNLOAD PLUS can use the negative integer option for either type of image copy (incremental or full).</td>
</tr>
<tr>
<td>UNLOADDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module.</td>
</tr>
<tr>
<td>PART 1</td>
<td>Designates the partition to unload.</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns.</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload.</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 10:

**Figure 52: JCL for example 10**

```
// JOB
//UNLOAD10 EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ,'ADUXM10','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
//UNLOAD INFILE IMAGECOPY INCREMENTAL -1
// PART 1
// UNLOADDN SYSREC ACTIVE NO
// SELECT *
// FROM BMC.EMPLS
/*
//SYSCNTL DD DSN=ADU.EXAMPL10.SYSCNTL,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCREC DD DSN=ADU.EXAMPL10.SYSCREC,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=* 
//UTPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
```

**SYSPRINT for example 10**

The following figure shows the SYSPRINT for example 10:

**Figure 53: SYSPRINT for example 10**
Example 10 - Unloading from incremental image copies

UNLOAD PLUS for DB2 Reference Manual

352
Example 11 - Using a DDLIN data set

This example uses the OBID option and the INFILE option with a file that contains DDL.

The SYSPRINT shows that UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

Table 82: Command options for example 11

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFILE BMCCPY</td>
<td>Unloads data from a sequential data set rather than a DB2 table space data set</td>
</tr>
<tr>
<td>UNLOADDDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
<tr>
<td>Command options used in JCL</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OBID</td>
<td>Specifies the OBID that will be associated with the selected table. This option is valid only when using the INFILE <em>ddname</em> option. Additionally, you <em>must</em> use this option when the following conditions exist:</td>
</tr>
<tr>
<td></td>
<td>■ You specify INFILE <em>ddname</em> with DDL (using a DDLIN data set).</td>
</tr>
<tr>
<td></td>
<td>■ You are either specifying multiple SELECT statements or unloading a multi-table table space.</td>
</tr>
<tr>
<td></td>
<td>In this example, the OBID for table ADU.ADU11TB1 is specified in the UNLOAD command and the OBID for table ADU.ADU11TB2 is specified in the DDLIN file.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Specifies the column to unload</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 11:

**Figure 54: JCL for example 11**

```plaintext
// JOB
//UNLOAD11 EXEC PGMP=ADUUMAIN,REGION=0M,
//       PARM=(DEHJ,'ADUXM11','NEW ','MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
//UNLOAD INFILE BMCCPY
//UNLOADDN SYSREC ACTIVE NO
//SELECT 'ADU11TB1', * FROM ADU.ADU11TB1 {OBID 3}
//SELECT 'ADU11TB2', * FROM ADU.ADU11TB2
/*
//DDLIN DD DSN=ADU.VQA.EXAMPLES(ADUXDDL1),DISP=SHR
//       DD DSN=ADU.VQA.EXAMPLES(ADUXDDL2),DISP=SHR
//BMCCPY01 DD DSN=ADU.QA.DSNICOPY.ADU11TSIA,DISP=SHR
//BMCCPY02 DD DSN=ADU.QA.DSNICOPY.ADU11TSB,DISP=SHR
//SYSCTRL DD DSN=ADU.EXAMPL11.SYCSTM1,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSRECO1 DD DSN=ADU.EXAMPL11.SYSRECO1,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSRECO2 DD DSN=ADU.EXAMPL11.SYSRECO2,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=*
//UTPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

The following figure shows the first DDLIN data set that example 11 uses:

**Figure 55: First DDLIN data set for example 11**

```plaintext
DROP DATABASE ADU11DB;
COMMIT
```
DROP STOGROUP ADUSTG1;
COMMIT;

CREATE STOGROUP ADUSTG1
VOLUMES(DEV185,PJF001)
VCAT DEHJCAT;
COMMIT;

CREATE DATABASE ADU11DB
CCSID EBCDIC
STOGROUP ADUSTG1;
COMMIT;

CREATE TABLESPACE ADU11TSA IN ADU11DB
USING STOGROUP ADUSTG1
FREEPAGE 10
PCTFREE 10
BUFFERPOOL BP32K
CCSID EBCDIC
LOCKSIZE PAGE
CLOSE YES
SEGSIZE 4;
COMMIT;

CREATE TABLE ADU.ADU11TB1
(COL1_INTEGER INTEGER NOT NULL WITH DEFAULT,
 COL2_CHAR_20 CHARACTER(20) NOT NULL WITH DEFAULT,
 COL3_CHAR_20 CHARACTER(20) NOT NULL WITH DEFAULT,
 COL4_CHAR_100 CHARACTER(100) NOT NULL WITH DEFAULT,
 COL5_CHAR_100 CHARACTER(100) NOT NULL WITH DEFAULT,
 COL6_INTEGER INTEGER NOT NULL WITH DEFAULT
 )
IN ADU11DB.ADU11TSA
AUDIT NONE;
COMMIT;

The following figure shows the second DDLIN data set that example 11 uses:

**Figure 56: Second DDLIN data set for example 11**
SYSPRINT for example 11

The following figure shows the SYSPRINT for example 11:

Figure 57: SYSPRINT for example 11

The following figure shows the SYSPRINT for example 11:
Example 12 - Unloading to CSV format

This example illustrates the use of the FORMAT CSV command option to unload data in comma-separated-value (CSV) format.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

Table 83: Command options for example 12

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT CSV</td>
<td>Tells UNLOAD PLUS to unload all rows in CSV format, from which data can be ported to other databases</td>
</tr>
<tr>
<td>UNLOADDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
</tbody>
</table>
### Command options used in JCL

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT *</td>
<td>Unloads all columns</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 12:

**Figure 58: JCL for example 12**

```
//        JOB
//UNLOAD12 EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM=(DEHJ,'ADUXM12','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB  DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN    DD   *
* UNLOAD
FORMAT CSV ENCLOSEDBY '"' AND '"' NULLSTRING ''
UNLOADDN SYSREC ACTIVE NO
SELECT *
FROM BMC.EMPLS
* //SYSCNTL DD DSN=ADU.EXAMPL12.SYSCNTL,DISP=(NEW,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSREC   DD DSN=ADU.EXAMPL12.SYSCREC,DISP=(NEW,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=* 
//UTPRINT  DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
SYSPRINT for example 12
The following figure shows the SYSPRINT for example 12:

**Figure 59: SYSPRINT for example 12**

```

### SYSPRINT for example 12

The following figure shows the SYSPRINT for example 12:

**Figure 59: SYSPRINT for example 12**

```

----- Example 12 - Unloading to CSV format -----
Chapter 5  Examples of UNLOAD PLUS jobs  359

Example 12 - Unloading to CSV format
Example 13 - Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism

This example illustrates using DB2 dynamic SQL and parallelism to unload data from multiple tables.

This job unloads multiple tables from the same table space into three different SYSREC output files.

- The first SELECT statement shows a subselect query against two tables.
- The second SELECT statement joins two tables on the EMPL_LNAME column of the two tables.
- The third SELECT statement shows an example of column functions against a single table.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT NO</td>
<td>Tells UNLOAD PLUS to use DB2 dynamic SQL to process the SELECT statements and read the table data. Using DIRECT NO allows you to unload data by using the full range of functionality that the DB2 SQL SELECT statement provides. This example illustrates the use of several SQL functions that are not available when you specify DIRECT YES.</td>
</tr>
<tr>
<td>CNTLCARDS DB2</td>
<td>Writes DB2 CREATE TABLE DDL and DB2 LOAD control statements to the SYSCNTL data set</td>
</tr>
<tr>
<td>CURRENTDEGREE</td>
<td>Tells UNLOAD PLUS to issue the SET CURRENT DEGREE command before executing your dynamically executed SELECT statements. This option allows DB2 to take advantage of parallelism where parallel processing is possible.</td>
</tr>
<tr>
<td>UNLOADDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
<tr>
<td>SELECT</td>
<td>Names the columns to unload</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>
The following figure shows the JCL for example 13:

**Figure 60: JCL for example 13**

```plaintext
// JOB
//UNLOAD13 EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM=(DEHJ,'ADUXM13','NEW ',,'MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN    DD   *
UNLOAD
DIRECT NO
CNTLCARDS DB2
CURRENTDEGREE ANY
UNLOADDN SYSREC ACTIVE NO
*
SELECT EMPL_LNAME,
  MAX(EMPL_SALARY)
FROM BMC.SAL_EMPLS A
GROUP BY EMPL_LNAME
HAVING MAX(EMPL_SALARY) = (SELECT MAX(EMPL_SALARY_HRLY)
  FROM BMC.RET_EMPLS
  WHERE EMPL_LNAME= A.EMPL_LNAME);
*
SELECT DISTINCT BMC.EMPLS.EMPL_LNAME,
  SSN,
  EMP_ID
FROM BMC.EMPLS INNER JOIN BMC.RET_EMPLS
  ON BMC.EMPLS.EMPL_LNAME = BMC.RET_EMPLS.EMPL_LNAME;
*
SELECT EMPL_DEPT_NO,
  MAX(EMPL_HRLY_RATE),
  MIN(EMPL_HRLY_RATE)
FROM BMC.HRLY_EMPLS
GROUP BY EMPL_DEPT_NO;
*
//SYSCNTL DD DSN=ADU.EXAMPL13.SYSCNTL,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSREC01 DD DSN=ADU.EXAMPL13.SYSREC01,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSREC02 DD DSN=ADU.EXAMPL13.SYSREC02,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSREC03 DD DSN=ADU.EXAMPL13.SYSREC03,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=* 
//UTPRINT  DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
SYSPRINT for example 13

The following figure shows the SYSPRINT for example 13:

**Figure 61: SYSPRINT for example 13**

```
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
**** UNLOAD PLUS FOR DB2 V11R1.00 ******
BMC50000I UTILITY EXECUTION STARTING 10/28/2013 13:08:17...
BMC50001I UTILITY ID = 'ADUXM13'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'ADU$OPTS'.
BMC50471I z/OS 2.1.0,PID=HBB7790,DFSMS FOR Z/OS=2.1.0,DB2=11.1.0
BMC50471I REGION=0M,BELOW 16M=8848K,ABOVE 16M=1413172K,IEFUSI=NO,CPUS=3
BMC50471I MEMLIMIT=17592186040320M,AVAILABLE=17592186040311M,MEMLIMIT SET BY:REGION=0
BMC50471I UNLOAD PLUS FOR DB2--V11.01.00
BMC50471I   NO MAINTENANCE TO REPORT
BMC50471I DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I   NO MAINTENANCE TO REPORT
```

**SYSPRINT for example 13**

The following figure shows the SYSPRINT for example 13:

**Figure 61: SYSPRINT for example 13**

---

Chapter 5  Examples of UNLOAD PLUS jobs  361
UNLOAD PLUS for DB2 Reference Manual
**Example 14 - Handling an abnormal termination**

This example illustrates how to set up UNLOAD PLUS to delete specific files and issue a particular return code should an abnormal termination occur.

This job illustrates the following features:

- If the unload is unsuccessful and you specify **ON FAILURE ALL TERMINATE**

```sql
UTILITY RETCODE returnCode, UNLOAD PLUS terminates and issues the
```

---

Example 14 - Handling an abnormal termination

This example illustrates how to set up UNLOAD PLUS to delete specific files and issue a particular return code should an abnormal termination occur.

This job illustrates the following features:

- If the unload is unsuccessful and you specify **ON FAILURE ALL TERMINATE**

```sql
UTILITY RETCODE returnCode, UNLOAD PLUS terminates and issues the
```
return code that you specify. Having UNLOAD PLUS issue a particular return code enables you to perform an additional step based on that return code.

- If you also specify DELETEFILES YES, UNLOAD PLUS deletes all SYSREC, SYSRED, and SORTWK files. Using this feature reduces DASD overhead and results in better data integrity by eliminating data sets that could contain only part of the data.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

Table 85: Command options for example 14

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT NO</td>
<td>Tells UNLOAD PLUS to use DB2 dynamic SQL to process the SELECT statements and read the table data. Using DIRECT NO allows you to unload data by using the full range of functionality that the DB2 SQL SELECT statement provides. However, be aware that DIRECT NO is not a high-performance solution for unloading large volumes of data.</td>
</tr>
<tr>
<td>ON FAILURE</td>
<td>Tells UNLOAD PLUS how to handle an abnormal termination (when UNLOAD PLUS ends with a return code that is greater than or equal to 8, or when it abends)</td>
</tr>
<tr>
<td>ALL TERMINATE UTILITY RETCODE 11</td>
<td>Specifies that, if a failure occurs in any phase, you want the utility to terminate and issue return code 11</td>
</tr>
<tr>
<td>DELETEFILES YES</td>
<td>Deletes the SYSREC, SYSRED, and SORTWK files if a failure occurs</td>
</tr>
<tr>
<td>UNLOADDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns</td>
</tr>
<tr>
<td>FROM</td>
<td>Specifies the table to unload</td>
</tr>
<tr>
<td>WHERE</td>
<td>Qualifies the data to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 14:

Figure 62: JCL for example 14

```plaintext
// JOB
//UNLOAD14 EXEC PGM=ADUUMAIN,REGION=0M.
```
SYSPRINT for example 14

The following figure shows the SYSPRINT for example 14:

Figure 63: SYSPRINT for example 14
Example 15 - Unloading LOB and XML data; using nonstandard null indicators

This example illustrates unloading LOB and XML data and using nonstandard null indicators with UNLOAD PLUS.

This example consists of the following cases:

- Case 1 uses DIRECT NO processing and writes the LOB and XML data directly to the unload data set (SYSREC).

- Case 2 uses DIRECT YES and specifies referenced files to which UNLOAD PLUS will write LOB and XML data.

UNLOAD PLUS automatically includes the PRESERVE WHITESPACE keywords in the DB2 LOAD control cards for the XML column (message BMC51809I). If you do...
not want to preserve white space during the load, change the control cards before you run them.

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

**Table 86: Command options for example 15**

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT NO</td>
<td>Tells UNLOAD PLUS to use DB2 dynamic SQL to process the SELECT statements and read the table data (case 1 only)</td>
</tr>
<tr>
<td>DIRECT YES</td>
<td>Tells UNLOAD PLUS to use internal processing to process the SELECT statements and read the table data (case 2 only)</td>
</tr>
<tr>
<td>FORMAT STANDARD</td>
<td>Provides output data in DB2 internal format except date, time, and timestamp data types, which are in external format. Either FORMAT STANDARD or FORMAT INTERNAL is required when unloading LOB or XML data.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>Specifies the dynamic allocation options for the referenced files (case 2 only)</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>Specifies the type of referenced file to which UNLOAD PLUS will load (case 2 only)</td>
</tr>
<tr>
<td>DSNAME</td>
<td>Specifies the data set name or data set name pattern for the referenced file (case 2 only)</td>
</tr>
<tr>
<td>SUBSETS</td>
<td>Specifies how many referenced files to allocate per base table space partition or per simple or segmented table space (case 2 only)</td>
</tr>
<tr>
<td>NULLTYPE L2</td>
<td>Specifies that the null indicator is two bytes long, preceding the column</td>
</tr>
<tr>
<td>NULLCHAR HIVAL</td>
<td>Tells UNLOAD PLUS to fill the null indicator field with high values</td>
</tr>
<tr>
<td>CNTLCARDS DB2LOAD 'REPLACE'</td>
<td>Tells UNLOAD PLUS to write DB2 LOAD control statements. UNLOAD PLUS adds 'REPLACE' to the DB2 LOAD control statement.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Names the columns to unload</td>
</tr>
<tr>
<td>INTO</td>
<td>Specifies names and characteristics of output fields (case 2 only)</td>
</tr>
<tr>
<td>CLOBF</td>
<td>Specifies the output descriptor of the referenced file to which UNLOAD PLUS is to unload this column (case 2 only)</td>
</tr>
<tr>
<td>FROM</td>
<td>Specifies the table to unload</td>
</tr>
</tbody>
</table>
The following figure shows the JCL for example 15, case 1:

**Figure 64: JCL for example 15, case 1**

```
//UNLOA15A EXEC PGM=ADUUMAIN,REGION=0M,
//       PARM=(DEHJ,'ADUXM15','NEW ',,'MSGLEVEL(1)',ADU$OPTS)
//STEPLIB DD DISP=SHR,DSN=product.libraries
//       DD DISP=SHR,DSN=DB2.DSNEXIT
//       DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSREC DD DSN=ADU.EXAMP15A.SYSREC,DISP=(NEW,CATLG),
//       UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCNTL DD DSN=ADU.EXAMP15A.SYSCNTL,DISP=(NEW,CATLG),
//       UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//UTPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
/*
//SYSIN DD *
UNLOAD
DIRECT NO
FORMAT STANDARD
NULLTYPE L2
NULLCHAR HIVAL
CNTLCARDS DB2LOAD 'REPLACE '
SELECT
    COL_SMALL,
    COL_INTEGER,
    COL_FLOAT,
    COL_CHAR,
    COL_VCHAR,
    COL_DATE,
    COL_TIME,
    COL_TIMESTAMP,
    COL.XML,
    COL_CLOB
FROM ADU.TADU_XML4K_NULL_32K_PLUS_C
/*
```

The following figure shows the JCL for example 15, case 2:

**Figure 65: JCL for example 15, case 2**

```
//UNLOA15B EXEC PGM=ADUUMAIN,REGION=0M,
//       PARM=(DEHJ,'ADUXM15','NEW ',,'MSGLEVEL(1)',ADU$OPTS)
//STEPLIB DD DISP=SHR,DSN=product.libraries
//       DD DISP=SHR,DSN=DB2.DSNEXIT
//       DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSREC DD DSN=ADU.EXAMP15B.SYSREC,DISP=(NEW,CATLG),
//       UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCNTL DD DSN=ADU.EXAMP15B.SYSCNTL,DISP=(NEW,CATLG),
//       UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//UTPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
/*
//SYSIN DD *
UNLOAD
DIRECT YES
FORMAT STANDARD
OUTPUT CLOB01 DSNAME('ADU.EXAMP15B.CLOB')
    DSNTYPE(PDS) DIR (100) SPACE (10,10) SUBSETS 1 
    UNIT WORK
OUTPUT XML001 DSNAME('ADU.EXAMP15B.XML')
    DSNTYPE(PDS) DIR (100) SPACE (10,10) SUBSETS 1 
    UNIT WORK
/*
Example 15 - Unloading LOB and XML data; using nonstandard null indicators

### SYSPRINT for example 15, case 1

The following figure shows the SYSPRINT for example 15, case 1:

Figure 66: SYSPRINT for example 15, case 1
Example 15 - Unloading LOB and XML data; using nonstandard null indicators

UNLOAD PLUS for DB2 Reference Manual
SYSPRINT for example 15, case 2

The following figure shows the SYSPRINT for example 15, case 2:

Figure 67: SYSPRINT for example 15, case 2
Example 15 - Unloading LOB and XML data; using nonstandard null indicators

UNLOAD PLUS for DB2 Reference Manual
Example 16 - Using dynamic allocation

This example illustrates using output descriptors to dynamically allocate primary and secondary unload data sets.

You can specify output descriptors that have a suffix with a number that corresponds to each individual partition, as you can do with DD statements for data sets that you allocate in your JCL when you use a single SELECT statement to unload a partitioned table space. UNLOAD PLUS supports a separate dynamically allocated data set with unique allocation parameters for each partition.

This example uses the nonsuffixed output descriptor BACKUP with a DSNAME pattern that specifies the &PART substitution variable to allocate the secondary unload data set (BACKUP). UNLOAD PLUS can use the &PART variable to generate secondary unload data sets by partition.

The SYSPRINT shows that UNLOAD PLUS generates DB2 LOAD control cards by default (because the job does not include a CNTLCARDS option, but the JCL includes a SYSCNTL DD statement).

Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

Table 87: Command options for example 16

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SYSRECnn</td>
<td>Specifies the dynamic allocation options for the primary unload data set for partitions one through four. These OUTPUT statements provide the dynamic allocation options for the output data sets.</td>
</tr>
<tr>
<td>OUTPUT BACKUP</td>
<td>Specifies the dynamic allocation options for the secondary unload data set for all partitions.</td>
</tr>
</tbody>
</table>
### Command options used in JCL

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DSNAME ADU.EXAMPL16.BKUP.P&amp;PART</strong></td>
<td>Specifies the data set name or data set name pattern for the dynamically allocated unload data sets. ADU.EXAMPL16.BKUP.P&amp;PART specifies the data set name pattern for the BACKUP output descriptor, which is assigned to the secondary unload file. Because the number of dynamically allocated secondary data sets must match the number of primary data sets, UNLOAD PLUS requires the &amp;PART variable for allocating individual data sets for each partition.</td>
</tr>
<tr>
<td><strong>UNLOADDN SYSREC, BACKUP</strong></td>
<td>Specifies the output descriptor names to match to the OUTPUT statements for the primary and secondary unload data sets</td>
</tr>
<tr>
<td><strong>ACTIVE (YES,YES)</strong></td>
<td>Tells UNLOAD PLUS to dynamically allocate both the primary and secondary unload data sets</td>
</tr>
<tr>
<td><strong>ANALYZE DB2STATS</strong></td>
<td>Tells UNLOAD PLUS to use statistical information from the DB2 catalog to estimate the number of rows for a specific table and partition</td>
</tr>
<tr>
<td>**SELECT ***</td>
<td>Unloads all columns</td>
</tr>
<tr>
<td><strong>INTO</strong></td>
<td>Specifies names and characteristics of output fields</td>
</tr>
<tr>
<td><strong>FROM</strong></td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>

The following figure shows the JCL for example 16:

**Figure 68: JCL for example 16**

```bash
// JOB
//UNLOAD16 EXEC PGM=ADUUMAIN,REGION=0M,
PARM=(DEHJ,'ADUXM16','NEW ','MSGLEVEL(1)')
//*********************************************************************
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
* UNLOAD
OUTPUT SYSREC01 UNIT SYSDA
SPACE (5,1)
DSNAME 'ADU.EXAMPL16.SYSREC01'
OUTPUT SYSREC02 UNIT SYSDA
SPACE (5,1)
DSNAME 'ADU.EXAMPL16.SYSREC02'
OUTPUT SYSREC03 UNIT SYSDA
SPACE (3,1)
DSNAME 'ADU.EXAMPL16.SYSREC03'
OUTPUT SYSREC04 UNIT SYSDA
SPACE (3,1)
DSNAME 'ADU.EXAMPL16.SYSREC04'
OUTPUT BACKUP UNIT SYSDA
SPACE (5,1)
DSNAME 'ADU.EXAMPL16.BKUP.P&PART'
* UNLOADDN SYSREC,BACKUP ACTIVE(YES, YES)
```
The following figure shows the SYSPRINT for example 16:

**Figure 69: SYSPRINT for example 16**
Example 16 - Using dynamic allocation

UNLOAD PLUS for DB2 Reference Manual
Example 17 - Changing overpunch values in zoned decimal data

In this example, any data that is in zoned decimal format is assigned an overpunch value during the unload process.

UNLOAD PLUS formats zoned decimal values with the system default zone digit, which is ‘C’ for positive values and ‘B’ for negative values. You can use the ZONEDDECOVP option to assign overpunch values to zoned decimal numeric values. These values can be either positive or negative. Use this option when you plan to use the unloaded data with an application that requires a specific zone value that is not the traditional default.

The following table shows two examples of the values that are produced when you use the ZONEDDECOVP option. The first example in this table assigns an overpunch value of (C,D) (which is the default) and the second example assigns an overpunch value of (F,F).
Message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS automatically located an available XBM subsystem to provide zIIP processing.

The following table describes the key command options for this job:

**Table 89: Command options for example 17**

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT EXTERNAL</td>
<td>Tells UNLOAD PLUS to use external format for all output data types</td>
</tr>
<tr>
<td>UNLOADDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
<tr>
<td>ZONEDDECOVP (F,F)</td>
<td>Assigns an overpunch value of (F,F) to decimal-zoned numeric values. The overpunch values can be positive or negative. In this example, specifying ZONEDDECOVP (F,F) assigns positive overpunch values for any data that is in zoned decimal format.</td>
</tr>
<tr>
<td>FIXEDVARCHAR YES</td>
<td>Tells UNLOAD PLUS to convert output records that would normally be variable length to fixed length by padding variable-length columns with spaces</td>
</tr>
<tr>
<td>CNTLCARDS ’REPLACE COPYDDN (SYSCPY1)’</td>
<td>Tells UNLOAD PLUS to generate DB2 LOAD control cards (which is the default) and to add the REPLACE and COPYDDN (SYSCPY1) options to the DB2 LOAD control statement</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns</td>
</tr>
<tr>
<td>INTO</td>
<td>Specifies names and characteristics of output fields</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
</tbody>
</table>
The following figure shows the JCL for example 17:

**Figure 70: JCL for example 17**

```plaintext
// JOB
//UNLOAD17 EXEC PGM=ADUUMAIN,COND=EVEN,
//      PARM='DEHJ,ADUXM17,NEW,,MSGLEVEL(1)'
//STEPLIB DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *

UNLOAD FORMAT EXTERNAL
UNLOADDN SYSREC ACTIVE NO
ZONEDDECOVP(F,F)
FIXEVARCHAR YES
CNTLCARDS 'REPLACE COPYDDN (SYSCPY1)'
SELECT *
INTO
   COL1 DECIMAL ZONED (12)
   COL2 VARCHAR (10)
   COL3 DECIMAL ZONED (12)
   COL4 VARCHAR (10)
   COL5 DECIMAL ZONED (12)
   COL6 VARCHAR (10)
   COL7 DECIMAL ZONED (12)
   COL8 VARCHAR (10)
   COL9 DECIMAL ZONED (12)
   COL10 VARCHAR (10)
   COL11 DECIMAL ZONED (12)
   COL12 VARCHAR (10)
FROM BMC.LONGNAME_TABLE_ON_EXAMPLE17_TBEXMP17
/

//SYSREC DD DSN=ADU.EXAMPLE17.SYSREC,DISP=(NEW,CATLG),
//      UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCTNL DD DSN=ADU.EXAMPLE17.SYSCNTL,,DISP=(NEW,CATLG),
//      UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=*
//UTPRINT DD SYSOUT=*`
Example 17 - Changing overpunch values in zoned decimal data

BMC50471I ORSWAIT=NONE  DBNAME=SYSDBA  USESELCT=NO
BMC50471I DSNLOCK=NONE  PLAN=ADUQA  USETASK=UP
BMC50471I EXCLUD=NONE  RECFM=AUTO  WRTSETY=SYSSYSLDA
BMC50471I FILESIZE=NONE  RMTSETX=100  ZIIP=ENABLED
BMC50471I FILL-MH  SUMP=NO  ZONEDDCPY=(C,R)
BMC50471I FORCE_A=(START,3)  SHRLEVEL=REFERENCE
BMC50471I TAPES=NONE

BMC50471I OUTPUT  = SYSREC  SYSSYS  SYSEX
BMC50471I UNIT     = SYSSYSLDA  SYSSYSLDA  SYSSYSLDA
BMC50471I COLCTN   = 0S  ZS  ZS
BMC50471I GDGLIMIT = 5  5  5
BMC50471I GDGEMPTY = NO  NO  NO
BMC50471I GDSCRT  = NO  NO  NO
BMC50471I STORCAS = NONE  NONE  NONE
BMC50471I DBCASLS = NONE  NONE  NONE
BMC50471I UNCTYNT  = 0  0  D
BMC50471I SPACE     = CYL  CYL  CYL
BMC50471I PCTPRIM  = 0  0  0
BMC50471I R恢复 = 0
BMC50471I MAXPRIM  = 0  0  0
BMC50471I SPACE     = CYL  CYL  CYL
BMC50471I TRCH     = NONE  NONE  NONE
BMC50471I DONTYPE  = NONE  NONE  NONE
BMC50471I VOLUMES=NONE  VOLUMES=NONE  VOLUMES=NONE
BMC504831I SYSEX  VOLUMES=NONE  VOLUMES=NONE  VOLUMES=NONE
BMC504831I SYSSYS  VOLUMES=NONE  VOLUMES=NONE  VOLUMES=NONE
BMC504831I SYSREC  VOLUMES=NONE  VOLUMES=NONE  VOLUMES=NONE

BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 6230
BMC51688I ESTIMATED ROWS FOR TABLE BMC.LONGNAME_TABLE_ON_EXAMPLE17 TBEXMP17 IS 6230 FROM TABLESPACE
BMC50102I FROM BMC.LONGNAME_TABLE_ON_EXAMPLE17 TBEXMP17
BMC50102I ,COL12 VARCHAR (10)
BMC50102I ,COL11 DECIMAL ZONED (12)
BMC50102I ,COL10 VARCHAR (10)
BMC50102I ,COL9  DECIMAL ZONED (12)
BMC50102I ,COL8  DECIMAL ZONED (12)
BMC50102I ,COL7  DECIMAL ZONED (12)
BMC50102I ,COL6  VARCHAR (10)
BMC50102I ,COL5  DECIMAL ZONED (12)
BMC50102I ,COL4  VARCHAR (10)
BMC50102I ,COL3  DECIMAL ZONED (12)
BMC50102I ,COL2  VARCHAR (10)
BMC50102I ,COL1  DECIMAL ZONED (12)
BMC50102I INTO
BMC50102I CNTLCARDS 'REPLACE COPYDDN (SYSCPY1)'
BMC50102I FIXEDVARCHAR YES
BMC50102I ZONEDDECOVP (F,F)
BMC50102I UNLOADDN SYSREC ACTIVE NO

RMC51645I DIRECT YES IN EFFECT

BMC51687I ESTIMATED ROWS FOR TABLE SPACE DBEXMP17.TSEXMP17 PART 0 IS 6230 FROM MBULAR
BMC51688I ESTIMATED ROWS FOR TABLE BMC.LONGNAME_TABLE_ON_EXAMPLE17_TBEXMP17 IS 6230 FROM TABLESPACE
BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 6230

BMC50004I UNLOAD PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC50162I FOR NAME 'SYSEX' DSM-ADV.EXTB17.SYSEX.DSNB=RECFM-VB,LSIZE=27993,LRECL=152
BMC50894I PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 13:16:12
BMC50650I ESTABLISHING A POINT OF CONSISTENCY FOR:
BMC50651I SPACE DBEXMP17.TSEXMP17 PART(0) PSID=000002
BMC50695I PAGE EXTERNALIZATION PROCESS COMPLETE. ELAPSED TIME = 00:00:00
BMC50004I D: ZIIP ENABLED (D) USING IBM SUBSYSTEM VM
BMC50471I BELOW 16M = 8492K, ABOVE 16M = 1406204K, CPUS = 3
BMC50472I MA TH ble 1: 0 MA X PARTITIONS PER TASK = 1, SORTS PER TASK = 0, MA X OPEN PARTITIONS PER TASK = 1
BMC50477I 1: PARTITION - D, RWS/KEYS = 3, 1/R WAYS = 2, DNAME = 5Y500002
BMC50478I 1: ROB LOCK WAITS = 0
BMC50476I DNAME = SYSEX, I/OS = 1, I/OWAYS = 1, ROB LOCK WAITS = 0
BMC50486I UNLOAD STATISTICS: 3 ROWS PROCESSED FROM SPACE 'DBEXMP17.TSEXMP17', 0 NOT SELECTED, 0 DISCARDED
BMC50478I UNLOAD STATISTICS: 3 RECORDS WRITTEN TO DDNAME 'SYSREC'
BMC50477I UNLOAD STATISTICS: 3 ROWS PROCESSED FROM SPACE 'DBEXMP17.TSEXMP17', 0 NOT SELECTED, 0 DISCARDED
BMC50471I DDNAME = SYSREC, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50477I 1: RDB LOCK WAITS = 0
BMC50471I 1: PARTITION = 0, ROWS/KEYS = 3, I/O WAITS = 2, DDNAME = SYS00002

BMC50004I FOR NAME 'SYSEX' DSM-ADV.EXTB17.SYSEX.DSNB=RECFM-VB,LSIZE=3120,LRECL=80

380 UNLOAD PLUS for DB2 Reference Manual
Example 18 - Unloading a table space and unloading data as Unicode

In this example, UNLOAD PLUS unloads all of the data from an EBCDIC employee table space by using a Unicode encoding scheme (specified by using the UNICODE keyword).

After unloading the data as Unicode, LOADPLUS loads it into identically defined tables on another DB2 subsystem. This example includes both the UNLOAD PLUS JCL and SYSPRINT, and the LOADPLUS JCL and SYSPRINT.

Specifying UNLOAD TABLESPACE tells UNLOAD PLUS to generate a SELECT statement for each object in the table space. The functionality of this option is the same as if you had specified SELECT * FROM for every table in the table space. The AUTOTAG YES option adds a four-byte character constant value at the beginning of each output record for each generated SELECT statement. This tag identifies the table source when you use a single output data set.

In both jobs, message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS and LOADPLUS automatically locate an available XBM subsystem to provide zIIP processing.

The following table describes the UNLOAD PLUS key command options for this job:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD TABLESPACE</td>
<td>Tells UNLOAD PLUS to generate a SELECT statement for each object in the table space. The functionality of this option is the same as if you had specified SELECT * FROM for every table in the table space. The AUTOTAG YES option adds a four-byte character constant value at the beginning of each output record for each generated SELECT statement. This tag identifies the table source when you use a single output data set.</td>
</tr>
</tbody>
</table>

In both jobs, message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS and LOADPLUS automatically locate an available XBM subsystem to provide zIIP processing.

The following table describes the UNLOAD PLUS key command options for this job:
Table 90: UNLOAD PLUS command options for example 18

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD TABLESPACE</td>
<td>Unloads table space BMCEXDB.BMCEXBTS and generates SELECT statements for the following tables (see message BMC51657I and BMC51658I in the SYSPRINT output):</td>
</tr>
<tr>
<td></td>
<td>■ BMC.RET_EMPLS</td>
</tr>
<tr>
<td></td>
<td>■ BMC.HRLY_EMPLS</td>
</tr>
<tr>
<td></td>
<td>■ BMC.SAL_EMPLS</td>
</tr>
<tr>
<td>UNICODE</td>
<td>Tells UNLOAD PLUS to use the DB2 installation default (DSNHDECP) CCSIDs to encode the unloaded data in Unicode</td>
</tr>
<tr>
<td>AUTOTAG YES</td>
<td>Precedes each output record with a unique four-byte value for each generated SELECT statement</td>
</tr>
<tr>
<td>CNTLCARDS 'REPLACE'</td>
<td>Tells UNLOAD PLUS to write DB2 LOAD control statements</td>
</tr>
<tr>
<td></td>
<td>'REPLACE' is added to the DB2 LOAD control statement.</td>
</tr>
<tr>
<td>UNLOADDN SYSREC ACTIVE NO</td>
<td>Disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
</tbody>
</table>

The following table describes the LOADPLUS key command options for this job:

Table 91: LOADPLUS command options for example 18

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPLACE</td>
<td>Tells LOADPLUS to delete the existing data in the tables before loading</td>
</tr>
<tr>
<td>UNICODE</td>
<td>Specifies that the data that you are loading from the SYSREC data set is in Unicode format</td>
</tr>
<tr>
<td>CCSID</td>
<td>Specifies the three CCSIDs used to encode the data coming from UNLOAD PLUS</td>
</tr>
<tr>
<td>INTO TABLE</td>
<td>Identifies the data to load by specifying the table name</td>
</tr>
<tr>
<td>WHEN</td>
<td>Specifies the condition that must be true for LOADPLUS to select a row</td>
</tr>
</tbody>
</table>

The SYSIN DD statement in the LOADPLUS JCL specifies the name of the SYSCNTL data set that contains the LOAD command options that UNLOAD PLUS created. Passing the options from UNLOAD PLUS to LOADPLUS in this way streamlines the data migration process by eliminating the need to manually modify options in your LOADPLUS JCL.
The following figure shows the UNLOAD PLUS JCL for example 18:

**Figure 72: UNLOAD PLUS JCL for example 18**

```plaintext
// JOB
// UNLOADZA EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ, 'ADUXM18A', 'NEW ', 'MSGLEVEL(1)')
// STEPLIB DD DISP=SHR, DSN=product.libraries
// DD DISP=SHR, DSN=DB2.DSNEXIT
// DD DISP=SHR, DSN=DB2.DSNLOAD
// SYSREC DD DSN=ADU.EXAMPL18.UNICODE.SYSCNTL,DISP=(NEW,CATLG),UNIT=SYSDA
// SYSCNTL DD DSN=ADU.EXAMPL18.UNICODE.SYSCNTL,DISP=(NEW,CATLG),UNIT=SYSDA
// SYSPRINT DD SYSPRINT DD SYSOUT=*,
// UTPRINT DD UTPRINT DD SYSOUT=*
// SYSPRINT DD SYSIN DD *

UNLOAD
TABLESPACE BMCEXDB.BMCEXBTS
UNICODE
AUTOTAG YES
CNTLCARDS 'REPLACE'
UNLOADDN SYSREC ACTIVE NO
```

The following figure shows the LOADPLUS JCL for example 18:

**Figure 73: LOADPLUS JCL for example 18**

```plaintext
// JOB
// LODPLUS1 EXEC PGM=AMUUMAIN,
// PARM='DECS,ADUXM18B,NEW,,MSGLEVEL(1)'
// STEPLIB DD DISP=SHR, DSN=product.libraries
// DD DISP=SHR, DSN=DB2.DSNEXIT
// DD DISP=SHR, DSN=DB2.DSNLOAD
// SYSPRINT DD SYSPRINT DD SYSOUT=*
// UTPRINT DD UTPRINT DD SYSOUT=*
// SYSPRINT DD SYSOUT=*
// UTPRINT DD SYSOUT=*,HOLD=YES
// SYSPRINT DD SYSOUT=*,HOLD=YES
// SORTOUT DD DD DSN=SYSOUT,UNIT=WORK,DISP=(NEW,PASS),SPACE=(CYL,(1,1))
// SYSUT101 DD DSN=SYSUT11,UNIT=WORK,DISP=(NEW,PASS)
// SORTWK01 DD DSN=SORTWK01,UNIT=WORK,SPACE=(CYL,(1,1)),DISP=NEW
// SORTWK02 DD DSN=SORTWK02,UNIT=WORK,SPACE=(CYL,(1,1)),DISP=NEW
// SORTWK03 DD DSN=SORTWK03,UNIT=WORK,SPACE=(CYL,(1,1)),DISP=NEW
// SORTWK04 DD DSN=SORTWK04,UNIT=WORK,SPACE=(CYL,(1,1)),DISP=NEW
// SYSERR DD DSN=SYSERR,UNIT=WORK,SPACE=(CYL,(1,1)),DISP=(NEW,PASS)
// SYSDISC DD DSN=SYSDISC,UNIT=WORK,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
// SYSREC DD DSN=ADU.EXAMPL18.SYSCNTL,DISP=SHR
// SYSIN DD DSN=ADU.EXAMPL18.SYSCNTL,DISP=SHR
```
**SYSPRINT for example 18 - UNLOAD PLUS**

The following figure shows the UNLOAD PLUS SYSPRINT for example 18:

---

**Figure 74: UNLOAD PLUS SYSPRINT for example 18**

---

Example 18 - Unloading a table space and unloading data as Unicode

---

UNLOAD PLUS for DB2 Reference Manual
Example 18 - Unloading a table space and unloading data as Unicode

The following figure shows the LOADPLUS SYSPRINT for example 18:

Figure 75: LOADPLUS SYSPRINT for example 18

---

SYSPRINT for example 18 - LOADPLUS

The following figure shows the LOADPLUS SYSPRINT for example 18:

Figure 75: LOADPLUS SYSPRINT for example 18

---

Chapter 5 Examples of UNLOAD PLUS jobs 385
Example 18 - Unloading a table space and unloading data as Unicode

<table>
<thead>
<tr>
<th>BMS50470I DTYPE</th>
<th>S</th>
<th>SFMAP</th>
<th>LOADCPY</th>
<th>LOCBCPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMS50470I ACTIVE</td>
<td>NO</td>
<td>YES</td>
<td>USE</td>
<td>USE</td>
</tr>
<tr>
<td>BMS50470I SMS</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMS50470I SMUNIT</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMS50470I CIZECP</td>
<td>(100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
</tr>
<tr>
<td>BMS50470I UNIT</td>
<td>(SYSSALLA,SYSSALLA)</td>
<td>(SYSSALLA,SYSSALLA)</td>
<td>(SYSSALLA,SYSSALLA)</td>
<td>(SYSSALLA,SYSSALLA)</td>
</tr>
<tr>
<td>BMS50470I UNITCNT</td>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMS50470I AVGOLSP</td>
<td>(30000,TRK),(30000,TRK)</td>
<td>(30000,TRK),(30000,TRK)</td>
<td>(30000,TRK),(30000,TRK)</td>
<td>(30000,TRK),(30000,TRK)</td>
</tr>
<tr>
<td>BMS50470I DSTYPE</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I DATACLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I MGMTCLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I DATACLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I MAINLIN</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
</tr>
<tr>
<td>BMS50470I THRESHLD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMS50470I MATT = (0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
</tr>
<tr>
<td>BMS50470I EXPDT</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>BMS50470I RETPD</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>BMS50470I DDTYPE</td>
<td>LOCBXCPY</td>
<td>REMPXCPY</td>
<td>REMBXCPY</td>
<td>LOCBXCPY</td>
</tr>
<tr>
<td>BMS50470I MGMTCLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I DATACLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I AVGVOLSP</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td>BMS50470I VOLCNT</td>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMS50470I UNIT</td>
<td>(SYSSALLA,SYSSALLA)</td>
<td>(SYSSALLA,SYSSALLA)</td>
<td>(SYSSALLA,SYSSALLA)</td>
<td>(SYSSALLA,SYSSALLA)</td>
</tr>
<tr>
<td>BMS50470I UNITCNT</td>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMS50470I AVGOLSP</td>
<td>(30000,TRK),(30000,TRK)</td>
<td>(30000,TRK),(30000,TRK)</td>
<td>(30000,TRK),(30000,TRK)</td>
<td>(30000,TRK),(30000,TRK)</td>
</tr>
<tr>
<td>BMS50470I DSTYPE</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I DATACLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I MGMTCLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I DATACLAS</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMS50470I MAINLIN</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
</tr>
<tr>
<td>BMS50470I THRESHLD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMS50470I MATT = (0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
<td>(0),(0)</td>
</tr>
</tbody>
</table>
| **Example 18** - Unloading a table space and unloading data as Unicode
**Example 18 - Unloading a table space and unloading data as Unicode**

**UNLOAD PLUS for DB2 Reference Manual**

```
Example 18 - Unloading a table space and unloading data as Unicode

UNLOAD PLUS for DB2 Reference Manual
```
Example 19 - Unloading data to LOADPLUS

In this example, UNLOAD PLUS uses the FORMAT BMCLOAD option to unload data to an output file that only LOADPLUS can read.

The corresponding LOAD command includes the FORMAT BMCCUNLOAD option to load this data. The combination of these FORMAT options provides a high-speed data migration solution.

The load job replaces a partitioned table space that has one non-unique partitioning index that is not a clustering index and one non-unique, nonpartitioning index. No clustering index exists on the table space.

In both jobs, message 50041I indicates the status of zIIP processing. For this example, the ZIIP installation option is ENABLED, but an XBM subsystem was not specified. UNLOAD PLUS and LOADPLUS automatically located an available XBM subsystem to provide zIIP processing.

To provide a complete picture of these features, this example includes the UNLOAD PLUS JCL and SYSPRINT, the LOADPLUS JCL and SYSPRINT, and output showing the loaded table.

The following table describes the UNLOAD PLUS key command options for this job:

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SYSREC</td>
<td>Specifies dynamic allocation options for the default unload data set</td>
</tr>
<tr>
<td>CNTLCARDS</td>
<td>Passes LOADPLUS command options to the SYSCNTL data set</td>
</tr>
<tr>
<td></td>
<td>The control cards that are specified with this option will generate a LOAD REPLACE run with dynamic allocation. Because FORMAT BMCLOAD is specified, UNLOAD PLUS forces CNTLCARDS BMCCLOAD, here overriding the default of DB2LOAD.</td>
</tr>
<tr>
<td>ENUMROWS</td>
<td>Tells UNLOAD PLUS to determine the value to include with the LOADPLUS ENUMROWS option from the number of rows that are unloaded</td>
</tr>
</tbody>
</table>

Table 92: UNLOAD PLUS command options for example 19
The |UNLOAD PLUS| SYSPRINT (Figure 78 on page 392) shows the LOADPLUS control cards that UNLOAD PLUS writes to the SYSCNTL data set.

The following table describes the key command options for loading the data using LOADPLUS:

### Table 93: LOADPLUS command options for example 19

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER YES</td>
<td>Sorts output records in data-sorting key sequence</td>
</tr>
<tr>
<td>FORMAT BMCLOAD</td>
<td>Produces a high speed unload and an output file that only LOADPLUS can use when you are moving data to tables with an identical table structure</td>
</tr>
<tr>
<td>SELECT *</td>
<td>Unloads all columns</td>
</tr>
<tr>
<td>INTO</td>
<td>Passes to LOADPLUS the name of the table into which LOADPLUS is to load the data This table is identical in structure to the table from which UNLOAD PLUS is unloading data.</td>
</tr>
<tr>
<td>FROM</td>
<td>Names the table to unload</td>
</tr>
<tr>
<td>INDSN</td>
<td>Names the SYSREC data set that UNLOAD PLUS created UNLOAD PLUS builds the data set name dynamically based on the DSNAME defaults in the UNLOAD PLUS installation options.</td>
</tr>
<tr>
<td>ENUMROWS</td>
<td>Indicates the estimated number of rows to load This example shows that UNLOAD PLUS calculated the number of new rows that LOADPLUS will load as 29 This number includes the header records that UNLOAD PLUS creates when using FORMAT BMCLOAD. LOADPLUS uses these header records to ensure that the table being loaded matches the table that was unloaded.</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Tells LOADPLUS to delete the existing data before loading</td>
</tr>
<tr>
<td>DELETENFILES YES</td>
<td>Tells LOADPLUS to delete all corresponding SORTOUT, SORTWK, SYSUT1, and SYSERR files after the load completes successfully, regardless of the JCL disposition of these files and whether the files were used</td>
</tr>
<tr>
<td>SYSDISC YES</td>
<td>Tells LOADPLUS to delete the SYSDISC file if it does not contain any discarded records</td>
</tr>
<tr>
<td>FORMAT BMCUNLOAD</td>
<td>Specifies that the input data in the SYSREC data set is in an internal format that was generated by UNLOAD PLUS This input data is data that was unloaded by using the UNLOAD PLUS option FORMAT BMCLOAD.</td>
</tr>
</tbody>
</table>
### Command options used in JCL

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTO TABLE</td>
<td>Identifies the table to load</td>
</tr>
<tr>
<td></td>
<td>This example contains the WHEN TABLE= obid option, which is required when</td>
</tr>
<tr>
<td></td>
<td>using FORMAT BMCUNLOAD. The OBID is the identifier that UNLOAD PLUS</td>
</tr>
<tr>
<td></td>
<td>generated for this table.</td>
</tr>
</tbody>
</table>

The SYSIN DD statement in the LOADPLUS JCL specifies the name of the SYSCNTL data set that contains the LOAD command options that UNLOAD PLUS created. Passing the options from UNLOAD PLUS to LOADPLUS in this way streamlines the data migration process because it eliminates the need to specify options in your LOADPLUS JCL. This method maximizes data integrity because it assures that the LOADPLUS control cards match the input data. Note that the JCL does not include a SYSREC DD statement because the input data set is specified with the INDSN option.

In the LOADPLUS SYSPRINT, note message BMC50138I that indicates that ENFORCE CHECK CONSTRAINTS (which is set as the default by the ENFORCE installation option) is not allowed when you specify FORMAT BMCUNLOAD. LOADPLUS sets the ENFORCE option to NO in this case.

The following figure shows the UNLOAD PLUS JCL for example 19:

**Figure 76: UNLOAD PLUS JCL for example 19**

```plaintext
// JOB
//UNLOAD19 EXEC PGM=ADUUMAIN,COND=EVEN,
//           PARM='DEHJ,ADUXM19,NEW,,MSGLEVEL(1)'
//STEPLIB DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSPRINT DD SYSOUT=*
//UTPRINT DD SYSOUT=*
//SYSCNTL DD DSN=ADU.EXAMPL19.SYSCNTL,
//         DISP=(NEW,CATLG),SPACE=(CYL,(5,2)),
//         UNIT=SYSDA
//SYSIN DD *
//UNLOAD
//       OUTPUT SYSREC UNIT SYSDA
//         SPACE (5,1)
//         DSNNAME 'ADU.EXAMPL19.SYSREC'
//         CNTLCARDS 'ENUMROWS' 'REPLACE'
//         'DELETEFILES YES SYSDISC YES'
//         ORDER YES
//         FORMAT BMCLOAD
//         SELECT * INTO NAME BMC.TBEXB19
//         FROM BMC.RET_EMPLS
```

The following figure shows the LOADPLUS JCL for example 19:

**Figure 77: LOADPLUS JCL for example 19**

```plaintext
// JOB
//LOADEX19 EXEC PGM=AMUUMAIN,
//         PARM='DEHJ,ADUXM19,NEW,,MSGLEVEL(1)'
//STEPLIB DD DISP=SHR,DSN=product.libraries
```
SYSPRINT for example 19 - UNLOAD PLUS

The following figure shows the UNLOAD PLUS SYSPRINT output for example 19:

Figure 78: UNLOAD PLUS SYSPRINT for example 19
Example 19 - Unloading data to LOADPLUS

The following figure shows the LOADPLUS SYSPRINT for example 19:

Figure 79: LOADPLUS SYSPRINT for example 19

---

**SYSPRINT for example 19 - LOADPLUS**

The following figure shows the LOADPLUS SYSPRINT for example 19:

---

**Figure 79: LOADPLUS SYSPRINT for example 19**

---
Example 19 - Unloading data to LOADPLUS

Example 19 - Unloading data to LOADPLUS

UNLOAD PLUS for DB2 Reference Manual

394 UNLOAD PLUS for DB2 Reference Manual
Example 19 - Unloading data to LOADPLUS

Chapter 5  Examples of UNLOAD PLUS jobs  395
SELECT output from example 19

The following figure shows the output of a SELECT on the loaded table for example 19:

Figure 80: SELECT output from example 19

<table>
<thead>
<tr>
<th>EMPLOYEE_NAME</th>
<th>EMPLOYEE_FNAME</th>
<th>EMPLOYEE_SALARY_CODE</th>
<th>EMPLOYEE_SALARY_HRLY</th>
<th>EMPLOYEE_DEPT_NO</th>
<th>EMPLOYEE_HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>Raymond</td>
<td>H</td>
<td>9.00</td>
<td>1100</td>
<td>01/16/1993</td>
</tr>
<tr>
<td>Broderick</td>
<td>Samuel</td>
<td>S</td>
<td>44000.00</td>
<td>2400</td>
<td>08/23/1992</td>
</tr>
<tr>
<td>Baker</td>
<td>Kathy</td>
<td>H</td>
<td>4.75</td>
<td>500</td>
<td>01/16/1993</td>
</tr>
<tr>
<td>Jackson</td>
<td>William</td>
<td>S</td>
<td>76000.00</td>
<td>2400</td>
<td>09/14/1992</td>
</tr>
<tr>
<td>Jackson</td>
<td>David</td>
<td>S</td>
<td>92000.00</td>
<td>2500</td>
<td>07/02/1991</td>
</tr>
<tr>
<td>Billingsly</td>
<td>Randy</td>
<td>H</td>
<td>8.10</td>
<td>2500</td>
<td>06/17/1988</td>
</tr>
<tr>
<td>Brown</td>
<td>Ellen</td>
<td>S</td>
<td>24000.00</td>
<td>2700</td>
<td>11/02/1992</td>
</tr>
<tr>
<td>Bronson</td>
<td>Parker</td>
<td>H</td>
<td>6.25</td>
<td>3200</td>
<td>03/05/1990</td>
</tr>
<tr>
<td>Joe</td>
<td>John</td>
<td>S</td>
<td>32000.00</td>
<td>3300</td>
<td>?</td>
</tr>
<tr>
<td>Welch</td>
<td>John</td>
<td>S</td>
<td>48000.00</td>
<td>3600</td>
<td>08/26/1993</td>
</tr>
<tr>
<td>Brittain</td>
<td>Michelle</td>
<td>H</td>
<td>32000.00</td>
<td>4200</td>
<td>12/24/1994</td>
</tr>
<tr>
<td>Franklin</td>
<td>Erica</td>
<td>S</td>
<td>32000.00</td>
<td>4300</td>
<td>04/18/1996</td>
</tr>
<tr>
<td>Williamson</td>
<td>Floyd</td>
<td>H</td>
<td>5.75</td>
<td>5100</td>
<td>04/03/1992</td>
</tr>
<tr>
<td>George</td>
<td>Hugh</td>
<td>H</td>
<td>7.50</td>
<td>5100</td>
<td>05/19/1989</td>
</tr>
<tr>
<td>Slaughter</td>
<td>Johnathon</td>
<td>H</td>
<td>6.25</td>
<td>5200</td>
<td>?</td>
</tr>
<tr>
<td>Johnson</td>
<td>Robert</td>
<td>H</td>
<td>6.90</td>
<td>6300</td>
<td>07/31/1991</td>
</tr>
<tr>
<td>Peterson</td>
<td>Barbara</td>
<td>H</td>
<td>36000.00</td>
<td>8200</td>
<td>12/19/1989</td>
</tr>
<tr>
<td>Kennedy</td>
<td>Jennifer</td>
<td>S</td>
<td>45000.00</td>
<td>8200</td>
<td>03/31/1993</td>
</tr>
</tbody>
</table>

SUCCESSFUL RETRIEVAL OF 20 ROW(S).
Tuning UNLOAD PLUS jobs

This chapter provides information that enables you to improve the performance or memory use of your jobs.

Tuning for performance

UNLOAD PLUS provides several options that you can use to influence its performance.

Some options affect the performance of the entire utility; others influence processing only for specific functions of UNLOAD PLUS. This section explains the effect that these options have on the performance of UNLOAD PLUS.

Setting installation options for optimal performance in UNLOAD PLUS

In general, you should use the values that were shipped with UNLOAD PLUS for the installation options. These values usually provide you with optimal performance.

However, there are certain environments for which BMC recommends that you make adjustments to these values. The following table describes these recommendations. For a complete description of each option, see “UNLOAD PLUS installation options” on page 417.

Note
If you modify these options after installation, you must rerun the installation job for these modifications to take effect.
### Table 94: Installation option changes for performance

<table>
<thead>
<tr>
<th>Installation option</th>
<th>Recommended value</th>
<th>Description</th>
<th>Reference to additional performance information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENTDEGREE</td>
<td>ANY</td>
<td>Set this value if you must use DIRECT NO processing. Under DIRECT NO, this value allows you to take advantage of parallelism when it is available.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>DRNDELAY, DRNRETRY, and FORCE</td>
<td></td>
<td>Set these values based on how long your application can wait for a successful drain.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>HISTORY</td>
<td>NO</td>
<td>Set this value if you do not need to update the BMCHIST table.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>SMCORE</td>
<td>(0K,0K)</td>
<td>This value is the default value, but it may have been changed to match your system sort options. BMC recommends that you use the default value.</td>
<td>“Performance considerations for sort processing options in UNLOAD PLUS” on page 405</td>
</tr>
<tr>
<td>TASKMAX</td>
<td>16</td>
<td>For most jobs, set this value in conjunction with UNLDMAX to maximize the number of tasks that UNLOAD PLUS uses.</td>
<td>Not applicable</td>
</tr>
<tr>
<td></td>
<td>200%</td>
<td>If you have only a few CPUs, and those CPUs are not saturated, use this value, in conjunction with the corresponding values for SMAX and UNLDMAX to drive more tasks. Adjust this number upward as the environment allows.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>UBUFFS</td>
<td>255</td>
<td>Set this value if you are unloading multiple partitioned table spaces or multiple tables into a single SYSREC and you have enough memory to support this value.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>UNLDMAX</td>
<td>200%</td>
<td>If you have only a few CPUs, and those CPUs are not saturated, use this value, in conjunction with the corresponding values for SMAX and TASKMAX to drive more tasks. Adjust this number upward as the environment allows.</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
### Installation option

<table>
<thead>
<tr>
<th>Installation option</th>
<th>Recommended value</th>
<th>Description</th>
<th>Reference to additional performance information</th>
</tr>
</thead>
</table>
| UNLOADDN_ACTIVE     | (YES,YES)         | Activate dynamic allocation for both the primary and secondary unload data sets.  
|                     |                   | ■ Ensure that the JCL does not specify unload data sets. If you specify unload data sets in the JCL, UNLOAD PLUS does not use dynamic allocation.  
|                     |                   | ■ BMC recommends that you do not activate dynamic allocation if you are using the FILTERPART option with a single SELECT statement. | Not applicable |
| UXSTATE             | PROB              | *If you can ensure* that all exits that UNLOAD PLUS calls can run in problem state, specify this value to obtain significant savings in CPU overhead. | “Performance considerations for the UXSTATE installation option in UNLOAD PLUS” on page 407 |

### Specifying command options for optimal performance in UNLOAD PLUS

This section describes command option recommendations to obtain optimal performance when running UNLOAD PLUS.

Some of these recommendations apply only to specific environments or conditions. Review the information in these tables to determine whether the recommendations meet your needs.

The following table lists UNLOAD command options that correspond to the installation options listed in “Setting installation options for optimal performance in UNLOAD PLUS” on page 399. BMC recommends that you specify the UNLOAD command options that are listed in this table if they are not already specified in the installation options. See the table in “Setting installation options for optimal performance in UNLOAD PLUS” on page 399 for comments about these command options.
Table 95: UNLOAD PLUS command options that correspond to installation options

<table>
<thead>
<tr>
<th>Command option</th>
<th>Corresponding installation option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENTDEGREE ANY</td>
<td>CURRENTDEGREE=ANY</td>
</tr>
<tr>
<td>RETRY $n$, where $n$ is based on your environment</td>
<td>DRNRETRY= $n$, where $n$ is based on your environment</td>
</tr>
<tr>
<td>RETRY_DELAY $n$, where $n$ is based on your environment</td>
<td>DRNDELAY= $n$, where $n$ is based on your environment</td>
</tr>
<tr>
<td>ACTIVE(YES,YES)</td>
<td>UNLOADDN_ACTIVE=(YES,YES)</td>
</tr>
</tbody>
</table>

The following table describes additional recommended command options. (No installation options correspond to these command options.)

Table 96: Command option values for performance

<table>
<thead>
<tr>
<th>Command option</th>
<th>Conditions and comments</th>
<th>Reference to additional performance information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCSID</td>
<td>Do not specify this option unless your job requires it. CCSID translation can degrade performance.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>CNTLCARDS BMCLOAD 'ENUMROWS' 'PRELOAD LOAD' 'SORTNUM 32'</td>
<td>If you are running UNLOAD PLUS in conjunction with LOADPLUS and you are generating LOADPLUS control cards by specifying CNTLCARDS BMCLOAD, specify these strings.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>FILTERPART</td>
<td>Specify this option when all of the following conditions exist:</td>
<td>“Using multiple unload data sets” on page 409</td>
</tr>
<tr>
<td></td>
<td>■ You do not want to unload all partitions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ You suspect that the data is not in all partitions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ You do not know in which partition the data that you need to unload resides.</td>
<td></td>
</tr>
<tr>
<td>FIXEDVARCHAR</td>
<td>Do not specify YES for this option unless your job requires it. This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Command option</td>
<td>Conditions and comments</td>
<td>Reference to additional performance information</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Specify the fastest FORMAT option that meets your needs. The options are listed from fastest to slowest and you can find a detailed description of each output format in “UNLOAD PLUS output format” on page 64.</td>
<td>Not applicable</td>
</tr>
<tr>
<td></td>
<td>■ FORMAT BMCLOAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ FORMAT INTERNAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ FORMAT STANDARD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ FORMAT DSNTIAUL</td>
<td>This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
</tr>
<tr>
<td></td>
<td>■ FORMAT EXTERNAL</td>
<td>This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
</tr>
<tr>
<td>MAXBLKSIZE</td>
<td>Do not specify this option unless the job requires it. This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
<td>“Tuning I/O” on page 410</td>
</tr>
<tr>
<td>PART</td>
<td>Specify this option to unload data only from the partitions that you need.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>SELECT</td>
<td>Specify only the columns that you need for the job.</td>
<td>“Using multiple unload data sets” on page 409</td>
</tr>
<tr>
<td>USELRECL</td>
<td>Do not specify this option unless the job requires it. This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**Additional performance information for UNLOAD PLUS installation and command options**

The following topics provide information, in addition to the information in the previous tables, for some of the installation and command options.
This information can help you make decisions about these options that can affect the performance of your unload jobs.

**Performance considerations for buffer installation options in UNLOAD PLUS**

You can use installation options to control the amount of buffer space that the data sets use during UNLOAD PLUS processing.

For each option, UNLOAD PLUS multiplies the number that you specify by a particular value (explained in the following sections) to determine the amount of virtual storage needed for the buffer pool. UNLOAD PLUS then allocates as many buffers as will fit into that space.

UNLOAD PLUS allocates all VSAM buffers and all sequential I/O buffers above the 16-megabyte line.

**DB2 data sets**

The DB2 data sets contain the table spaces that UNLOAD PLUS is unloading. UNLOAD PLUS reads these data sets during the UNLOAD phase. While considering available virtual storage and the number of partitions that it is processing concurrently, UNLOAD PLUS assigns buffers for these data sets. Providing as much virtual storage as possible allows UNLOAD PLUS to process several partitions concurrently. The virtual storage for these buffers is acquired above the 16-megabyte line. For more information, see “Providing maximum virtual storage” on page 410.

**IBUFFS installation option**

UNLOAD PLUS uses the IBUFFS installation option to determine the amount of buffer pool storage to allocate for reading image copy data sets.

The multiplier that UNLOAD PLUS applies to the IBUFFS value varies as follows:

- *For copy data sets on tape*, UNLOAD PLUS uses a multiplier of 256 KB (to allow for block sizes greater than 32 KB). In this case, if the IBUFFS option has a value greater than 6, UNLOAD PLUS ignores that setting and uses 6.

- *For all other copy data sets*, UNLOAD PLUS uses a multiplier of 32 KB. In this case, BMC recommends a value of 25 for the IBUFFS option, which is the value that is shipped with UNLOAD PLUS.

Virtual storage for these buffers is acquired above the 16-megabyte line.
Providing as much virtual storage as possible allows processing of several partitions concurrently. For more information, see “Providing maximum virtual storage” on page 410.

**UBUFFS installation option**

UNLOAD PLUS uses the UBUFFS installation option to determine the amount of buffer pool storage to allocate for writing each SYSREC and SYSRED data set.

The multiplier that UNLOAD PLUS applies to the UBUFFS installation option is 32 KB. For UBUFFS, BMC recommends a value of 25 (the value that is shipped with UNLOAD PLUS). Depending on the device type containing the data sets, UNLOAD PLUS determines the optimal block size of the output data sets unless you use the MAXBLKSIZE command option. Virtual storage for these buffers is acquired above the 16-megabyte line.

**Performance considerations for sort processing options in UNLOAD PLUS**

This topic describes options that enable you to control the performance of sort processing in UNLOAD PLUS.

The BMC Software BMCSORT technology provides UNLOAD PLUS with more control of the sort process than external sort routines provide. This added control helps prevent memory problems during the sort process. UNLOAD PLUS allocates the amount of resources to each sort process based on the amount of work that UNLOAD PLUS determines that sort process will perform.

**Controlling memory usage (SMCORE option)**

UNLOAD PLUS provides the SMCORE installation option to give you control, when necessary, over the amount of memory that BMCSORT uses during an unload job. This option contains two parameters, total memory and below-the-line memory, which the following sections describe.

BMC strongly recommends that you use the values 0K and 0K for the SMCORE option. These values tell UNLOAD PLUS to determine the appropriate amount of memory to use for each sort based on the following criteria:

- Value that you specified for REGION in either your JCL or system exits
- Amount of memory that is available during optimization
- Number of sorts to process

Generally, you can obtain the highest sort performance for your UNLOAD PLUS job by using the values 0K and 0K for the SMCORE option.
**Total memory**: The first parameter value of the SMCORE option tells UNLOAD PLUS how much total memory above and below the 16-megabyte line you want BMCSORT to use during a single invocation. BMC strongly recommends that you specify 0K, which allows UNLOAD PLUS to determine the optimal amount of total memory to use. In addition to 0K, however, valid values are 4096K through 65536K. You can also specify this value in megabytes (either 0M or 4M through 64M).

Whether UNLOAD PLUS determines the value for total memory or you specify a value, UNLOAD PLUS multiplies this value by the number of required sort processes to get a value for the total memory that the current job requires. Depending on the workload and system environment, UNLOAD PLUS distributes this total memory among the sort processes for the job.

For example, if you specify 4096K and UNLOAD PLUS determines that it needs four sort processes for this job, UNLOAD PLUS calculates that it needs 16384 KB of total memory for the job. If the workload for each sort process is different, UNLOAD PLUS invokes BMCSORT for each sort process with varying amounts of memory. Some of these amounts will be lower or higher than the 4096 KB that you specified.

The following additional considerations apply to the first value of the SMCORE option:

- The region size that is available for your unload job in conjunction with the value that you specify for this parameter can constrain the number of sort processes that UNLOAD PLUS starts. Because the region size must include space for buffers and other required structures, the entire region size is not available for sort processing. You can avoid this constraint by using a value of 0K, allowing UNLOAD PLUS to determine the optimal amount of total memory to use.

- When you allow UNLOAD PLUS to optimize total memory, UNLOAD PLUS never uses more than the value of your region parameter and never uses more than an average of 24 MB per sort process.

**Below-the-line memory**: The second parameter value of the SMCORE option indicates how much memory BMCSORT should use below the 16-megabyte line during a single invocation. With the exception of the situation described in the following considerations, BMC recommends that you specify 0K, which allows UNLOAD PLUS to determine the optimal amount of below-the-line memory to use. In addition to 0K, valid values are 256K through 4096K. You can also specify this value in megabytes (either 0M or 1M through 4M).

The following additional considerations apply to the second value of the SMCORE option:

- If you specify a value for the SORTNUM option that is greater than 32 (indicating that you want to use more than 32 sort work files), BMC strongly recommends that you specify 384K for the second parameter of SMCORE.
BMCSORT never needs more than 384 KB of memory below the line. Specifying a value greater than 384K can limit the number of sort tasks that UNLOAD PLUS can start concurrently.

**Controlling the number of sort processes (S MAX option)**

UNLOAD PLUS determines the optimal number of sorts executing concurrently, depending on available resources. You can specify the maximum number of concurrent sorts with the S MAX installation option.

Under normal circumstances, you should allow UNLOAD PLUS to control the number of sorts that it processes concurrently. The value of the S MAX option that is shipped with the product is 16. However, if processing constrains system resources or if other problems occur, you can specify a different value for the S MAX option to limit the number of sort processes running concurrently.

You may also specify the MAXSORTS command option to reduce the number of concurrent sorts.

**Performance considerations for the UXSTATE installation option in UNLOAD PLUS**

UNLOAD PLUS invokes DB2 user exits during processing.

UNLOAD PLUS calls the MODESET SVC to perform this switch before invoking an exit, and again to reset the mode after returning from the exit.

However, invoking MODESET is an expensive CPU process. If you can ensure that all exits that UNLOAD PLUS calls can run in problem state, you can specify UXSTATE=PROB in the installation options. This specification causes UNLOAD PLUS to invoke user exits in problem state (PSW key=7), and achieves significant savings in CPU overhead.

For more information, see “DB2 user exits with UNLOAD PLUS” on page 77.

**Enabling multitasking for performance in UNLOAD PLUS**

You can take advantage of the multitasking capabilities in UNLOAD PLUS to improve performance of your unload jobs.

UNLOAD PLUS determines the most effective arrangement of tasks when the utility runs in a multi-processor environment. Although UNLOAD PLUS runs very efficiently on single-processor computers, it performs even faster on large multi-processor systems. Unloading data from DB2 tables requires several distinct tasks,
including sorting and converting data. UNLOAD PLUS can perform these tasks concurrently, reducing the elapsed time of the unload process.

If multiple tasks unload to a single output data set, as in the case of a partitioned table space or multiple table spaces, you can ensure that the contention for the single unload data set does not become detrimental to overall performance. If you are using multiple tasks, BMC recommends that you use multiple output data sets.

To improve performance and tailor multitasking activities, you can start more than one task per CPU for individual or all processing phases of UNLOAD PLUS. The installation options shown in “Setting installation options for optimal performance in UNLOAD PLUS” on page 399, when used together, can be set to maximize the number of tasks that UNLOAD PLUS uses.

**Limitations on multitasking**

The following conditions limit the ability of UNLOAD PLUS to multitask:

- When you specify ORDER YES and you are unloading a table-controlled partitioned table space with a clustering index, UNLOAD PLUS uses a single task when either of the following conditions exists:
  - The clustering key does not match, or is not a subset of, the partitioning key.
  - The clustering index is not partitioned and the clustering key is a subset of the partitioning key.

  This restriction does not apply when you specify ORDER YES BYTASK.

- When you specify ORDER BY and either of the following conditions exists, UNLOAD PLUS sets MAXSORTS to 1 and assigns all partitions to a single task:
  - The columns that you specify in the ORDER BY option are different from the partitioning index key columns.
  - The columns that you specify in the ORDER BY option are the same or a subset of the partitioning index key columns but are in a different order.

- When you specify ORDER YES or ORDER BY and you are unloading a partition-by-growth table space, UNLOAD PLUS uses a single task.

- When you specify LE COBOL, COBOL II, C, or LE C user exit routines, UNLOAD PLUS sets MAXSORTS to 1.

- When unloading to a BatchPipes file, you must set MAXSORTS or MAXCONNECT to 1.
Unloading from LOB or XML table spaces

To optimize multitasking when you are unloading LOB or XML data and DIRECT YES is in effect, unload to multiple referenced files and specify the SUBSETS option on the OUTPUT statement for the referenced files. When UNLOAD PLUS allocates multiple data sets, a separate LOB or XML table space subtask unloads to each data set. For more information, see “SUBSETS” on page 191.

Unloading from multiple table spaces

When you unload from multiple table spaces, UNLOAD PLUS first optimizes the arrangement of tasks for each partitioned table space that it can unload with multiple tasks. UNLOAD PLUS then determines the most effective arrangement of tasks for the remaining nonpartitioned table spaces, assigning one table space per task.

If you are unloading from multiple table spaces, BMC recommends that you use multiple unload data sets to reduce contention that could impact performance.

Unloading from multi-table segmented table spaces

UNLOAD PLUS processes only those segments that pertain to the tables that it is unloading. The utility optimizes processing of tables in a multi-table segmented table space. UNLOAD PLUS does not read data in tables that you did not specify in the selection criteria.

Using multiple unload data sets

When you unload a partitioned table space with a single SELECT statement, BMC recommends that you use one unload data set per partition unless the number of partitions being unloaded exceeds the maximum threshold. When you unload from multiple table spaces, use one unload data set per SELECT statement. Using multiple unload data sets in these cases enables UNLOAD PLUS to achieve maximum throughput and minimum elapsed time. To obtain the best results, put each unload data set on a different DASD volume. This precaution produces the least amount of interference between multiple tasks and ensures the smallest elapsed and CPU time. For more information, see “Using JCL to specify multiple unload data sets” on page 289.

In general, UNLOAD PLUS opens all unload data sets at the same time. This processing requires sufficient region size and tape devices to execute.

When you specify the FILTERPART YES option, however, the only data sets that UNLOAD PLUS opens are data sets for the unloaded partitions. UNLOAD PLUS dynamically eliminates processing of partitions that do not meet the conditions of the WHERE clause.
Additional performance tuning recommendations for UNLOAD PLUS

The following topics describe additional steps that you can take to improve the performance of your unload jobs.

**Tuning I/O**

The single most important factor that affects performance in UNLOAD PLUS is I/O processing. During typical unload processing, UNLOAD PLUS reads and writes large amounts of data.

To maximize I/O performance, UNLOAD PLUS handles all its own buffering and performs I/O operations at the lowest level possible. This ability allows UNLOAD PLUS to read or write several blocks of data with each I/O operation. Furthermore, UNLOAD PLUS can prefetch subsequent data. UNLOAD PLUS always calculates the optimum block size for output data sets unless you use the MAXBLKSIZE option.

To avoid I/O queueing, allocate UNLOAD PLUS output data sets on separate channels and drives. If sufficient channels are not available, use separate drives and control units.

Because UNLOAD PLUS I/O processing is always sequential, DASD caching provides no benefit. In fact, try to avoid DASD caching because the overhead might slightly increase I/O processing time.

**Providing maximum virtual storage**

Each task requires virtual storage for processing. UNLOAD PLUS balances the multi-processing performed with the amount of virtual storage that is available. UNLOAD PLUS and its components use virtual storage, primarily for I/O buffers.

Because UNLOAD PLUS uses as much virtual storage as is required, you should always specify as much virtual storage as possible. BMC recommends that you specify REGION=0M in the JOB or EXEC statement of the execution JCL to tell the system to allocate all available virtual storage to the UNLOAD job. If your data center does not permit you to specify REGION=0M, or if the installation exit IEFUSI overrides your specification of the region, specify the amount that allows the most virtual storage above and below the 16-megabyte line.

Using a region size that is less than optimal risks the following potential issues:

- Running less efficiently, which could result in additional CPU and elapsed time
- Encountering memory failures or jobs that fail when new versions implement changes that require additional memory
Additionally, if you specify a value for REGION other than 0M, ensure that you have an appropriate value set for the MEMLIMIT parameter, either as your site’s default SMF option or on your JOB statement or EXEC statement. BMC makes the following recommendations for the MEMLIMIT option:

- Specify NOLIMIT to allow unlimited above-the-bar memory.
- If you are unable to specify NOLIMIT, specify at least 4 GB; if you are unloading LOB or XML data, specify at least 32 GB.

**Performance tuning for specific scenarios in UNLOAD PLUS**

The following table describes the steps that you can take to tune specific types of unload jobs to improve performance:

<table>
<thead>
<tr>
<th>Scenario type</th>
<th>Scenario</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unloading data</td>
<td>Data migration between DB2 objects with identical or almost-identical definitions and both LOADPLUS and UNLOAD PLUS are installed</td>
<td>Use the FORMAT BMCLOAD feature in conjunction with the FORMAT BMCUNLOAD feature in LOADPLUS. When you use this feature, BMC recommends that you generate LOADPLUS control cards by specifying the following UNLOAD command option: CNTLCARDS BMCLOAD 'ENUMROWS' 'PRELOAD LOAD' 'SORTNUM 32'</td>
</tr>
<tr>
<td>Unloading LOB or XML data</td>
<td>(DIRECT YES only)</td>
<td>Unload to multiple referenced files and specify the SUBSETS option on the OUTPUT statement for those files.</td>
</tr>
<tr>
<td>Unloading data to other platforms</td>
<td></td>
<td>Specify FORMAT CSV or FORMAT XML to unload data to other platforms.</td>
</tr>
<tr>
<td>High-transaction environments</td>
<td></td>
<td>Consider specifying NONE for either the DRNWAIT installation option or the DRAIN_WAIT command option.</td>
</tr>
<tr>
<td>Sorting data</td>
<td>Unloading data using UNLOAD PLUS and loading data using LOADPLUS and you need sorted data</td>
<td>Perform the sort in UNLOAD PLUS rather than LOADPLUS.</td>
</tr>
<tr>
<td>SELECT statements</td>
<td>Single SELECT statement</td>
<td>Specify one SYSREC per partition.</td>
</tr>
<tr>
<td></td>
<td>Multiple SELECT statements</td>
<td>Specify one SYSREC per SELECT statement.</td>
</tr>
<tr>
<td></td>
<td>SELECT statement with a predicate on the clustering key</td>
<td>Use the FILTERPART option to reduce the number of partitions that UNLOAD PLUS has to scan.</td>
</tr>
</tbody>
</table>
### Interpreting performance-related messages from UNLOAD PLUS

UNLOAD PLUS issues performance-related messages if you specify MSGLEVEL(1) in the execution JCL. Information in these messages can help you monitor performance of UNLOAD PLUS and fine-tune future runs.

**BMC50471I environmentInformation**

This message displays current values for each option in the installation options module. Use this information to verify that UNLOAD PLUS is using the option value that you want to use.

**BMC50474I BELOW 16M = nK, ABOVE 16M = nK, CPUS = n**

This message provides information about virtual storage and CPU usage. It displays the following information:

- Amount of available virtual storage that is below the 16-megabyte line
- Amount of available virtual storage that is above the 16-megabyte line
- Number of physical CPUs available in the processor

Use this information to ensure that adequate virtual storage is available for UNLOAD PLUS. For more information, see “Providing maximum virtual storage” on page 410 and message BMC50475I.

**BMC50475I MAX TASKS = t, PARTITIONS PER TASK = p, SORTWKS PER TASK = s**

This message provides the results of task optimization during sort processing. This message contains the following information:

- Maximum number of concurrent tasks
- Number of partitions that UNLOAD PLUS can process per task
- Number of sort work files that UNLOAD PLUS assigned to each task

The maximum number of tasks depends on the availability of system resources such as virtual storage, the number of physical CPUs, and sort work file space. Increasing the amount of virtual storage or allocating more sort work files can help UNLOAD PLUS run more tasks concurrently.

This message can appear multiple times when you unload from multiple table spaces. UNLOAD PLUS first performs task optimization for each partitioned table space that the utility can unload using multiple tasks. UNLOAD PLUS subsequently determines the optimal arrangement of tasks for the remaining table spaces, assigning one table space per task.

**Task optimization for partitioned table spaces**

Using the size of the partition and available space for the sort work file, UNLOAD PLUS determines the number of partitions to process per task that will allow the fastest sort. The actual number of partitions that UNLOAD PLUS processes per task might be fewer than the maximum that the message states.

**Task optimization for remaining table spaces**

Using the number of table spaces, the size of the largest table space, and available sort work file space, UNLOAD PLUS determines the number of tasks to use. The number of partitions processed per task is the number of partitions that UNLOAD PLUS processes from the table space with the most partitions.

```
BMC50476I DDNAME = ddname, I/Os = n, I/O WAITS = w, RDB LOCK WAITS = r
```

This message provides performance information about I/O operations to sequential data sets. The message contains the following information:

- ddname that UNLOAD PLUS associates with the I/O operations
- Number of I/Os (blocks) that UNLOAD PLUS writes to or reads from the data set
- Number of waits that UNLOAD PLUS issues for I/O completion
- Number of waits for serialization of the data set

A wait count that is greater than 10 percent of the block count can indicate degraded performance, which you can remedy with more buffers for the associated data set. See “Performance considerations for buffer installation options in UNLOAD PLUS” on page 404 for more information. A high number of serialization waits might indicate the need to decrease the number of concurrent tasks or to use multiple
output data sets, one for each partition. If you are already using multiple output data sets, ensure that you are using different DASD devices for each data set.

**BMC50477I** taskNumber: PARTITION = partitionNumber, ROWS/KEYS = n, I/O WAITS = w, DDNAME = ddname

This message provides performance information about I/O operations to VSAM data sets. This message contains the following information:

- Processing task number
- Table space physical partition number
- Number of rows or keys in the partition
- Number of waits that UNLOAD PLUS issued for I/O completion
- ddname that UNLOAD PLUS associated with the I/O operations

Use the ddname to find the actual number of I/Os issued to the associated data set. A wait count that is greater than 20 percent of the actual I/Os can indicate degraded performance. Try to allow more virtual storage if UNLOAD PLUS issues message BMC50397I, and provide more sort work file space if UNLOAD PLUS issues message BMC50398I.

**BMC50478I** taskNumber: RDB LOCK WAITs = r

This message provides the total number of task waits for serial access to sequential data sets.

A wait count that is greater than 10 percent of the total number of rows that the UNLOAD PLUS task unloaded can indicate task contentions and degraded performance. If this problem occurs, use the MAXSORTS command option to reduce the number of concurrent tasks (that message BMC50475I specifies), or use one output data set per partition to eliminate serialization and provide maximum performance.

## Tuning to improve memory use in UNLOAD PLUS

To help you determine how to improve memory usage, consider the information in the following topics.
Tuning sort processing

Specifying a value greater than 0 for the SORTNUM option tells UNLOAD PLUS that BMCSORT will control sort work allocations, which should eliminate sort work constraints.

The amount of multitasking that sort processing performs depends on the number of CPUs, the SMAX or MAXSORTS option value, the number of SORTWK data sets, and available memory. You might be able to improve memory usage by adjusting the SMAX or MAXSORTS option value or the number of SORTWK data sets.

Tuning when unloading objects with a large number of partitions

Unloading objects with a large number of partitions increase the potential for encountering such issues as performance problems and memory restrictions.

Use the following information to help you avoid these issues:

- Ensure that you have specified a region size that allows the system to allocate as much virtual storage as possible to the UNLOAD PLUS job. BMC recommends that you specify REGION=0M in the JOB or EXEC statement of your execution JCL.

- If you have specified REGION=0M and your unload job fails with a constrained resources error, consider unloading fewer partitions in a single job.

- When specifying the &PART variable, you should limit the number of dynamically allocated SYSREC data sets to no more than 300 to 500. This limitation is due to the amount of below-the-line storage that is available to the job step.

- If you are unloading a large number of partitions, consider lowering the value of the VOLCNT option from the default of 25 to avoid data set allocation limitations.
UNLOAD PLUS installation options

This chapter describes the options that you can specify during installation.

Overview of UNLOAD PLUS installation options

Options set during installation establish default values that UNLOAD PLUS uses during your unload job.

When you use the BMC Installation System to install the UNLOAD PLUS for DB2 product, the configuration process generates a customized installation data set. This data set contains customized jobs that install UNLOAD PLUS into your specific DB2 environment. The following jobs establish the default processing option values that UNLOAD PLUS uses:

- $730DOP2 establishes the defaults for the basic UNLOAD PLUS processing options and the dynamic allocation options. (The job assembles the $ADUOUTP specifications within the standard options module.)

- $532SOPT contains options for the BMC Software BMCSORT technology.

These jobs assemble options macros. The macros contain the UNLOAD PLUS processing options and the values for those options that are shipped with UNLOAD PLUS and BMCSORT. When the Installation System-generated configuration job is submitted, it links the ADU$OPTS installation options module in the APF-authorized library that is designated by your site.

You can customize the installation of UNLOAD PLUS by changing the values for the UNLOAD PLUS installation options. However, if you change any of the values in $730DOP2 or $532SOPT after UNLOAD PLUS is installed, you must rerun the jobs before these changes take effect.

You can also create additional options modules that allow you to use different values for these options for different executions of UNLOAD PLUS. For example, you might use the default installation options module for most jobs but create
another options module with customized values for certain options for special situations. For information about specifying an options module at execution, see “Building and executing UNLOAD PLUS jobs” on page 271. For more information about customizing your installation of UNLOAD PLUS, see the BMC Products and Solutions for DB2 Customization Guide.

To find a description of each options macro, use the following table:

**Table 98: Customized jobs and installation macros**

<table>
<thead>
<tr>
<th>Job</th>
<th>Macro name</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$730DOP2</td>
<td>$ADUOPTS</td>
<td>Basic options</td>
<td>“Basic UNLOAD PLUS installation options” on page 418</td>
</tr>
<tr>
<td></td>
<td>$ADUOUTP</td>
<td>Options for dynamic allocation</td>
<td>“Dynamic allocation installation options” on page 447</td>
</tr>
<tr>
<td>$532SOPT</td>
<td>$AUPSMAC</td>
<td>Options for BMCSORT</td>
<td>“DYNALOC installation option” on page 463</td>
</tr>
</tbody>
</table>

### Basic UNLOAD PLUS installation options

This topic provides a quick reference to the options contained in the $ADUOPTS macro in $730DOP2.

For each option, the table provides the value that ships with this version of UNLOAD PLUS (or *No value* if the option was shipped without a value), a brief description, and a reference to more details. If an option ships with no value, the table shows a recommended value or example value.

**Table 99: Basic UNLOAD PLUS installation options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZE</td>
<td>(DB2STATS,NOLIMIT)</td>
<td>Whether to use statistics to estimate the number of rows</td>
<td>“ANALYZE=(DB2STATS,NOLIMIT)” on page 422</td>
</tr>
<tr>
<td>CENTURY</td>
<td>(1950,2049)</td>
<td>Century specification for two-digit years</td>
<td>“CENTURY=(1950,2049)” on page 423</td>
</tr>
<tr>
<td>CHANGE_CONSISTENT</td>
<td>NO</td>
<td>SHRLEVEL CHANGE CONSISTENT default</td>
<td>“CHANGE_CONSISTENT=NO” on page 423</td>
</tr>
<tr>
<td>CHANGE QUIESCE</td>
<td>NO</td>
<td>QUIESCE default</td>
<td>“CHANGE QUIESCE=NO” on page 424</td>
</tr>
<tr>
<td>Option</td>
<td>Shipped value</td>
<td>Brief description</td>
<td>Reference</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>CMAX</td>
<td>16</td>
<td>Maximum number of concurrent tasks when using dynamic SQL</td>
<td>“CMAX=16” on page 425</td>
</tr>
<tr>
<td>CMRATIO</td>
<td>50</td>
<td>Compression ratio</td>
<td>“CMRATIO=50” on page 425</td>
</tr>
<tr>
<td>CONSTRULES</td>
<td>BMC</td>
<td>How to handle character constants in SELECT lists</td>
<td>“CONSTRULES=BMC” on page 425</td>
</tr>
<tr>
<td>CURRENTDEGREE</td>
<td>NONE</td>
<td>Whether to issue SET CURRENT DEGREE before dynamically executed SELECT statements</td>
<td>“CURRENTDEGREE=NONE” on page 426</td>
</tr>
<tr>
<td>DELFILES</td>
<td>YES</td>
<td>Whether to delete work files on failure</td>
<td>“DELFILES=YES” on page 426</td>
</tr>
<tr>
<td>DRNDELAY</td>
<td>1</td>
<td>Minimum number of seconds to wait before the next attempt to obtain a drain</td>
<td>“DRNDELAY=1” on page 427</td>
</tr>
<tr>
<td>DRNRETRY</td>
<td>255</td>
<td>Maximum number of times to attempt to obtain a drain before terminating</td>
<td>“DRNRETRY=255” on page 427</td>
</tr>
<tr>
<td>DRNWAIT</td>
<td>NONE</td>
<td>Timeout value for object drains</td>
<td>“DRNWAIT=NONE” on page 428</td>
</tr>
<tr>
<td>DSPLOCKS</td>
<td>DRNFAIL</td>
<td>How to display locks if the drain times out</td>
<td>“DSPLOCKS=DRNFAIL” on page 429</td>
</tr>
<tr>
<td>EXCLUDUMP</td>
<td>(X37,X22,X06)</td>
<td>System codes to ignore during dump processing</td>
<td>“EXCLUDUMP=(X37,X22, X06)” on page 429</td>
</tr>
<tr>
<td>FILEREFDN</td>
<td>SYSREF</td>
<td>Default output descriptor for referenced output files</td>
<td>“FILEREFDN=SYSREF” on page 429</td>
</tr>
<tr>
<td>FILL</td>
<td>NO</td>
<td>Whether to fill numeric external data types with zeros</td>
<td>“FILL=NO” on page 430</td>
</tr>
<tr>
<td>FORCE</td>
<td>NONE</td>
<td>Whether to cancel DB2 threads that might prevent a drain process from completing</td>
<td>“FORCE=NONE” on page 430</td>
</tr>
<tr>
<td>FORCE_AT</td>
<td>(START,3)</td>
<td>When to cancel DB2 threads that might prevent a drain process from completing</td>
<td>“FORCE_AT=(START,3)” on page 431</td>
</tr>
<tr>
<td>Option</td>
<td>Shipped value</td>
<td>Brief description</td>
<td>Reference</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>FORCE_RPT</td>
<td>NO</td>
<td>Whether to display a report of the canceled threads</td>
<td>“FORCE_RPT=NO” on page 432</td>
</tr>
<tr>
<td>HISTORY</td>
<td>YES</td>
<td>Whether to update the BMCHIST table for each table space</td>
<td>“HISTORY=YES” on page 432</td>
</tr>
<tr>
<td>IBUFFS</td>
<td>25</td>
<td>Multiplier that controls the amount of buffer pool storage for image copy input data sets</td>
<td>“IBUFFS=25” on page 432</td>
</tr>
<tr>
<td>INLINE</td>
<td>NO</td>
<td>Whether to unload from inline image copies</td>
<td>“INLINE=NO” on page 433</td>
</tr>
<tr>
<td>KEYDSNAM</td>
<td>No value</td>
<td>Name of the key data set to use with encrypted copies</td>
<td>“KEYDSNAM=” on page 433</td>
</tr>
<tr>
<td></td>
<td>Example value: ACP.EKEYDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOADDECP</td>
<td>NO</td>
<td>Whether to use a DSNHDECP other than the local subsystem default</td>
<td>“LOADDECP=NO” on page 433</td>
</tr>
<tr>
<td>LOCKROW</td>
<td>YES</td>
<td>Serialization method for BMCSYNC and BMCUTIL</td>
<td>“LOCKROW=YES” on page 434</td>
</tr>
<tr>
<td>MAXP</td>
<td>5</td>
<td>Maximum number of partitions to unload concurrently</td>
<td>“MAXP=5” on page 434</td>
</tr>
<tr>
<td>MSGLEVEL</td>
<td>1</td>
<td>Default for the message-level execution parameter</td>
<td>“MSGLEVEL=1” on page 434</td>
</tr>
<tr>
<td>NULLCHAR</td>
<td>?</td>
<td>Default character for the null indicator</td>
<td>“NULLCHAR=?” on page 435</td>
</tr>
<tr>
<td>NULLTYPE</td>
<td>T1</td>
<td>Whether null indicator is leading or trailing and 1 byte or 2 bytes</td>
<td>“NULLTYPE=T1” on page 435</td>
</tr>
<tr>
<td>OPNDB2ID</td>
<td>YES</td>
<td>Whether to use the user’s RACF ID instead of the DB2 RACF ID</td>
<td>“OPNDB2ID=YES” on page 436</td>
</tr>
<tr>
<td>PLAN</td>
<td>ADU1120</td>
<td>Product plan name</td>
<td>“PLAN=ADU1120” on page 436</td>
</tr>
<tr>
<td>RECFM</td>
<td>AUTO</td>
<td>Record format for unload data sets</td>
<td>“RECFM=AUTO” on page 436</td>
</tr>
<tr>
<td>ROWSETSZ</td>
<td>100</td>
<td>Size of rowset for a single FETCH request</td>
<td>“ROWSETSZ=100” on page 437</td>
</tr>
</tbody>
</table>

Basic UNLOAD PLUS installation options

420 UNLOAD PLUS for DB2 Reference Manual
<table>
<thead>
<tr>
<th>Option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDUMP</td>
<td>(ALLPSA,CSA,RGN,SQA,LSQA,SUM,TRT,IO)</td>
<td>System dump parameters</td>
<td>“SDUMP=(ALLPSA,CSA,RGN,SQA,LSQA,SUM,TRT,IO)” on page 437</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>REFERENCE</td>
<td>SHRLEVEL option default</td>
<td>“SHRLEVEL=REFERENCE” on page 438</td>
</tr>
<tr>
<td>SMAX</td>
<td>16</td>
<td>Maximum number of sort processes to invoke concurrently</td>
<td>“SMAX=16” on page 438</td>
</tr>
<tr>
<td>SMCORE</td>
<td>(0K,0K)</td>
<td>Maximum amount of sort memory</td>
<td>“SMCORE=(0K,0K)” on page 439</td>
</tr>
<tr>
<td>SORTNUM</td>
<td>32</td>
<td>Number of sort work files per task for BMCSORT to allocate dynamically</td>
<td>“SORTNUM=32” on page 440</td>
</tr>
<tr>
<td>SQLDELAY</td>
<td>3</td>
<td>Number of seconds between retry attempts after SQLCODE -911</td>
<td>“SQLDELAY=3” on page 440</td>
</tr>
<tr>
<td>SQLRETRY</td>
<td>100</td>
<td>Number of retry attempts after SQLCODE -911</td>
<td>“SQLRETRY=100” on page 441</td>
</tr>
<tr>
<td>TAPEDISP</td>
<td>DELETE</td>
<td>Tape file disposition</td>
<td>“TAPEDISP=DELETE” on page 441</td>
</tr>
<tr>
<td>TAPES</td>
<td>NONE</td>
<td>List of default tape units</td>
<td>“TAPES=NONE” on page 441</td>
</tr>
<tr>
<td>TASKMAX</td>
<td>200%</td>
<td>Global setting for the maximum number of concurrent tasks to start</td>
<td>“TASKMAX=200%” on page 441</td>
</tr>
<tr>
<td>UBUFFS</td>
<td>25</td>
<td>Multiplier that controls the amount of buffer pool storage for unload data sets</td>
<td>“UBUFFS=25” on page 441</td>
</tr>
<tr>
<td>UNLDMAX</td>
<td>*</td>
<td>Maximum number of tasks to start per CPU during the UNLOAD phase</td>
<td>“UNLDMAX=*” on page 442</td>
</tr>
<tr>
<td>UNLOADDN_ACTIVE</td>
<td>(YES,NO)</td>
<td>Whether to activate dynamic allocation for unload data sets</td>
<td>“UNLOADDN_ACTIVE=(YES,NO)” on page 442</td>
</tr>
<tr>
<td>UNLOADDN</td>
<td>(SYSREC, SYSRED)</td>
<td>Default ddnames or output descriptors for unload data sets</td>
<td>“UNLOADDN=(SYSREC, SYSRED)” on page 442</td>
</tr>
<tr>
<td>USELRECL</td>
<td>NO</td>
<td>Whether to use the LCRECL value in the JCL</td>
<td>“USELRECL=NO” on page 443</td>
</tr>
</tbody>
</table>
### Descriptions of basic UNLOAD PLUS installation options

This topic describes each of the options.

For more information about setting the values of some of these options at runtime, see “Syntax of the UNLOAD command” on page 79 For more information about performance implications of these options, see “Tuning UNLOAD PLUS jobs” on page 399.

**ANALYZE=(DB2STATS, NOLIMIT)**

The ANALYZE option specifies whether UNLOAD PLUS uses statistics to estimate the number of rows for a specific table or partition to calculate space requirements for data sets during dynamic allocation. This option can direct UNLOAD PLUS to use a specific type and age for the statistics that it uses during the analysis.

Specify the first operand if you want UNLOAD PLUS to estimate the number of rows for a specific table and page set by using one of the following values:

- DB2STATS, to obtain information from the DB2 catalog statistics
- HURBA (high-used RBA) from the DB2 table space
  
  HURBA disables the statistics and uses the sizing information from the VSAM table space definitions. For details about the HURBA formula, see “ANALYZE” on page 123.

To specify the age of the statistics, you can specify one of the following values for the second operand:
- NOLIMIT, which tells UNLOAD PLUS to use the catalog information regardless of the age of the statistics

- A specific limit in number of days (an integer value)

**Considerations**

The following considerations apply to the ANALYZE option:

- If you are using DIRECT NO or INFILE processing, UNLOAD PLUS bypasses ANALYZE processing.

- If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

**Overriding this option**

You can override the value for this option by using the ANALYZE command option (“ANALYZE” on page 123).

**CENTURY=(1950,2049)**

The CENTURY option specifies the 100-year range that determines the century for the DATE and TIMESTAMMP external formats that contain two-digit year values. The value for this option is two four-digit years in the format (ccyy,ccyy). The first four-digit year value must be less than the second four-digit year. You must specify both values, and these values must span 100 years.

Any two-digit year between the first yy specification and 99 uses the first cc value as a prefix to create a four-digit year. Any two-digit year between 00 and the second yy specification uses the second cc value as a prefix to create a four-digit year.

For example, if you specify CENTURY(1950,2049), UNLOAD PLUS places 19 in front of each two-digit year with a value 50 through 99, and places 20 in front of each two-digit year with a value 00 through 49. The date 99/12/31 becomes 1999/12/31 and 00/12/31 becomes 2000/12/31.

**Overriding this option**

You can override the value for this option by using the CENTURY command option (see “CENTURY” on page 175) or the CENTURY field specification option (see “DATE” on page 241 or “TIMESTAMP” on page 242).

**CHANGE_CONSISTENT=NO**

The CHANGE_CONSISTENT option specifies whether UNLOAD PLUS should attempt to maintain consistency with a point-in-time image of the data when using
SHRLEVEL CHANGE. UNLOAD PLUS ignores the CHANGE_CONSISTENT option if you specify the SHRLEVEL REFERENCE command option or SHRLEVEL=REFERENCE installation option.

To maintain consistency with a point-in-time image of the data while enabling read/write access to the data, specify YES. UNLOAD PLUS provides this capability by using XBM or SUF to produce a snapshot image of the data in a table space. UNLOAD PLUS briefly restricts access to the object when externalizing pages. For more information, see “Running SHRLEVEL CHANGE CONSISTENT YES” on page 53.

Specify NO if you do not need to maintain consistency with a point-in-time image of the data when using SHRLEVEL CHANGE. When you specify NO, UNLOAD PLUS does not restrict access to the object except for the brief time required to externalize pages if you specify CHANGE_QUIESCE=YES. For more information about specifying NO, see the CHANGE_QUIESCE option.

**Overriding this option**

You can override the value for this option by using the CONSISTENT option of the SHRLEVEL CHANGE command option. See “SHRLEVEL” on page 103 for more information.

**CHANGE_QUIESCE=NO**

The CHANGE_QUIESCE option specifies whether UNLOAD PLUS should externalize pages for table spaces that are being unloaded using SHRLEVEL CHANGE CONSISTENT NO. To externalize these pages for table spaces, specify CHANGE_QUIESCE=YES. Specify CHANGE_QUIESCE=NO if you do not want to restrict access to the objects to externalize pages from the DB2 buffer pool.

**WARNING**

If you specify QUIESCE NO, UNLOAD PLUS does not restrict access to the objects, nor does it externalize pages from the DB2 buffer pool. These actions might result in the following consequences:

- UNLOAD PLUS might not process any updated pages in the buffer pool.
- If you run UNLOAD PLUS on a DB2 object immediately following the creation of the data set for that object, any data involved might exist only in DB2 buffer pools, causing UNLOAD PLUS to terminate.
- If UNLOAD PLUS encounters any pages that have uncommitted data, it unloads the uncommitted data.

UNLOAD PLUS ignores this option when SHRLEVEL REFERENCE or SHRLEVEL CHANGE CONSISTENT YES is in effect.
**Overriding this option**

You can override the value for this option by using the CONSISTENT NO QUIESCE option of the SHRLEVEL CHANGE command option. See “SHRLEVEL” on page 103 for more information.

**CMAX=16**

The CMAX option specifies the maximum number of tasks that UNLOAD PLUS can execute concurrently when you specify DB2 dynamic SQL. A value of 16 for this option allows UNLOAD PLUS to run in optimized mode.

**Overriding this option**

You can override the value for this option by using the MAXCONNECT command option (“MAXCONNECT” on page 148).

**CMRATIO=50**

The CMRATIO option specifies a percentage value that UNLOAD PLUS uses as a compression ratio to calculate the estimated number of rows for a table space when any of the following conditions exists:

- When ANALYZE HURBA is specified
- As a default because there are no DB2 statistics
- When the statistics are out of date

The value (1 through 100) that you specify for this option represents the percentage value to use. For example, if a page normally holds 10 rows in an uncompressed state, specifying a compression ratio of 50 indicates that these 10 rows occupy only 50% of the page so that the number of rows that fit in the page is now 20.

**Overriding this option**

You can override the value for this option or specify it at runtime by using the CMRATIO keyword of the ANALYZE command option (“ANALYZE” on page 123).

**CONSTRULES=BMC**

The CONSTRULES option specifies the way that UNLOAD PLUS handles character constants in the SELECT list:

- Specify BMC if UNLOAD PLUS should treat character constants in the SELECT list according to the value of the DIRECT keyword. If you specify DIRECT YES, UNLOAD PLUS treats character constants as a character data type of fixed length.
If you specify DIRECT NO, UNLOAD PLUS treats character constants as a character data type of variable length.

- Specify STANDARD for this option if UNLOAD PLUS should treat all character constants in the SELECT list as a character data type of variable length for both DIRECT YES and DIRECT NO. UNLOAD PLUS always uses STANDARD when you specify FORMAT DSNTIAUL.

**CURRENTDEGREE=NONE**

The CURRENTDEGREE option specifies that UNLOAD PLUS can issue the SET CURRENT DEGREE command before executing dynamically executed SELECT statements. You can specify CURRENTDEGREE ANY or CURRENTDEGREE 1 to issue the command, or specify NONE if you do not want UNLOAD PLUS to issue it. With NONE, DB2 uses the system default value to unload the data.

UNLOAD PLUS ignores the CURRENTDEGREE option if DIRECT YES is in effect.

*Overriding this option*

You can override the value for this option by using the CURRENTDEGREE command option (“CURRENTDEGREE” on page 102).

**DELFILES=YES**

The DELFILES option tells UNLOAD PLUS whether to delete the SYSREC, SYSRED, SORTWK, and referenced files when the unload is unsuccessful. (For more information, see “ON FAILURE” on page 171.)

You can specify one of the following options:

- A value of YES tells UNLOAD PLUS to delete the following data sets when the unload job fails to complete successfully:
  - All data sets whose ddnames match the SYSREC, SYSRED, and SORTWK ddnames or prefixes
  - All referenced file data sets
- A value of NO tells UNLOAD PLUS not to delete any of these files.

*Additional considerations*

Note the following considerations when you specify YES:

- UNLOAD PLUS does not delete files when the job terminates with an x22 abend.
UNLOAD PLUS does not delete the following files:

- If DIRECT NO is in effect, any referenced files
- Any HFS referenced files
- Any files that are defined with an abnormal allocation disposition other than DELETE
- Any files that are defined with an expiration date or retention period that prevents the delete from being honored

To specify the final disposition of tape files when you specify DELFILES=YES, see the TAPEDISP installation option.

**Running in a worklist environment**

If you are running the unload job in a worklist environment, UNLOAD PLUS functions differently. In this environment, UNLOAD PLUS ignores the value you set for the DELFILES installation option and, by default, does not delete the files. UNLOAD PLUS functions this way so that the files exist for subsequent executions in the worklist. If you specify DELETFILES YES on the UNLOAD command in the worklist, UNLOAD PLUS deletes only the dynamically allocated SYSREC, SYSRED, or referenced file data sets.

**Overriding this option**

You can override the value for this option by using the DELETFILES command option (“DELETFILES” on page 173).

**DRNDELAY=1**

After a drain times out, the DRNDELAY option specifies the minimum number of seconds that UNLOAD PLUS waits before it tries again to obtain the drain. The number of seconds can range from 1 through 1800.

**Overriding this option**

You can override the value for this option by using the RETRY_DELAY command option (“RETRY_DELAY” on page 178).

**DRNRETRY=255**

The DRNRETRY option specifies the maximum number of times that UNLOAD PLUS attempts to obtain a drain before it terminates the job. The number of retry attempts can range from 0 through 255.

**Overriding this option**
You can override the value for this option by using the RETRY command option ("RETRY" on page 177).

**DRNWAIT=NONE**

The DRNWAIT option specifies the drain timeout value to use. Specify one of the following values:

- **NONE**, which tells the drain request that UNLOAD PLUS issues to time out immediately if the drain cannot acquire the lock
  
  NONE prevents any application transactions from being queued during the drain process. BMC recommends that you specify NONE in high-transaction environments.

- **UTIL**, which tells UNLOAD PLUS to use the standard DB2 utility timeout value defined in DSNZPARMS for your site (IRLMRWT multiplied by UTIMOUT)
  
  The wait time applies to each object involved in the unload process.

- **SQL**, which tells UNLOAD PLUS to use the standard SQL timeout value (IRLMRWT) as the drain timeout value
  
  The wait time applies to each object involved in the unload process.

- Any integer value from 0 through 1800
  
  — 0 is equivalent to the value UTIL.

  — 1 through 1800 specifies the number of seconds to wait to obtain the drain for each drain retry before timing out.

If UNLOAD PLUS cannot drain all of the objects within the time period specified by DRNWAIT, UNLOAD PLUS completes the following process:

1. Releases the drains that it has obtained so far

2. Waits the length of time that you specify in the DRNDELAY installation option (or RETRY_DELAY command option)

3. Tries again to drain the objects for the number of times that you specify in the DRNRETRY installation option (or RETRY command option)

**Note**

The FORCE and FORCE_AT options tell UNLOAD PLUS whether to cancel DB2 threads that might prevent a successful drain and, if so, when to cancel them. The value that you specify for the FORCE and FORCE_AT options might affect the drain process described here.

**Overriding this option**
You can override the value for this option by using the DRAIN_WAIT command option ("DRAIN_WAIT" on page 175).

**DSPLOCKS=DRNFAIL**

The DSPLOCKS option tells UNLOAD PLUS what action to take regarding displaying claims and locks if a drain attempt times out. The following values are valid for this option:

- **DRNFAIL** tells UNLOAD PLUS to display the claims and locks once, after the final attempt to obtain the drain times out.
- **NONE** tells UNLOAD PLUS not to display any claims or locks.
- **RETRY** tells UNLOAD PLUS to display the claims and locks after each drain timeout.

*Overriding this option*

You can override the value for this option by using the DSPLOCKS command option ("DSPLOCKS" on page 178).

**EXCLDUMP=(X37,X22,X06)**

The EXCLDUMP option allows you to limit the conditions under which UNLOAD PLUS generates a system dump when the SDUMP installation option contains values other than NO. The EXCLDUMP option tells UNLOAD PLUS to exclude the listed abend codes when generating the dump.

Specify a three-digit abend code, such as 806 or 222, or prefix an X to a two-digit abend code to exclude all abend types that end in those two digits. For example, if you specify X37, UNLOAD PLUS will not generate a system dump for all abend codes that end in 37 (such as B37 or E37). You can specify up to ten abend codes with the EXCLDUMP option.

Specify EXCLDUMP=0 (without parentheses) if you want all abend codes to be candidates for a system dump.

**FILEREFDN=SYSREF**

The FILEREFDN option specifies the default output descriptor for any referenced files that UNLOAD PLUS will allocate or use.

BMC recommends that you use the default value for this option, SYSREF. The name that you specify must be different than any output descriptors or ddnames specified for the UNLOADDNN installation or command option.
**FILL=NO**

The FILL option specifies whether UNLOAD PLUS externalizes all leading zeros when it converts numeric columns to their external representations. You can specify one of the following options:

- Specify NO if you want to remove all leading zeros when converting numeric columns to their external representations.

- Specify YES to externalize all leading zeros when converting numeric columns to their external representations. As the sign of the number, the first character in the external representation of the number is either a minus sign (‘-’) or is blank. You can specify this option to ensure that the data format includes leading zeros so that applications such as COBOL can edit and check unloaded data properly.

**Overriding this option**

You can override the value for this option by using the FILL command option or by using the FILL option on the individual field specifications.

**FORCE=NONE**

The FORCE option specifies whether to cancel DB2 threads that might prevent a drain process from completing. Specify one of the following options:

- NONE tells UNLOAD PLUS not to cancel DB2 threads.

- ALL tells UNLOAD PLUS to cancel both read and write claimers at the point specified by the FORCE_AT option.

You cannot use this option to cancel threads when UNLOAD PLUS is attempting to acquire a lock on a table. For information about when UNLOAD PLUS uses lock table processing instead of drain processing, see “Drain and lock table processing in UNLOAD PLUS” on page 51.

**Overriding this option**

You can override the value for this option by using the FORCE command option. The FORCE command option provides an additional option, REPORTONLY, that tells UNLOAD PLUS to display a report of the threads that would have been canceled if you had specified FORCE ALL. For more information, see “FORCE” on page 179.
FORCE_AT=(START,3)

The FORCE_AT option tells UNLOAD PLUS when to cancel DB2 threads that might prevent the drain process from completing. This option is applicable only when FORCE ALL or FORCE REPORTONLY is in effect.

The first parameter of this option indicates at which point during the drain process to begin canceling threads. The second parameter indicates how long to delay the start of the cancelation process from the specified point in the drain process.

For the first parameter of this option, specify one of the following values:

- **START** tells UNLOAD PLUS to start canceling threads when the drain request begins.
- **RETRY** tells UNLOAD PLUS to start canceling threads the first time the drain process times out and UNLOAD PLUS attempts to retry the drain.
- **LASTRETRY** tells UNLOAD PLUS to start canceling threads at the beginning of the last retry attempt following a drain process timeout. You can control the number of drain retry attempts with the RETRY command option or DRNRETRY installation option.

**Note**

The cancelation process includes a small amount of time to issue the cancel commands, plus any time that DB2 needs to roll back the canceled transactions. If the drain wait time (controlled by the DRNWAIT option) is shorter than the total cancelation process time, the drain fails. The START and RETRY values allow for additional drain attempts after the start of the cancelation process; therefore, these values increase the likelihood that the drain will succeed when a short drain wait time is in effect. (Alternatively, you can increase the drain wait time.)

For the second parameter of this option, specify an integer value to indicate hundredths of a second. For example, specify 7 to tell UNLOAD PLUS to wait .07 seconds. Specify an integer value of 0 or greater. The default, 0, tells UNLOAD PLUS to start the cancelation process immediately upon reaching the point specified by the FORCE_AT option.

**Overriding this option**

You can override the value for the first parameter of this option by using the FORCE_AT suboption of the FORCE command option. You can override the value for the second parameter by using the FORCE_DELAY suboption of the FORCE command option. For more information, see “FORCE” on page 179.
FORCE_RPT=NO

The FORCE_RPT option tells UNLOAD PLUS whether to display a report of the canceled threads.

Specify one of the following values for this option:

- YES tells UNLOAD PLUS to display the report. This value is applicable only when FORCE ALL is in effect.
  - If your JCL includes a BMCFORCE DD statement, UNLOAD PLUS sends the canceled threads report to that data set. Otherwise, the report is displayed in the UNLOAD PLUS SYSPRINT.

- NO tells UNLOAD PLUS not to display the report.

UNLOAD PLUS ignores FORCE_RPT when you specify FORCE REPORTONLY on the UNLOAD command.

Overriding this option

You can override the value for this option by using the FORCE_RPT suboption of the FORCE command option. For more information, see “FORCE” on page 179.

HISTORY=YES

The HISTORY option specifies whether to update the BMCHIST table. If you specify YES, UNLOAD PLUS updates the BMCHIST table for each table space. If you specify NO for this option, UNLOAD PLUS bypasses any updates to the BMCHIST table.

IBUFFS=25

The IBUFFS option specifies a multiple of either 256 KB or 32 KB to use to define the amount of buffer pool storage that is allocated for image copy input data sets. Valid values are 1 through 255.

- For copy data sets on tape, UNLOAD PLUS uses a multiplier of 256 KB (to allow for block sizes greater than 32 KB). In this case, if the IBUFFS option has a value greater than 6, UNLOAD PLUS ignores that setting and uses 6.

- For all other copy data sets, UNLOAD PLUS uses a multiplier of 32 KB.
**INLINE=NO**

The INLINE option specifies whether to unload data from an inline image copy. This image copy is associated with a ddname specified with the INFILE option. You can specify one of the following options:

- Specify YES to treat the specified image copy as an inline copy. This option enables UNLOAD PLUS to unload data from out-of-sequence pages. If you specify INLINE=YES, you can use the same production jobs to unload from standard copies.

- Specify NO to not treat the specified image copy as an inline copy.

**WARNING**

UNLOAD PLUS supports unloading data from an inline image copy when you specify INFILE *ddname* and INLINE YES. Inline image copies can have out-of-sequence pages. However, UNLOAD PLUS cannot determine pages that might be out of sequence before processing begins unless you also specify INLINE YES. Using an inline copy that contains pages that are out of sequence can cause a serious page error (see message BMC50251S) if you specify INLINE NO.

**Overriding this option**

You can override the value for this option by using the INLINE option of the INFILE command option (“INFILE” on page 110).

**KEYDSNAM=**

The KEYDSNAM option specifies the key data set name to use when unloading encrypted copies that were created by COPY PLUS.

For more information about unloading encrypted copies, see the IMAGECOPY option of the INFILE command option (“INFILE” on page 110). For more information about this key data set, see the *COPY PLUS for DB2 Reference Manual*.

**LOADDECP=NO**

When unloading rows from an image copy that was originally created in a subsystem that is different from the local subsystem, it might be necessary to also access the DSNHDECP from the subsystem where the image copy was created.

To reference a DSNHDECP that is different from the local subsystem default, specify LOADDECP=YES in the UNLOAD PLUS installation options module and include the library that contains the desired DSNHDECP in the STEPLIB concatenation of the UNLOAD PLUS job.
To obtain all DB2 installation default parameters (except SSID) from the DSNHDECP that resides in the local subsystem, specify LOADDECP=NO.

SSID is always obtained from the DSNHDECP in the STEPLIB regardless of the value of LOADDECP in the UNLOAD PLUS installation options module.

**LOCKROW=**YES

The LOCKROW option tells UNLOAD PLUS which serialization method to use when updating the BMCSYNC and BMCUTIL tables.

- LOCKROW=YES tells UNLOAD PLUS to use MVS enqueues instead of the SQL LOCK TABLE statements for serialization. Using LOCKROW=YES should prevent most SQL -911 return codes that occur when multiple BMC products concurrently update the BMCSYNC and BMCUTIL tables.

- LOCKROW=NO tells UNLOAD PLUS to use SQL LOCK TABLE statements for serialization when updating the BMCSYNC and BMCUTIL tables.

**Additional considerations**

The following considerations apply to the LOCKROW option:

- Using LOCKROW=YES requires row-level locking. You must define the BMCSYNC and BMCUTIL tables with LOCKSIZE ROW to achieve row-level locking.

- The following BMC products must use the same LOCKROW value within a subsystem: CHECK PLUS, COPY PLUS, LOADPLUS, RECOVER PLUS, REORG PLUS, and UNLOAD PLUS. Regardless of the value assigned for LOCKROW, COPY PLUS and RECOVER PLUS always behave as if LOCKROW=YES.

- In a future version, UNLOAD PLUS will no longer require or support the LOCKROW installation option.

**MAXP=5**

The MAXP option specifies the maximum number of partitions to unload concurrently from any one task. This option is also useful for limiting the number of tape drives that UNLOAD PLUS requires simultaneously if you specify the INFILE IMAGECOPY command option to unload from image copies on tape.

**MSGLEVEL=1**

The MSGLEVEL option controls the messages that UNLOAD PLUS returns in the SYSPRINT and SYSPRIN2 data sets.
- MSGLEVEL=0 returns minimal messages.
- MSGLEVEL=1 returns additional messages to help you diagnose problems and fine-tune performance.

**Overiding this option**

You can override the value for this option by using the MSGLEVEL parameter on the EXEC statement. For more information about this parameter, see “Utility parameters on the UNLOAD PLUS EXEC statement” on page 272.

**NULLCHAR=?**

The NULLCHAR option specifies the null indicator character in the output record. Specify one of the following values:

- ?
- Any valid character constant for C-type assembler language
- Any valid hexadecimal assembler constant, in the format X'xx'
- HIVAL, which fills the null indicator field with high values

This value signifies only a single byte. If the null indicator field is two bytes long, UNLOAD PLUS propagates the constant to the second byte.

When you are using the CNTLCARDS SQL/DS, SQL/DS-DDL, or SQL/DS-LOAD option, the only valid specification for NULLCHAR is ?. If you try to use any other NULLCHAR value, UNLOAD PLUS issues message BMC51610E and terminates.

---

**Note**

When you specify or default to ?, the job SYSPRINT displays a value of X'6F'.

---

**Overiding this option**

You can override the value for this option by using the NULLCHAR command option (“NULLCHAR” on page 155).

**NULLTYPE=T1**

The NULLTYPE option specifies the location and length of the null indicator field in the output record. Specify one of the following values:

- T1 (the null indicator is one byte long and follows the column)
■ T2 (the null indicator is two bytes long and follows the column)
■ L1 (the null indicator is one byte long and precedes the column)
■ L2 (the null indicator is two bytes long and precedes the column)

When you are using the CNTLCARDS SQL/DS, SQL/DS-DDL, or SQL/DS-LOAD option, the only valid specification for NULLTYPE is T1. If you try to use any other NULLTYPE value, UNLOAD PLUS issues message BMC51610E and terminates.

Overriding this option

You can override the value for this option by using the NULLTYPE command option (“NULLTYPE” on page 156).

OPNDB2ID=YES

The OPNDB2ID option tells UNLOAD PLUS whether to use the DB2 RACF ID or the user’s RACF ID.

■ OPNDB2ID=YES tells UNLOAD PLUS to use the DB2 RACF ID (instead of the RACF ID of the user who is running UNLOAD PLUS) when opening DB2 data sets.
■ OPNDB2ID=NO tells UNLOAD PLUS to use the RACF ID of the user who is running UNLOAD PLUS. If you specify NO, the user must have the appropriate RACF authorization.

For any security system other than RACF, you should specify OPNDB2ID=NO so that UNLOAD PLUS uses the ID of the user who is running the utility. For more information, see “Setting UNLOAD PLUS authorizations” on page 43.

PLAN=ADU1120

The PLAN option specifies the name of the product plan. This plan contains the packages that provide the various capabilities of UNLOAD PLUS.

RECFM=AUTO

The RECFM option allows you to override the record format for the unload data sets that UNLOAD PLUS determines to be optimal. You can specify one of the following keywords:

■ AUTO tells UNLOAD PLUS to determine the optimal record format.
■ VB tells UNLOAD PLUS to always use variable block record format (VB).
Note

If the record length of your unload file is greater than the system maximum block size, UNLOAD PLUS defines the data set as variable-block spanned (VBS) instead of VB, and issues message BMC51745W.

If you are unloading LOB or XML data, UNLOAD PLUS ignores this option for any referenced files that it allocates.

Overriding this option

You can override the value for this option by using the RECFM command option (“RECFM” on page 128).

**ROWSETSZ=100**

This option is valid only if DIRECT NO is in effect.

The ROWSETSZ option tells UNLOAD PLUS how many rows to include in a rowset for a single FETCH request.

You can specify one of the following values:

- 0 or 1 tells UNLOAD PLUS to fetch a single row at a time.
- Any integer value from 2 through 200 tells UNLOAD PLUS to fetch the specified number of rows (as a rowset).

Restriction

UNLOAD PLUS ignores ROWSETSZ when you are unloading LOB or XML data.

Overriding this option

You can override the value for this option by using the DIRECT NO ROWSETSZ command option.

**SDUMP=(ALLPSA,CSA,RGN,SQA,LSQA,SUM,TRT,IO)**

The SDUMP option tells UNLOAD PLUS to generate a system dump, using the information listed in the option, if an unload job abnormally terminates. The values listed provide diagnostic information to BMC Customer Support. Although you can specify any values that IBM allows for the SDATA parameter on the SDUMPX macro, BMC recommends that you do not change the values that were shipped with the product. For a complete list and description of the values that you can specify for this option, see the IBM SDUMPX macro description.
UNLOAD PLUS uses the system-defined dump data set to hold the data. In cases where multiple abends occur, UNLOAD PLUS generates the dump for only the first abend. The SDUMP option allows you to generate a system dump, regardless of your access to storage keys 0 through 7.

You can also specify SDUMP=NO to not generate a system dump.

**Note**
UNLOAD PLUS displays SDUMP=YES in message BMC50471I in the SYSPRINT list of options when SDUMP contains one or more values.

To limit the conditions under which UNLOAD PLUS generates the system dump, you can exclude selected abend codes by using the EXCLDUMP installation option.

**SHRLEVEL=REFERENCE**

The SHRLEVEL option specifies the default SHRLEVEL that UNLOAD PLUS uses to unload objects. SHRLEVEL=REFERENCE restricts access to objects to read-only during UNLOAD PLUS processing. UNLOAD PLUS then externalizes the pages of the table space. When UNLOAD PLUS processing completes, the utility restores the object to its original status.

If you are unloading from a table space, specify SHRLEVEL=CHANGE to allow read/write access to the table space during unload processing. UNLOAD PLUS ignores the SHRLEVEL option if you specify the DIRECT NO or INFILE ddname command options.

**Overriding this option**

You can override the value for this option by using the SHRLEVEL command option (“SHRLEVEL” on page 103).

**SMAX=16**

The SMAX option specifies the number of subtasks that can execute concurrently during an unload. If the unload requires sorting, this option limits the number of concurrent sorts.

A value of 16 allows UNLOAD PLUS to run in optimized mode. To reduce the amount of system resources that UNLOAD PLUS uses if other processing requirements constrain resources, you might need to specify a lower value for this option.

For more information, see “Performance considerations for sort processing options in UNLOAD PLUS” on page 405.
Overriding this option

You can override the value for this option by using the MAXSORTS command option (“MAXSORTS” on page 147).

SMCORE=(0K,0K)

The SMCORE option specifies the amount of memory that you want each invocation of BMCSORT to use. BMC strongly recommends that you use the 0K and 0K values for this option. For more information, see “Performance considerations for sort processing options in UNLOAD PLUS” on page 405.

The values 0K and 0K tell UNLOAD PLUS to determine the appropriate amount of memory to use for each sort based on the following criteria:

- The number of sorts to process
- The amount of memory that is available during optimization
- The value specified for REGION in either your JCL or system exits

The first value specifies the total amount of memory to use both above and below the 16-megabyte line for each sort. You can specify this value in either kilobytes or megabytes. The following values are valid for this parameter:

- 0K (or 0M) tells UNLOAD PLUS to determine the appropriate amount.
- Any value from 4096K through 65536K (or 4M through 64M) tells UNLOAD PLUS to use the specified amount.

The second value specifies the amount of memory required below the 16-megabyte line for each sort. You can specify this value in either kilobytes or megabytes. The following values are valid for this parameter:

- 0K (or 0M) tells UNLOAD PLUS to determine the appropriate amount.
- Any value from 256K through 4096K (or 1M through 4M) tells UNLOAD PLUS to use the specified amount.

Note

If you specify a value that is greater than 32 for the SORTNUM option, BMC recommends that you specify 384K for the second value of the SMCORE option.
**SORTNUM=32**

The SORTNUM option affects the allocation of sort work files when BMCSORT is allocating your sort work files dynamically. You can specify any integer from 0 through 255.

The following table describes the action that BMCSORT takes for each value that you can specify for this option. The table also provides any additional considerations for these values.

**Table 100: SORTNUM values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Additional considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BMCSORT honors the value of the third parameter of the BMCSORT DYNALOC installation option. (This parameter tells BMCSORT whether to dynamically allocate sort work files.)</td>
<td>For more information about how this parameter affects dynamic allocation, see “SORTWK data sets in UNLOAD PLUS” on page 281. For more information about this parameter, see “DYNALOC installation option” on page 463.</td>
</tr>
<tr>
<td>1-32</td>
<td>BMCSORT dynamically allocates the number of sort work files that it needs, up to 32 minus any sort work files that are allocated in your JCL. This number is per task.</td>
<td>None</td>
</tr>
<tr>
<td>33-255</td>
<td>BMCSORT dynamically allocates the number of sort work files that it needs, up to the number that you specify minus any sort work files that are allocated in your JCL. This number is per task.</td>
<td>In this case, BMC strongly recommends that you change the second value of the SMCore option to 384K.</td>
</tr>
</tbody>
</table>

If the value of the third parameter in the BMCSORT DYNALOC installation option is OFF, specifying a value greater than 0 for the SORTNUM option turns BMCSORT dynamic allocation on, and BMCSORT allocates sort work files as needed. For information about when BMCSORT allocates your sort work files dynamically, see “SORTWK data sets in UNLOAD PLUS” on page 281.

**Overriding this option**

You can override the value for this option by using the SORTNUM command option (“SORTNUM” on page 142).

**SQLDELAY=3**

The SQLDELAY option specifies the number of seconds to wait between retry attempts after receiving an SQL-911 return code. This interval is in addition to the time that elapses when DB2 waits for a timeout or deadlock. The number of seconds can range from 1 to 655.
**SQLRETRY=100**

The SQLRETRY option specifies the number of times to retry an SQL statement. The number of attempts can range from 1 to 255.

**TAPEDISP=DELETE**

The TAPEDISP option specifies the final disposition of tape files when the value of the DELETEFILES option is YES. TAPEDISP=DELETE specifies that each tape file will have a disposition of OLD, DELETE, DELETE when the tape is deallocated.

If you specify TAPEDISP=UNCATLG, each tape will have a disposition of OLD, UNCATLG, UNCATLG when the tape is deallocated. Depending on your tape management environment, using UNCATLG can prevent a tape remount.

**TAPES=NONE**

The TAPES option specifies a list of tape units that tells UNLOAD PLUS to identify a unit as a tape device for dynamically allocated data sets. This option is not required in most circumstances because UNLOAD PLUS dynamically determines the device type for its output data sets. You should not need to specify the tape units unless UNLOAD PLUS is unable to identify your units automatically.

You can specify a list of tapes in the format (tape1,tape2,.......tapen).

**TASKMAX=200%**

The TASKMAX option specifies the maximum number of tasks to start and is the global setting for all tasks. You can specify one of the following values:

- An integer from 1 through 16, indicating that UNLOAD PLUS can start up to that number of tasks
- A percentage, indicating that UNLOAD PLUS can start that percentage of the number of online CPUs on the system

You can override this option with other options, such as UNLDMAX, for specific multitasking operations.

**UBUFFS=25**

The UBUFFS option specifies a multiple of 32 KB to use to define the amount of buffer pool storage that is allocated for unload (SYSREC) data sets. Valid values are 1 through 255. For more information, see “Performance considerations for buffer installation options in UNLOAD PLUS” on page 404.
**UNLDMAX=**

The UNLDMAX option specifies the maximum number of tasks to start per CPU during the UNLOAD phase. Although you can specify an integer or percentage value, BMC strongly recommends that you accept the default of * and manage multitasking by using the TASKMAX and SMAX options. An asterisk value tells UNLOAD PLUS to use the value for the TASKMAX option.

Regardless of the value that you specify, UNLOAD PLUS does not start more than 16 tasks for this phase.

**UNLOADDN_ACTIVE=(YES,NO)**

The UNLOADDN_ACTIVE option activates or deactivates dynamic allocation of the primary and secondary unload data sets. Specify YES for the primary or secondary parameter to activate dynamic allocation for the corresponding unload data set. Specify NO to deactivate dynamic allocation for the corresponding unload data set. You can specify the ACTIVE keyword following the UNLOADDN specification (for example, UNLOADDN(SYSREC, SYSRED) ACTIVE(YES,YES)) in the SYSIN command stream to override the specified installation value.

If you activate dynamic allocation in installation options, UNLOAD PLUS dynamically allocates any unload data sets that are required but are not specified in the JCL. If you specify all required unload data sets in the JCL, UNLOAD PLUS does not use dynamic allocation.

**UNLOADDN=(SYSREC, SYSRED)**

The UNLOADDN option provides UNLOAD PLUS with one of the following pieces of information:

- The default ddname or ddname prefix for the primary and secondary unload data sets
  - If you change the ddnames that BMC supplies, you must also change the names in your JCL or OUTPUT statements.

- The output descriptor or prefix to match to the OUTPUT statement (or $ADUOUTP macro) when allocating unload data sets dynamically
  - During installation, the names that you specify for this option become the default output descriptor names (see “Options that are common to disk and tape data sets” on page 449) that are associated with the dynamic allocation options for the primary and secondary unload data sets. For example, if you specify (OUT1, OUT2) for this option, the Installation System prompts you to supply dynamic allocation option values for OUT1 and OUT2.
The output descriptors or output descriptor prefixes for the primary unload data sets, secondary unload data sets, and referenced files must all be unique. (The FILEREFDN option specifies the default output descriptor for referenced files.)

Overriding this option

You can override the value for this option by using the UNLOADDN command option (“UNLOADDN” on page 121).

USELRECL=NO

The USELRECL option specifies whether UNLOAD PLUS checks and uses the logical record length (LRECL) value either from the JCL DD DCB statement or the OUTPUT statement that you specify in your command options for the primary unload data set that corresponds to each SELECT statement. Specify NO if you want UNLOAD PLUS to use the calculated record length. Specify YES to use the specified LRECL.

If you specify YES, UNLOAD PLUS

- Uses fixed block for the data set record format and pads it with spaces of the appropriate encoding scheme when necessary

  Note
  If you specify RECFM VB on your UNLOAD command (or it is in effect by default from the RECFM installation option) UNLOAD PLUS uses variable block format for the data set records.

- Forces the same DCB values in the corresponding secondary unload data set if you request or specify a secondary data set

- Calculates the LRECL if you did not specify an LRECL value on the DD statement or in the OUTPUT statement

  If UNLOAD PLUS calculates the LRECL value, the data set is fixed, variable, or variable-spanned according to the standard record length and record format calculations for UNLOAD PLUS.

Restrictions

Note the following restrictions on USELRECL=YES:

- If the specified LRECL is shorter than the length that is required to contain the largest formatted row, UNLOAD PLUS ends with error message BMC51649E.

- UNLOAD PLUS ignores USELRECL=YES if you specify any of the following options:
— FORMAT CSV  
— FORMAT XML  
— FORMAT INTERNAL  
— FORMAT BMCLOAD

- If you are unloading LOB or XML data, UNLOAD PLUS ignores USELRECL=YES for any referenced files that it allocates.

**Overriding this option**

You can override the value for this option by specifying the USELRECL command option (“USELRECL” on page 126).

**UXSTATE=SUP**

The UXSTATE option tells UNLOAD PLUS how to invoke DB2 user exits:

- UXSTATE=SUP specifies that UNLOAD PLUS should call DB2 user exits (such as EDITPROCs) in supervisor state (and PSW key=7).

- UXSTATE=PROB tells UNLOAD PLUS to call DB2 user exits in problem state (and PSW key=7).

The requirements of the exits dictate the UXSTATE setting. Check with the exit author (or vendor) before changing to UXSTATE=PROB.

**WORKUNIT=SYSALLDA**

The WORKUNIT option specifies a temporary work data set and is used when running a SHRLEVEL CHANGE unload or during the dynamic bind process. VIO is an acceptable value for this option.

**XBMID=**

The XBMID option identifies a specific XBM subsystem ID (SSID) to use for snapshot processing and zIIP processing in UNLOAD PLUS.

The SSID is the unique identifier that was specified when XBM or SUF was installed. If you are using XBM or SUF in a DB2 data sharing environment, you can use the value of the XBMGROUP parameter instead of the XBM SSID. The XBMGROUP name is the name of the cross-system coupling facility (XCF) group that is defined to the XBM subsystem, and its default value is XBMGROUP.

If you specify an XBM subsystem, it must be:
Available

At a supported maintenance level

Enabled for the required function

If you do not specify an XBM subsystem (either here or with the XBMID installation option), UNLOAD PLUS automatically searches for an XBM subsystem that meets the same criteria. If you have multiple subsystems that meet these criteria, UNLOAD PLUS can use any one of these subsystems.

For more information about using XBM or SUF for snapshot processing with UNLOAD PLUS, see “Running SHRLEVEL CHANGE CONSISTENT YES” on page 53.

**Overriding this option**

You can override the value for this option by using the XBMID command option (“XBMID” on page 108).

**ZIIP=ENABLED**

The ZIIP option tells UNLOAD PLUS whether to attempt to use IBM z Integrated Information Processors (zIIPs). UNLOAD PLUS can use enclave service request blocks (SRBs) to enable zIIP processing automatically while running jobs. Using zIIP processing can reduce the overall CPU time for UNLOAD PLUS jobs.

You can specify one of the following options:

- ENABLED tells UNLOAD PLUS to attempt to offload eligible processing to an available zIIP. If the zIIP is busy or not available, normal processing continues on a general-purpose processor.

- DISABLED tells UNLOAD PLUS to not attempt to use zIIP processing.

To enable and use zIIP processing with UNLOAD PLUS, you must

- Have an installed authorized version of XBM or SUF

- Start and maintain an XBM subsystem in your environment

- Have a zIIP available in your environment

**Using XBM or SUF**

You can specify a particular XBM subsystem to use by specifying a value for the XBMID installation or command option.
XBM and SUF are licensed, installed, and maintained separately from UNLOAD PLUS. You can use either XBM or SUF, depending on the license that you have obtained:

- A license for the full version of the XBM product authorizes you to use all features of XBM.
- A license for SUF authorizes you to use only the snapshot and zIIP-processing features of XBM.

**Note**
If you are licensed only for a BMC solution that contains UNLOAD PLUS, your license authorizes you to use SUF, not the full version of XBM.

For more information about XBM and SUF, see the *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide*.

**Overriding this option**

You can override the value for this option by using the ZIIP command option (“ZIIP” on page 106).

**ZONEDDECOVP=(C,D)**

The ZONEDDECOVP option assigns overpunch values to decimal-zoned numeric values. The overpunched values can be positive or negative.

- The first operand specifies the zone overpunch value for positive numbers. Standard positive overpunch values are A, C, E, and F.
- The second operand specifies the zone overpunch value for negative numbers. Standard negative overpunch values are B and D.

This option can be useful when you plan to use the unloaded data with an application that requires a specific zone value that is not the traditional default.

To obtain the absolute value of a number or to ensure that the value is in data external format, specify ZONEDDECOVP (F,F).

**Overriding this option**

You can override the value for this option by using the ZONEDDECOVP command option (“ZONEDDECOVP” on page 165).
Dynamic allocation installation options

The $ADUOUTP macro produces the installation options for dynamic allocation.

You can use installation default settings or UNLOAD command options to activate and configure dynamic allocation of primary unload data sets, secondary unload data sets, and referenced files. Specify the default settings for dynamic allocation in the $ADUOUTP macro during installation. The $ADUOUTP specifications are assembled within the standard options module.

To activate dynamic allocation of unload data sets in the installation options module, specify UNLOADDN_ACTIVE=YES. The values for the UNLOADDN option in the $ADUOPTS macro provide the output descriptor names for the primary and secondary unload data sets. To activate dynamic allocation for these data sets on the UNLOAD command, specify UNLOADDN outputDescriptor ACTIVE YES.

UNLOAD PLUS always dynamically allocates referenced file data sets.

Default output descriptor options for dynamic allocation

This topic describes the values shipped with UNLOAD PLUS for the output descriptor options.

UNLOAD PLUS uses these options to allocate output data sets dynamically. You can specify some options for both disk and tape. You can use others only with disk data sets (“Options for disk data sets” on page 457) and others only with tape data sets (“Options for tape data sets” on page 461).

Table 101 on page 448 shows the dynamic allocation default options. For each option, the table provides the value that ships with this version of UNLOAD PLUS (or No value if the option was shipped without a value), a brief description, and a reference to more details. If an option ships with no value, the table shows a recommended value or example value.

An output descriptor describes the characteristics of the primary unload data sets, secondary unload data sets, and referenced output files, whether they are written to disk or tape. An output descriptor must specify either a disk or a tape data set, but not both. Each output descriptor DD has its own set of installation option values.

You can override many default values at runtime by specifying new values in an OUTPUT descriptor statement in the input control data.
Table 101: UNLOAD PLUS installation options for dynamic allocation

<table>
<thead>
<tr>
<th>Option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATACLAS</td>
<td>No value</td>
<td>SMS data class name</td>
<td>“DATACLAS=“ on page 456</td>
</tr>
<tr>
<td></td>
<td>Example value: STANDARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISKEXPD</td>
<td>No value</td>
<td>Expiration date for disk data sets</td>
<td>“DISKEXPD=“ on page 461</td>
</tr>
<tr>
<td></td>
<td>Example value: 2006020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISKRETN</td>
<td>No value</td>
<td>Retention period for disk data sets</td>
<td>“DISKRETN=“ on page 460</td>
</tr>
<tr>
<td></td>
<td>Example value: 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNNAME</td>
<td>Value depends on the type of data set</td>
<td>Default name for the output data set</td>
<td>“DSNAME=“ on page 450</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>(for SYSREF) PDS (for SYSREC and SYSRED) NONE</td>
<td>Type of referenced file or unload data set being used or allocated</td>
<td>“DSNTYPE=PDS or NONE” on page 453</td>
</tr>
<tr>
<td>EXPDT</td>
<td>99000</td>
<td>Expiration date for tape data sets</td>
<td>“EXPDT=99000“ on page 462</td>
</tr>
<tr>
<td>FILESZPCT</td>
<td>100</td>
<td>Percentage of calculated data set size to allocate when using automatic sizing</td>
<td>“FILESZPCT=100“ on page 459</td>
</tr>
<tr>
<td>GDGEMPTY</td>
<td>NO</td>
<td>Whether to uncatalog all data sets when the limit is reached</td>
<td>“GDGEMPTY=NO“ on page 455</td>
</tr>
<tr>
<td>GDGLIMIT</td>
<td>5</td>
<td>Maximum number of generation data sets in a group</td>
<td>“GDGLIMIT=5“ on page 455</td>
</tr>
<tr>
<td>GDGSCRATCH</td>
<td>NO</td>
<td>Whether to keep DSCB information when the data sets are uncataloged</td>
<td>“GDGSCRATCH=NO“ on page 455</td>
</tr>
<tr>
<td>MAXPRIM</td>
<td>0</td>
<td>Maximum primary allocation</td>
<td>“MAXPRIM=0“ on page 458</td>
</tr>
<tr>
<td>MAXSECD</td>
<td>0</td>
<td>Maximum secondary allocation</td>
<td>“MAXSECD=0“ on page 458</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>No value</td>
<td>SMS management class name</td>
<td>“MGMTCLAS=“ on page 456</td>
</tr>
<tr>
<td></td>
<td>Example value: VSAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBRSECD</td>
<td>AUTO</td>
<td>Number of secondary extents</td>
<td>“NBRSECD=AUTO“ on page 459</td>
</tr>
<tr>
<td>PCTPRIM</td>
<td>AUTO</td>
<td>Primary allocation percentage of disk space</td>
<td>“PCTPRIM=AUTO“ on page 457</td>
</tr>
<tr>
<td>RETPD</td>
<td>No value</td>
<td>Retention period for tape data sets</td>
<td>“RETPD=“ on page 462</td>
</tr>
<tr>
<td></td>
<td>Example value: 30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Options that are common to disk and tape data sets

You can use the following options for disk and tape data sets.

**output**

The output option specifies the name to use for this default output descriptor. The default output descriptor for each type of output data set must match the corresponding name that you specified in the UNLOADDN or FILEREFDN installation option.

During installation, you specify the value for this option by using the UNLOADDN or FILEREFDN option. For example, if you specify (OUT1, OUT2) for the UNLOADDN option, the Installation System prompts you to supply dynamic allocation option values for the output descriptors OUT1 and OUT2. You cannot change these names on the dynamic allocation options panels.

**UNIT=SYSALLDA**

The UNIT option specifies the device name for syntax checking and dynamic disk allocation. The name that BMC supplies for the unit is SYSALLDA. UNLOAD PLUS compares the value of UNIT to a list of tape devices that UNLOAD PLUS retrieves from MVS. If the value does not match one of the devices on the list, UNLOAD PLUS assumes that the device is disk.
Specify $NO_UNIT$ to tell UNLOAD PLUS not to pass a unit value to dynamic allocation. This value is particularly useful in an SMS environment.

If you also specify DSNTYPE=EXTPREF or DSNTYPE=EXTREQ, ensure that the unit supports extended attributes.

**Overriding this option**

You can override the value for this option by using the UNIT command option ("UNIT" on page 183).

**DSNAME=**

The DSNAME option specifies a default data set name for the output data set. At runtime, any value that you provide for DSNAME overrides this option. The value that is shipped with UNLOAD PLUS depends on the type of output data set:

<table>
<thead>
<tr>
<th>Type of file</th>
<th>Shipped default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary and secondary unload data sets</td>
<td>&amp;USERID.&amp;JOBNAME.&amp;TS.&amp;TYPE.S&amp;SELNUM</td>
</tr>
<tr>
<td>Referenced files</td>
<td>&amp;USERID.&amp;JOBNAME.&amp;TYPE.S&amp;SELNUM.P&amp;POSNUM</td>
</tr>
</tbody>
</table>

### Unload data sets

You can use symbolic variables to construct names for your unload data sets, specifying any or all nodes of a data set name. When you use a symbolic variable, you can prefix it with an alphabetic character, but you cannot append characters. For example, XX&TS and &TSXX are valid, but &&TSXX is invalid. Table 103 on page 451 describes the symbolic variables that you can use.

### Referenced files

For a referenced file, you must specify the fully qualified data set or file system name with the DSNAME option. You can use a pattern with DSNAME as follows:

- For DSNTYPE PDS or LIBRARY, you can use a pattern to create your data set names.

- For DSNTYPE HFS when you specify DIRECT YES, you can use a pattern to have UNLOAD PLUS create additional subdirectories in your file system path. In the following example, the root portion of the path (/home/rdacxb) already exists and is the mount point for the file system. UNLOAD PLUS adds subdirectories to...
this path for the subsystem ID and partition number based on the two variables in
the DSNAME pattern.

```
OUTPUT CLOB01 DSNAME('/home/rdacxb/&SSID/p&PART') DSNTYPE(HFS)
```

When you specify DIRECT NO, specifying a pattern does not result in additional
subdirectories in your file system path. UNLOAD PLUS terminates if it cannot
find a file system path that uses the name that resolves from your pattern.

Note the following considerations about the name that you specify:

- When DSNTYPE is PDS or LIBRARY, you cannot specify an existing data set.
- You cannot specify a PDS member name or HFS file name. UNLOAD PLUS
generates PDS members and HFS files.

**Symbolic variables for data set name patterns**

The following table lists the symbolic variables that you can use for data set name
patterns, a definition of each, and limits on length. Note the following additional
information about data set name patterns and using these variables:

- The maximum total length that UNLOAD PLUS allows for a data set name is 44
  bytes.
- UNLOAD PLUS removes any trailing blanks in the result.
- For variables that result in a number, you must prefix those symbols with one or
  more alphabetic characters.
- UNLOAD PLUS assigns values for date and time variables when the utility
  allocates the output data set. All dynamically allocated data sets have the same
  value.

**Table 103: Symbolic variables for data set names within SYSIN**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ATTACH</td>
<td>DB2 group attachment name or subsystem ID</td>
<td>4 maximum</td>
</tr>
<tr>
<td>&amp;DATE</td>
<td>Current date (in the format YYMMDD)</td>
<td>6</td>
</tr>
<tr>
<td>&amp;DAY</td>
<td>Current day (in the format DD)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;DB</td>
<td>Database</td>
<td>8</td>
</tr>
<tr>
<td>&amp;HOUR</td>
<td>Current hour (in the format HH)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;JDATE</td>
<td>Current Julian date (in the format YYDDD)</td>
<td>5</td>
</tr>
<tr>
<td>&amp;JDAY</td>
<td>Current Julian day (in the format DDD)</td>
<td>3</td>
</tr>
<tr>
<td>&amp;JOBNAME</td>
<td>Job name</td>
<td>8 maximum</td>
</tr>
<tr>
<td>Variable</td>
<td>Definition</td>
<td>Length (bytes)</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>&amp;MIN</td>
<td>Current minute (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;MINUTE</td>
<td>Current minute (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;MONTH</td>
<td>Current month (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;OBID</td>
<td>OBID of the table that UNLOAD PLUS is unloading</td>
<td>4 bytes maximum</td>
</tr>
<tr>
<td>&amp;PART</td>
<td>Data set or partition that UNLOAD PLUS is unloading</td>
<td>3 bytes in the following circumstances:</td>
</tr>
<tr>
<td></td>
<td>The following information applies to &amp;PART:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ If you specify the LOGICAL keyword with the PART option, the &amp;PART variable indicates the logical partition rather than the physical partition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Consider &amp;PART to limit the number of partitions unloaded in a single step to avoid encountering data set allocation restrictions of the operating system.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ For referenced file data sets, UNLOAD PLUS creates one data set for each base table space partition. If you use this variable, be aware that UNLOAD PLUS will terminate if it attempts to allocate more than 256 data sets for referenced files.</td>
<td></td>
</tr>
<tr>
<td>&amp;POSNUM</td>
<td>Position number of the XML or LOB column within the select list</td>
<td>3</td>
</tr>
<tr>
<td>&amp;SEC</td>
<td>Current second (in the format SS)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;SECOND</td>
<td>Current second (in the format SS)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;SELNUM</td>
<td>SELECT statement number</td>
<td>3</td>
</tr>
<tr>
<td>&amp;SSID</td>
<td>DB2 subsystem ID</td>
<td>4</td>
</tr>
<tr>
<td>&amp;STEPNAME</td>
<td>Step name</td>
<td>8 maximum</td>
</tr>
<tr>
<td></td>
<td>Note: UNLOAD PLUS ignores PROC names.</td>
<td></td>
</tr>
<tr>
<td>&amp;TIME</td>
<td>Current time (in the format HHMMSS)</td>
<td>6</td>
</tr>
<tr>
<td>&amp;TS</td>
<td>Table space</td>
<td>8 maximum</td>
</tr>
</tbody>
</table>

Note: UNLOAD PLUS ignores PROC names.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;TYPE</td>
<td>Data set type that UNLOAD PLUS is allocating This symbol resolves to a value of SYSREC, SYSRED, or SYSREF (where SYSREF indicates that this is a referenced file).</td>
<td>6</td>
</tr>
<tr>
<td>&amp;USERID</td>
<td>Job or TSO user</td>
<td>7 maximum</td>
</tr>
<tr>
<td>&amp;UTIL</td>
<td>Utility ID Note the following information about the utility ID variable:  ■ UNLOAD PLUS truncates values longer than 8 bytes.  ■ Utility IDs that include special characters might cause UNLOAD PLUS to generate invalid data set names.</td>
<td>8 maximum</td>
</tr>
<tr>
<td>&amp;YEAR</td>
<td>Current year (in the format YY)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Overriding this option**

You can override the value for this option by using the DSNAME command option (“DSNAME” on page 184).

**DSNTYPE=PDS or NONE**

The DSNTYPE option provides the following functionality:

- For referenced files, the PDS, LIBRARY, and HFS values for DSNTYPE tell UNLOAD PLUS which type of file you are using. UNLOAD PLUS requires this option when you are unloading LOB or XML data to a referenced file. For more information about using referenced files, see “Unloading LOB and XML data” on page 54.

- For unload data sets (SYSREC and SYSRED), the remaining values for DSNTYPE tell UNLOAD PLUS what type of data set to allocate dynamically.

The following table describes the valid values for this option:
Table 104: Valid DSNTYPE values

<table>
<thead>
<tr>
<th>Type of file</th>
<th>DSNTYPE value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced files</td>
<td>PDS</td>
<td>Allocate a partitioned data set.</td>
</tr>
<tr>
<td></td>
<td>LIBRARY</td>
<td>Allocate an extended partitioned data set</td>
</tr>
<tr>
<td></td>
<td>HFS</td>
<td>Use a hierarchical file system. The only other valid dynamic allocation installation options with DSNTYPE HFS are the output descriptor name, DSNAME, and SUBSETS. The HFS named in DSNAME must be preallocated and mounted before the UNLOAD PLUS job starts.</td>
</tr>
<tr>
<td>Unload data sets</td>
<td>NONE</td>
<td>Allocate an unload data set without extended attributes.</td>
</tr>
<tr>
<td></td>
<td>LARGE</td>
<td>Allocate an unload data set as a large format sequential data set (larger than 65,535 tracks).</td>
</tr>
<tr>
<td></td>
<td>BASIC</td>
<td>Allocate an unload data set as a basic sequential data set (limited to 65,535 tracks).</td>
</tr>
<tr>
<td></td>
<td>EXTREQ</td>
<td>Allocate an unload data set as an extended format data set. Ensure that the unit specified with the UNIT option supports extended attributes.</td>
</tr>
<tr>
<td></td>
<td>EXTPREF</td>
<td>Allocate an unload data set as an extended format data set if possible. If this allocation is not possible, the data set is allocated as a basic format data set. Ensure that the unit specified with the UNIT option supports extended attributes.</td>
</tr>
</tbody>
</table>

Overriding this option

You can override the value for this option by using the DSNTYPE command option (“DSNTYPE” on page 189).

VOLCNT=25

The VOLCNT option specifies the value for the largest number of volumes that you expect UNLOAD PLUS to process when it unloads any tape data set. The valid range of values for VOLCNT is 0 through 255, and the value that BMC supplies is 25. The number must be large enough to accommodate the number of volumes that UNLOAD PLUS produces for the single largest unload.

For disk data set allocations, UNLOAD PLUS ignores VOLCNT. Use UNITCNT to request a multi-volume disk data set.

Additional considerations
Note the following considerations for VOLCNT:

- Do not use a higher value than your system allows.
- To use the MVS default, set VOLCNT=0.
- If you are using SMS in your system, BMC recommends that you set VOLCNT=0 if your ACS routines are set up to provide a volume count.
- If you anticipate unloading a large number of partitions, consider lowering the value of this option to avoid data set allocation limitations.

**Overriding this option**

You can override the value for this option by using the VOLCNT command option ("VOLCNT" on page 193).

**GDGLIMIT=5**

The GDGLIMIT option specifies the maximum number of generation data sets that can be in the group. The number can range from 1 through 255.

**Overriding this option**

You can override the value for this option by using the GDGLIMIT command option ("GDGLIMIT" on page 193).

**GDGEMPTY=NO**

The GDGEMPTY option specifies how you want UNLOAD PLUS to uncatalog the generation data sets. NO tells UNLOAD PLUS to uncatalog only the oldest generation data set in the generation data group when the utility has allocated the maximum number that the GDGLIMIT option specifies, and one or more data sets will be added to the group.

If you specify YES for this option, UNLOAD PLUS uncatalogs all generation data sets in the generation data group when the utility has allocated the maximum number that the GDGLIMIT option specifies.

**GDGSCRATCH=NO**

GDGSCRATCH=NO specifies that when UNLOAD PLUS uncatalogs a generation data set, the utility does not remove the data set control block (DSCB) from the VTOC of the volume on which each data set resides.
If you specify YES for this option, UNLOAD PLUS removes (scratches) the descriptive DSCB of each generation data set from the VTOC when the utility uncatalogs the generation data set.

**STORCLAS=**

The STORCLAS option specifies an SMS storage class name for output data sets. The name must be a valid SMS storage class name that does not exceed eight characters. If you do not specify a value for the command option or installation option, UNLOAD PLUS uses the SMS installation value.

UNLOAD PLUS uses any SMS or STORCLAS values that are in the current default output descriptor for both disk and tape data set allocations unless you override the value in an associated OUTPUT descriptor statement. (You can use STORCLAS NONE for this purpose.) Check your options settings in the current default output descriptor.

*Overriding this option*

You can override the value for this option by using the STORCLAS command option (“STORCLAS” on page 194).

**DATACLAS=**

The DATACLAS option specifies an SMS data class name for output data sets. The name must be a valid SMS data class name that does not exceed eight characters. If you do not specify a value for the command option or installation option, UNLOAD PLUS uses the SMS installation value.

UNLOAD PLUS uses any SMS or DATACLAS values that are in the current default output descriptor for both disk and tape data set allocations unless you override the value in an associated OUTPUT descriptor statement. (You can use DATACLAS NONE for this purpose.) Check your options settings in the current default output descriptor.

*Overriding this option*

You can override the value for this option by using the DATACLAS command option (“DATACLAS” on page 195).

**MGMTCLAS=**

The MGMTCLAS option specifies an SMS management class name for output data sets. The name must be a valid SMS management class name that does not exceed eight characters. If you do not specify a value for the command option or installation option, UNLOAD PLUS uses the SMS installation value.
UNLOAD PLUS uses any SMS or MGMTCLAS values that are in the current default output descriptor for both disk and tape data set allocations unless you override them in an associated OUTPUT descriptor statement. (You can use MGMTCLAS NONE for this purpose.) Check your options settings in the current default output descriptor.

*Overriding this option*

You can override the value for this option by using the MGMTCLAS command option (“MGMTCLAS” on page 195).

**UNITCNT=0**

The UNITCNT option specifies the unit count used for dynamic allocation. Valid values are 0 through 59. If you specify 0, UNLOAD PLUS does not specify the unit count for the allocation.

For a referenced file when the UNLOAD command includes DIRECT NO, specify 0 or 1.

*Overriding this option*

You can override the value for this option by using the UNITCNT command option (“UNITCNT” on page 196).

### Options for disk data sets

You can use the following options for disk data sets only.

**SPACE=CYL**

The SPACE option specifies the space unit of allocation that UNLOAD PLUS uses when dynamically allocating output data sets. Specify CYL for cylinders or TRK for tracks.

*Overriding this option*

You can override the value for this option if you use the SPACE command option in an OUTPUT descriptor statement (“SPACE” on page 197).

**PCTPRIM=AUTO**

The PCTPRIM option specifies a value for the amount of the disk space that UNLOAD PLUS should allocate as primary space. The value that is shipped with the product is AUTO. If you specify AUTO for SMS allocations, UNLOAD PLUS uses...
the largest available extent as the value for the primary allocation. If you specify
AUTO for non-SMS allocations, UNLOAD PLUS uses a value of 100. If you do not
specify AUTO, you can specify any integer from 1 through 100.

Note
The primary allocation that UNLOAD PLUS calculates for large table spaces using
PCTPRIM can be too large. You can use the MAXPRIM installation option to
override the calculated value.

UNLOAD PLUS ignores this option for referenced files.

Overriding this option
You can override the value for this option by specifying the PCTPRIM command
option in an OUTPUT descriptor statement (“PCTPRIM” on page 197).

MAXPRIM=0

The MAXPRIM option specifies the maximum amount of disk space (in the units
that SPACE specified) that UNLOAD PLUS may allocate as primary space. The
value of 0 specifies that there is not a limit, but a nonzero value establishes an upper
limit on the value that PCTPRIM calculates. Valid values are

- 0 through 65535 when the value of SPACE is TRK
- 0 through 4369 when the value of SPACE is CYL

UNLOAD PLUS ignores this option for referenced files.

Overriding this option
You can override the value for this option by specifying the MAXPRIM command
option in an OUTPUT descriptor statement (“MAXPRIM” on page 198).

MAXSECD=0

The MAXSECD option specifies the maximum amount of disk space (in the units
specified by the SPACE installation or command option) that UNLOAD PLUS may
allocate as secondary space. The value of 0 specifies that there is not a limit. A
nonzero value establishes an upper limit on the secondary space allocation. Valid
values are 0 through 65535 when the value of SPACE is TRK.

The following considerations apply to this option:

- UNLOAD PLUS ignores this option for referenced files.
Specifying the MAXSECD option when a data set is being sized automatically may result in additional units being allocated and may cause the data set to be undersized.

When you are using the SMS-guaranteed space allocation, UNLOAD PLUS applies MAXSECD to all of the volumes that you specified in the VOLUMES command option or in the installation options module.

**Overriding this option**

You can override the value for this option by using the MAXSECD command option in an OUTPUT descriptor statement ("MAXSECD" on page 199).

**FILESZPCT=100**

The FILESZPCT option specifies the amount by which UNLOAD PLUS should adjust the total calculated space for all data sets that your OUTPUT statements describe when using automatic data set sizing. Values from 1 through 99 indicate a reduction in file size. Values from 101 through 999 indicate an increase in file size. A value of 100 indicates no change in the file size.

UNLOAD PLUS uses the following formula to calculate the resulting total file size:

\[
\text{resultantFileSize} = \text{ROUND} \left( \frac{\text{calculatedFileSize} \times \text{FILESZPCT/100}}{100} \right)
\]

UNLOAD PLUS uses the value calculated by FILESZPCT as the total size (in kilobytes) of the data set before it calculates the primary and secondary space values specified by the PCTPRIM, NBRSECD, and MAXSECD options.

UNLOAD PLUS ignores this option for referenced files.

**Overriding this option**

You can override the value for this option by using the FILESZPCT command option ("FILESZPCT" on page 200).

**NBRSECD=AUTO**

The NBRSECD option specifies the size of the secondary allocation of disk space. After UNLOAD PLUS calculates the primary space allocation, the remaining space is secondary space that may be whole (1 part) or divided into as many as 123 extents per volume. You can specify AUTO, or an integer from 1 through 123.

If you specify AUTO, UNLOAD PLUS considers the following criteria and uses a value of either 16 or 123:

- If the allocation is non-SMS or it is SMS and the data set will not be in extended format, UNLOAD PLUS uses a value of 16.
If the allocation is SMS and UNLOAD PLUS will allocate the data set in the extended format, UNLOAD PLUS uses a value of 123.

If you specify an integer for non-SMS allocations, the maximum value that you can specify for NBRSECD is 16. If you specify a value that is greater than 16, UNLOAD PLUS still uses a value of 16.

UNLOAD PLUS ignores this option for referenced files.

_Overriding this option_

You can override the value for this option if you use the NBRSECD command option in an OUTPUT descriptor statement (“NBRSECD” on page 200).

**VOLUMES**=

The VOLUMES option specifies a default list of disk volumes. Specify the VOLUMES option only when you are not using SMS and want to direct the unload output to specific volumes.

To specify a list of volumes, use the format `(vol1,vol2,....,voln)`. Note, however, that the length of the option name and its value is restricted to 255, which limits the number of volumes that you can specify in the options module. Additionally, the number of entries in the list must not exceed the value of the VOLCNT option.

For referenced files, UNLOAD PLUS functions as follows:

- For DSNTYPE HFS, UNLOAD PLUS ignores this option.
- For DSNTYPE PDS or LIBRARY, UNLOAD PLUS uses only the first volume in the list.

_Overriding this option_

You can override the value for this option by using the VOLUMES command option in an OUTPUT descriptor statement (“VOLUMES” on page 201).

**DISKRETN**=

The DISKRETN option specifies the retention period in days for a disk data set. The number of days must be in the range 1 through 9999.

_**Note**_

When you specify the DISKEXPD installation option, DISKEXPD supersedes DISKRETN.
Restriction

This option is restricted for referenced files as follows:

- For DSNTYPE HFS, UNLOAD PLUS ignores this option.
- For DSNTYPE PDS or LIBRARY, UNLOAD PLUS terminates if you specify a value for this option.

Overriding this option

You can override the value for this option by specifying the DISKRETN command option in an OUTPUT descriptor statement (“DISKRETN” on page 202).

DISKEXPD=

The DISKEXPD option specifies the expiration date for disk data sets. If you do not specify DISKEXPD, UNLOAD PLUS does not provide an expiration date. The date must be in the format YYYYDDD.

- YYYY represents the four-digit year.
- DDD represents the 3-digit Julian day (001 through 366).

Note

When you specify the DISKEXPD installation option, DISKEXPD supersedes DISKRETN.

Restriction

This option is restricted for referenced files as follows:

- For DSNTYPE HFS, UNLOAD PLUS ignores this option.
- For DSNTYPE PDS or LIBRARY, UNLOAD PLUS terminates if you specify a value for this option.

Overriding this option

You can override the value for this option by specifying the DISKEXPD command option in an OUTPUT descriptor statement (“DISKEXPD” on page 203).

Options for tape data sets

You can use the following options for tape data sets only.
These options are not valid for referenced files.

**RETPD=**

The RETPD option specifies the retention period in days for a tape data set. The number of days must range from 1 through 9999.

*Note*
When you specify the EXPDT installation option, EXPDT supersedes RETPD.

**Overriding this option**

You can override the value for this option by using the RETPD command option (“RETPD” on page 204).

**EXPDT=99000**

The EXPDT option specifies the expiration date for tape data sets. The number 99000 specifies that the tape data sets have no expiration date. The date must be in the format *YYYYDDD*.

- *YYYY* represents the four-digit year.
- *DDD* represents the 3-digit Julian day (001 through 366).

*Note*
When you specify the EXPDT installation option, EXPDT supersedes RETPD.

**Overriding this option**

You can override the value for this option by specifying the EXPDT command option in an OUTPUT descriptor statement (“EXPDT” on page 205).

**TRTCH=NONE**

The TRTCH option specifies whether UNLOAD PLUS should use data compression for tape data sets. Specify TRTCH=COMP to provide tape data compression, or specify TRTCH=NOCOMP to prevent tape data compression. TRTCH=NONE, the value that BMC supplies, specifies that you want to use the MVS default.

**Overriding this option**

You can override the value for this option by specifying the TRTCH command option in an OUTPUT descriptor statement (“TRTCH” on page 203).
The DYNALOC installation option of BMCSORT provides information for dynamically allocating SORTWK data sets.

BMCSORT deallocates these data sets at the end of each sort. The content of the $AUPSMAC macro in $532SOPT follows, showing DYNALOC and the values that are shipped with BMCSORT.

```
$AUPSMAC DYNALOC=(SYSDA,3,ON,ON,6000000,3000000,3390,SC=,RETRY=(0,0)) X
DYNAMIC_ALLOC_OPTIONS_FOR_SORT
```

The values that you specify in this macro apply to all invocations of BMCSORT. BMCSORT uses the same options module for all BMC products that invoke BMCSORT.

Table 105 on page 463 describes each parameter of the DYNALOC option. These parameters are positional. The values that you specify for these parameters should correspond to your site’s standards for any system sort routine.

BMCSORT overrides the values that you supplied if BMCSORT determines that it can complete sorting more efficiently than the specified values allow. An invoking product’s options might also override the BMCSORT options values that you specify when one of the following conditions exists:

- The values in the invoking product’s dynamic allocation installation options or corresponding command options conflict with the values that you specify.
- You turn on BMCSORT SORTWK dynamic allocation from the product that invokes BMCSORT, and you specify OFF for the position 3 parameter.

BMCSORT dynamically allocates SORTWK files as necessary.

**Table 105: DYNALOC parameters**

<table>
<thead>
<tr>
<th>Parameter name or position</th>
<th>Description</th>
<th>Initial value</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 1</td>
<td>This parameter specifies the generic unit name from which the BMC product should dynamically allocate SORTWK data sets. This parameter applies only when the Data Facility Storage Management System (DFSMS) product from IBM is not installed or is not active for temporary DASD work data sets. If DFSMS is active, use the SC parameter.</td>
<td>SYSDA</td>
<td>Use a unit name up to 8 characters.</td>
</tr>
<tr>
<td>Parameter name or position</td>
<td>Description</td>
<td>Initial value</td>
<td>Valid values</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Position 2</td>
<td>Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>3</td>
<td>Do not change this value.</td>
</tr>
</tbody>
</table>
| Position 3                | This parameter tells the BMC product whether to dynamically allocate SORTWK files. BMC recommends that you not change this value. | ON | ■ ON dynamically allocates SORTWK.  
■ OFF does not dynamically allocate SORTWK. |
| Position 4                | Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro. | ON | Do not change this value. |
| Position 5                | Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro. | 6000000 | Do not change this value. |
| Position 6                | Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro. | 3000000 | Do not change this value. |
| Position 7                | This parameter specifies the DASD type with the smallest track capacity that a dynamically allocated SORTWK data set might encounter at your site. | 3390 | ■ 3380, track capacity of 47968  
■ 3390, track capacity of 56664  
■ 9345, track capacity of 46456 |
<p>| SC                        | This parameter specifies the name of the DFSMS storage class from which to dynamically allocate SORTWK. If DFSMS is active and you do not specify a value for this parameter, the BMC product uses the value from the first DYNALOC parameter. <strong>Note:</strong> If your installation has an automatic class selection (ACS) routine, it can override this specification. | blank | Use any valid DFSMS storage class. |</p>
<table>
<thead>
<tr>
<th>Parameter name or position</th>
<th>Description</th>
<th>Initial value</th>
<th>Valid values</th>
</tr>
</thead>
</table>
| RETRY                     | This parameter specifies how you want the BMC product to handle retry attempts for SORTWK dynamic allocation:  
  ■ The first subparameter indicates the number of times that you want the BMC product to retry the request.  
  ■ The second subparameter indicates the number of minutes to wait between each retry.  
  Using this parameter allows you to avoid a capacity-exceeded condition when disk space is not immediately available for a SORTWK dynamic allocation request.  
  BMC recommends that you do not change this value because it can affect the elapsed time of your jobs. However, if you currently use SyncSort and rely on the retry function, BMC recommends that you use the same values as your SyncSort RETRY installation parameter. | (0,0) | If you use this parameter, BMC recommends that you specify the same values as your SyncSort RETRY installation parameter. The following values are valid for this parameter:  
  ■ 0 through 16 for the first subparameter  
  0 indicates that you do not want the BMC product to retry the request.  
  ■ 0 through 15 for the second subparameter  
  0 indicates that you do not want the BMC product to retry the request. |
This chapter describes the contents of the common utility tables, considerations for these tables, and how to maintain them if necessary.

Overview of common utility tables

The BMC common utility tables contain information about the BMC utilities that you generate and submit through a BMC utility product.

Table 106 on page 467 lists the tables that each utility uses and each table’s default name and synonym.

Table 106: Common utility tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Default name</th>
<th>Synonym</th>
<th>Utilities that use this table</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCDICT</td>
<td>CMN_BMCDICT</td>
<td>BMC_BMCDICT</td>
<td>■ LOADPLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ REORG PLUS</td>
</tr>
<tr>
<td>BMCHIST</td>
<td>CMN_BMCHIST</td>
<td>BMC_BMCHIST</td>
<td>■ CHECK PLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ COPY PLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ LOADPLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ RECOVER PLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ REORG PLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ UNLOAD PLUS</td>
</tr>
<tr>
<td>Table</td>
<td>Default name</td>
<td>Synonym</td>
<td>Utilities that use this table</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| BMCLGRNX  | CMN_BMCLGRNX | BMC_BMCLGRNX | • COPY PLUS  
                        • Log Master  
                        • RECOVER PLUS  
                        • RECOVERY MANAGER |
| BMCSYNC   | CMN_BMCSYNC  | BMC_BMCSYNC  | • CHECK PLUS  
                        • COPY PLUS  
                        • DASD MANAGER PLUS (BMCSTATS)  
                        • LOADPLUS  
                        • RECOVER PLUS  
                        • RECOVERY MANAGER  
                        • REORG PLUS  
                        • UNLOAD PLUS |
| BMCTRANS  | CMN_BMCTRANS | BMC_BMCTRANS | • Log Master  
                        • RECOVERY MANAGER |
<table>
<thead>
<tr>
<th>Table</th>
<th>Default name</th>
<th>Synonym</th>
<th>Utilities that use this table</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCUTIL</td>
<td>CMN_BMCUTIL</td>
<td>BMC_BMCUTIL</td>
<td>■ CHECK PLUS&lt;br&gt;■ COPY PLUS&lt;br&gt;■ DASD MANAGER PLUS (BMCSTATS)&lt;br&gt;■ LOADPLUS&lt;br&gt;■ RECOVER PLUS&lt;br&gt;■ RECOVERY MANAGER&lt;br&gt;■ REORG PLUS&lt;br&gt;■ UNLOAD PLUS</td>
</tr>
<tr>
<td>BMCXCOPY</td>
<td>CMN_BMCXCOPY</td>
<td>BMC_BMCXCOPY</td>
<td>■ COPY PLUS&lt;br&gt;■ Log Master&lt;br&gt;■ RECOVER PLUS&lt;br&gt;■ RECOVERY MANAGER&lt;br&gt;■ REORG PLUS&lt;br&gt;■ UNLOAD PLUS</td>
</tr>
</tbody>
</table>

**Warnings and considerations for common utility tables**

This topic describes important information that you need to know when using the common utility tables.
The following warnings apply to the common utility tables:

- Do not run LOADPLUS, REORG PLUS, or UNLOAD PLUS against the BMC common utility tables or table spaces. Doing so can cause unpredictable results.

- Because RECOVER PLUS uses BMC tables during the recovery process, you cannot use RECOVER PLUS to recover any BMC table except the BMCHIST table.

- Do not run the RUNSTATS utility against the BMC common utility tables. Doing so can negatively impact utility performance.

- BMC strongly recommends that you use the ISOLATION (UR) bind option and issue SQL COMMIT statements when querying the tables in the BMC database. If objects in the BMC database are restricted for UPDATE, the executing BMC utilities might not be able to complete successfully.

Note the following considerations:

- Some columns in the tables are present for compatibility with specific BMC utilities and are not used by all of the utilities.

- If you have applications that depend on the structure or content of these tables, be aware that these tables are subject to change.

- In general, the utility tables should not require maintenance, with the exception of BMCHIST.

- You should back up the BMC table spaces on a regular basis to enable recoveries. If you use COPY PLUS as the copy utility, you must use SHRLEVEL CHANGE for the following spaces:
  - BMCUTIL
  - BMCHIST
  - BMCSYNC
  - BMCXCOPY

- Supported versions of the following BMC products support the LOCKROW installation option:
  - CHECK PLUS
  - LOADPLUS
  - REORG PLUS
If the value of the option is YES (which is the default value), the products use MVS enqueues instead of SQL LOCK TABLE statements to serialize updates to the BMCSYNC and BMCUTIL tables.

The following BMC products always use MVS enqueues for serialization when updating the BMCSYNC and BMCUTIL tables:

— COPY PLUS

— DASD MANAGER PLUS

— RECOVER PLUS

Managing common utility tables

This topic provides basic procedures for working with the common utility tables.

To determine your site’s table names

The names of the common utility tables can be changed during installation.

1 To determine the names that your site uses, perform one of the following actions:

   ■ Use your utility to run a job with restart parameters of MAINT and MSGLEVEL(1).
     Specifying MSGLEVEL(1) with MAINT prints the names of the BMC tables that your utility uses and identifies the applied maintenance. The utility does not perform any other processing, and the job ends without affecting any utility that is running.

   ■ Run the following SQL statement, replacing tableName with a BMC common utility table name (listed in “Overview of common utility tables” on page 467):

     ```sql
     SELECT CREATOR,NAME FROM SYSIBM.SYSTABLES
     WHERE TSNAME='tableName';
     ```

   ■ Get the names from your DB2 system administrator.

To query the tables

1 Run SQL statements similar to the following examples.
Example

This example queries the BMCXCOPY table to access information about the rows in an index space:

```sql
SELECT *
FROM creatorName.CMN_BMCXCOPY
WHERE DBNAME = 'databaseName'
  AND IXNAME = 'indexSpaceName'
ORDER BY START_RBA;
```

This example identifies (from the BMCHIST table) the database name, table space name, elapsed time, and when the utility completed:

```sql
SELECT DBNAME, SPNAME, CHAR(ELAPSED, ISO), CHAR(TIME, ISO)
FROM creatorName.CMN_BMCHIST
WHERE UTILID = 'utilityID';
```

To display BMC utility status

1. To display the status of all BMC utilities that are executing or awaiting restart for a given table space or index space, use the following SQL statements:

```sql
SELECT * FROM creatorName.CMN_BMCUTIL
WHERE DBNAME = 'databaseName'
  AND SPNAME = 'tableSpaceName'
SELECT * FROM creatorName.CMN_BMCSYNC
WHERE NAME1 = 'databaseName'
  AND NAME2 = 'spaceName';
```

To terminate a BMC utility

1. To terminate a BMC utility, perform one of the following actions:

   ■ To terminate a BMC utility that is executing, use the following SQL statements:

   ```sql
   DELETE FROM creatorName.CMN_BMCUTIL
   WHERE UTILID = 'utilityID';
   DELETE FROM creatorName.CMN_BMCSYNC
   WHERE UTILID = 'utilityID';
   DELETE FROM creatorName.CMN_BMCDICT -- for LOADPLUS and REORG PLUS
   WHERE UTILID = 'utilityID';
   ```

   The utility terminates with return code 8 when the next checkpoint is taken.

   ■ To clean up a BMC utility that is not executing, run the utility with the correct utility ID and specify the TERM restart parameter.

BMCDICT table

The BMCDICT table stores the compression dictionary during load or reorganization processing.

Table 107 on page 473 describes the contents of the BMCDICT table.
Table 107: Contents of the BMCDICT table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>TSNAME</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>PARTITION</td>
<td>SMALLINT</td>
<td>Partition number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a nonpartitioned table space, the value is 0.</td>
</tr>
<tr>
<td>SEQNO</td>
<td>SMALLINT</td>
<td>Sequence number</td>
</tr>
<tr>
<td>DICTDATA</td>
<td>VARCHAR(4000)</td>
<td>Dictionary data</td>
</tr>
</tbody>
</table>

BMCDICT table considerations

This topic describes important information that you need to know about the BMCDICT table:

- If you are processing a large number of compressed partitions, you might need to increase the size of the BMCDICT table space significantly from the standard size that was allocated during installation. To estimate the allocation, multiply 64 KB by the number of compressed partitions that you are processing concurrently (loading with LOADPLUS or reorganizing with REORG PLUS).

- LOADPLUS inserts rows into the BMCDICT table during the PRELOAD phase and deletes those rows following compression processing in the LOAD phase.

- REORG PLUS inserts rows into the BMCDICT table during the UNLOAD phase and deletes those rows following compression processing in the RELOAD phase.

Maintaining the BMCDICT table

If LOADPLUS or REORG PLUS abends during the time between building the compression dictionary and completing compression, rows might remain in the BMCDICT table.

On rare occasions, you might need to take action to control expansion of the BMCDICT table.

To control expansion of the BMCDICT table

1 Delete any rows in the BMCUTIL table that you know are no longer valid.
Do not delete any rows for instances of utilities that are awaiting restart.

2 Use the following SQL statement to delete rows from the BMCDICT table:

```sql
DELETE
FROM creatorName.CMN_BMCDICT
WHERE UTILID NOT IN
(SELECT UTILID FROM creatorName.CMN_BMCUTIL);
```

Note

The names of the BMCUTIL and BMCDICT tables might have been changed at your site during installation.

BMCHIST table

The BMCHIST table contains information about completed executions of the BMC utilities for DB2.

The following configuration or installation options control use of the BMCHIST table:

- HISTORY (for COPY PLUS, RECOVER PLUS, and UNLOAD PLUS)
- BMCHIST (for REORG PLUS)

If the option value is NO, the utility bypasses any updates to the BMCHIST table. If the value is YES (or the utility does not use a configuration or installation option), the utility inserts rows into the BMCHIST table during the UTILTERM phase.

For COPY PLUS, if the value is SUMMARY, the utility inserts only summary information about the COPY PLUS execution into the BMCHIST table. This option provides less information than the YES option.

Table 108 on page 474 describes the contents of the BMCHIST table.

<table>
<thead>
<tr>
<th>Table 108: Contents of the BMCHIST table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column name</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>DBNAME</td>
</tr>
<tr>
<td>SPNAME</td>
</tr>
<tr>
<td>Column name</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>UTILNAME</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>UTILID</td>
</tr>
<tr>
<td>AUTHID</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>TIME</td>
</tr>
<tr>
<td>ELAPSED</td>
</tr>
<tr>
<td>PARTITION</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>OBJNAME</td>
</tr>
<tr>
<td>PHASE_1</td>
</tr>
<tr>
<td>ELAPSED_1</td>
</tr>
<tr>
<td>PHASE_2</td>
</tr>
<tr>
<td>ELAPSED_2</td>
</tr>
</tbody>
</table>
BMCHIST table considerations for COPY PLUS

COPY PLUS uses the BMCHIST table to record completed COPY and COPY IMAGECOPY command executions.

HISTRETN is available as a COPY PLUS installation option or as an option on the OPTIONS command. HISTRETN tells COPY PLUS the number of days to keep entries in the BMCHIST table.

**WARNING**

If you want to use BMCHIST, allocate adequate space for the table. COPY PLUS makes an entry in the table for every copied space. If you are copying a large number of partitions, you might need to increase the size of the BMCHIST table space from the standard size that was allocated during installation.

BMCHIST table considerations for RECOVER PLUS

For each execution of AFRMAIN, RECOVER PLUS writes a single row to the BMCHIST table.

DBNAME, SPNAME, and OBJNAME columns will always be blank.

RECOVER PLUS accumulates elapsed time for each of the following phases using the RECOVER PLUS phase shown:

- PHASE_1: LOGSORT
- PHASE_2: MERGE (includes RESTORE phase)
- PHASE_3: SNAP
- PHASE_4: REBUILD (includes UNLOAD phase)
- PHASE_5: DB2UUTIL (the time spent in DSNUTILB)

The elapsed time for each of the phases is a sum for all objects. The utility elapsed time, ELAPSED, is the duration from the start of the utility to until it finishes. Because RECOVER PLUS multitasks, the sum of the phases might be greater than the total elapsed time of the utility. The elapsed time columns have a limit of 24 hours.

## Maintaining the BMCHIST table

When a utility completes successfully, it inserts a row into the BMCHIST table. You can control expansion of this table by deleting old rows. If you use REORG PLUS, you can also control inserts into the BMCHIST table.

### To delete old rows from the BMCHIST table

1. To delete selected rows from the BMCHIST table based on the date that the utility completed, use the following sample SQL statement:

   ```sql
   DELETE
   FROM creatorName.CMN_BMCHIST
   WHERE DATE < 'yyyy-mm-dd';
   ```

### To control inserts into the BMCHIST table (REORG PLUS only)

1. Use the TERMEXIT option to specify a user exit that controls inserts into the BMCHIST table.

## BMCLGRNX table

The BMCLGRNX table contains log ranges that show when a table space was open for updates.

Table 109 on page 477 describes the contents of the BMCLGRNX table.

### Table 109: Contents of the BMCLGRNX table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGRDBID</td>
<td>CHAR(2)</td>
<td>DBID of the modified object</td>
</tr>
<tr>
<td>LGRPSID</td>
<td>CHAR(2)</td>
<td>OBID of the modified object</td>
</tr>
<tr>
<td>LGRUCDT</td>
<td>CHAR(6)</td>
<td>Modification date (mmddyy)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>LGRUCTM</td>
<td>CHAR(8)</td>
<td>Modification time (hhmmsssth)</td>
</tr>
<tr>
<td>LGRSRBA</td>
<td>CHAR(6)</td>
<td>Starting RBA</td>
</tr>
<tr>
<td>LGRSPBA</td>
<td>CHAR(6)</td>
<td>Stopping RBA</td>
</tr>
<tr>
<td>LGRPART</td>
<td>SMALLINT</td>
<td>Table space partition number</td>
</tr>
<tr>
<td>LGRSLRSN</td>
<td>CHAR(6)</td>
<td>Starting LRSN of update log records for data sharing</td>
</tr>
<tr>
<td>LGRELRSN</td>
<td>CHAR(6)</td>
<td>Ending LRSN of update log records for data sharing</td>
</tr>
<tr>
<td>LGRMEMBER</td>
<td>CHAR(2)</td>
<td>Data sharing member ID of the modifying DB2 subsystem</td>
</tr>
</tbody>
</table>

**Note**

RECOVERY MANAGER uses the BMCLGRNX table only for DB2 Versions 9 and 10. RECOVERY MANAGER uses the SYSIBM.SYSLGRNX table for DB2 versions greater than Version 10.

---

**BMCSYNC table**

The BMCSYNC table contains information about the status of the objects that the currently executing utilities are accessing.

Table 110 on page 478 describes the contents of the BMCSYNC table. The BMCSYNC table synchronizes and controls access to DB2 spaces by concurrently executing BMC utility products. If you have more than one BMC utility installed, all of these utilities should share the same BMCSYNC table.

The utilities insert rows into the BMCSYNC table during the UTILINIT phase. While the job executes, the utilities update the table as the status of the object changes. The utilities delete rows from the BMCSYNC table during the UTILTERM phase.

**Table 110: Contents of the BMCSYNC table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier (RECOVER PLUS) This column is blank when a RECOVER UNLOADKEYS command creates the row and then a RECOVER BUILDINDEX command reads and deletes the row.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| NAME1       | CHAR(8)   | Database name or creator name  
(DASD MANAGER PLUS) This value is the database name.  
(CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS) If the value for NAME1 would exceed 8 bytes or the value for NAME2 would exceed 18 bytes, NAME1 contains the DBID for the object. |
| NAME2       | CHAR(18)  | Space, table, or index name  
(DASD MANAGER PLUS) The BMCSTATS utility always inserts the space name (limited to a maximum of 8 characters).  
(CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS) If the value for NAME1 would exceed 8 bytes or the value for NAME2 would exceed 18 bytes, NAME2 contains the table OBID or index ISOBID of the object in hexadecimal format. |
| KIND        | CHAR(2)   | Type of object:  
■ IP (index partition)  
■ IX (index)  
■ TB (table)  
■ TP (table space partition)  
■ TS (table space)  
■ DD, DW, D1, D2 (dynamic work file allocation)  
■ CI (copy information)  
■ RD (restart data set block) |
| PARTITION   | SMALLINT  | Physical partition number:  
■ Null or 0 for a single data set nonpartitioned space  
■ Data set number for a multi-data-set, nonpartitioned space  
■ Partition number for a partitioned space  
(CHECK PLUS, COPY PLUS, DASD MANAGER PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS) The value is null or 0 for any nonpartitioned space. |
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCID</td>
<td>SMALLINT</td>
<td>Internal identifier of the object. DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>UTILNAME</td>
<td>CHAR(8)</td>
<td>Name of the executing utility:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ CHECK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ COPY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ STATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LOAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RECOVER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ REORG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ UNLOAD</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>CHAR(1)</td>
<td>Degree to which utilities can share this object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank means that no status is requested, and any other utility can obtain any status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ S allows sharing among any number of SHRLEVEL S utilities.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ X indicates that exclusive control is required. No other utility can run with SHRLEVEL X.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see “Shared access levels of BMC utilities” on page 46.</td>
</tr>
<tr>
<td>STATUS</td>
<td>CHAR(1)</td>
<td>Status of the utility or object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (indicates no processing has been done)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ C (for CHECK PLUS, indicates checked)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ L (for LOADPLUS, indicates loaded)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ U (for UNLOAD PLUS, indicates unloaded)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ R (for REORG PLUS, indicates reloaded)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XCOUNT</td>
<td>INTEGER</td>
<td>Number of rows or keys processed in the current phase DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>DDNAME</td>
<td>CHAR(8)</td>
<td>Check, load, unload, or work ddname DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>INTEGER</td>
<td>Number of blocks for the check, load, unload, or work data set DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>ORIG_STATUS</td>
<td>CHAR(8)</td>
<td>Encoded representation of the original DB2 status of the space (RECOVER PLUS) This column restores the DB2 status of a space after recovery, if necessary. DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>EXTRBA</td>
<td>CHAR(10)</td>
<td>(RECOVER PLUS) Log point at which this space was externalized RECOVER PLUS serialization logic uses this column. The other utilities do not use this column. Note: RECOVER PLUS no longer uses EXTRBA.</td>
</tr>
<tr>
<td>STATE</td>
<td>LONG VARCHAR</td>
<td>Restart information for the space For example, the STATE indicates the object state and sync information. DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>INSTANCE</td>
<td>SMALLINT</td>
<td>(RECOVERY MANAGER and RECOVER PLUS) Instance number of the current base objects (table and index) The default value is 1. The other utilities do not use this column.</td>
</tr>
</tbody>
</table>

**BMCSYNC table considerations**

This topic contains important information that you need to know about the BMCSYNC table:

- By default, DASD MANAGER PLUS uses the BMCSYNC table to synchronize access to DB2 spaces. However, if you want to turn this feature off, you may do so by specifying No for the BMCSYNC installation option. If you specify No for this option, DASD MANAGER PLUS does not use the BMCSYNC table and the product bypasses BMCUTIL table access, UTILID enqueue logic, and object name enqueue logic used for BMC utility concurrency control. Turning this feature off can lead to VSAM data set access failures in BMCSTATS or other utilities due to utility conflicts that are no longer detected.
- You might need to increase the size of the BMCSYNC table space from the standard size that was allocated during installation when any of the following conditions exists:
  
  — You are processing a large number of partitions.
  
  Estimate this allocation based on the following factors:
  
  — Number of utilities that you are executing concurrently
  
  — Number of partitions that you are processing concurrently
  
  — Number of files that you are allocating dynamically
  
  — You are loading a partition-by-growth table space.
  
  Estimate this allocation based on the following factors:
  
  — Number of utilities that you are running concurrently
  
  — Value of MAXPARTITIONS
  
  — Number of files that you are allocating dynamically
  
  — You are loading or unloading XML data and the XML table space is partition-by-growth.
  
  Estimate this allocation based on the following factors:
  
  — Number of utilities that you are executing concurrently
  
  — Number of XML columns that you are loading or unloading
  
  — Value of MAXPARTITIONS (a minimum of 256 partitions in this case)
  
  — Number of files that you are allocating dynamically
  
  — You are loading or unloading LOB data.
  
  Estimate this allocation based on the following factors:
  
  — Number of utilities that you are executing concurrently
  
  — Number of LOB columns that you are loading or unloading
  
  — Number of partitions in the base table space
  
  — Number of files that you are allocating dynamically
Maintaining the BMCSYNC table

When a utility abends, rows might remain in the BMCSYNC table. On rare occasions, you might need to take action to control expansion of the BMCSYNC table.

To control expansion of the BMCSYNC table

1. Use one of the following methods to delete rows in the BMCSYNC table:
   - Use the TERM restart parameter on the EXEC statement to delete rows from both the BMCUTIL and BMCSYNC tables. Do not delete any rows for instances of utilities that are awaiting restart.
   - Delete invalid rows from the BMCUTIL table. Do not delete any rows for instances of utilities that are awaiting restart.

   Then use the following SQL statement to delete rows from the BMCSYNC table:

   ```sql
   DELETE
   FROM creatorName.CMN_BMCSYNC
   WHERE UTILID NOT IN
   (SELECT UTILID FROM creatorName.CMN_BMCUTIL);
   
   Note
   The names of the BMCUTIL and BMCSYNC tables might have been changed at your site during installation.
   
Cleaning up RECOVER UNLOADKEYS entries

Successful completion of a RECOVER UNLOADKEYS job leaves rows in BMCSYNC with blank utility IDs for table space partitions and indexes related to the unloaded keys. The table space rows prevent other BMC utilities from obtaining exclusive control of the table space.

To clean up RECOVER UNLOADKEYS entries

1. Use one of the following methods to remove the invalid BMCSYNC rows:
   - Run a RECOVER BUILDINDEX job.
   - Run a job that uses the following statement for the table space and each index:

     ```sql
     DELETE FROM creatorName.CMN_BMCSYNC
     WHERE UTILID=''
     AND NAME1='databaseName'
     AND NAME2='spaceName'
     AND UTILNAME='RECOVER';
     ```
Shared access levels of BMC utilities

BMC utility jobs register DB2 objects in the BMCSYNC table.

The registering utility assigns a sharing level to each registered object. The sharing level controls access to that object from other BMC utilities. For partitioned DB2 spaces, registration is performed at the partition level.

**Note**

All BMC utility products use the BMCUTIL table to control the use of utility IDs, which identify executions of BMC utilities. Each BMC utility product must have a unique ID for restart purposes. This unique ID is stored in the BMCUTIL table. For more information about this table, see “BMCUTIL table” on page 488.

The BMCSYNC table allows multiple BMC utilities (or multiple instances of a utility) to operate concurrently on different partitions of a DB2 space if no nonpartitioning indexes are involved. In addition, some BMC utilities can operate concurrently on the same object or partition. For information about which products can operate concurrently, see the following table. For additional serialization and concurrency issues for each utility, see that utility’s reference manual.

The "Access level" column in the following table refers to the value of the SHRLEVEL column name in the BMCSYNC table (“BMCSYNC table” on page 478). The level can be one of the following values:

- S indicates shared access. Any other utility that registers with shared access (S) can run against the object.
- X indicates exclusive access. No other utility can run against the object.
- A blank value indicates that no status is requested and any other utility can run against the object.

**Table 111: Shared access levels of BMC utilities**

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK PLUS</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>COPY PLUS</td>
<td>S or blank</td>
<td>If you specify COPY IMAGECOPY, COPY PLUS registers the object with no access status (blank). Otherwise, COPY PLUS registers the object with shared access (S).</td>
</tr>
<tr>
<td>Product</td>
<td>Access level</td>
<td>Additional information</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| DASD MANAGER PLUS (BMCSTATS)    | S            | - If BMCSTATS is processing multiple objects and encounters an object that is held by another utility, the BMCSTATS job issues a warning. The warning identifies the object and the utility that is using it. BMCSTATS continues processing the next object.  
- If BMCSTATS is processing an object and another utility requires exclusive control of that object, the other utility stops execution at initialization time. |
**WARNING**
Do not run an IBM utility, command, or SQL statement that attempts to manipulate the structure, data, or status of an object that a BMC utility is currently processing. For example, commands and SQL statements such as -STOP, -START, EXCHANGE, and ALTER will produce unpredictable results.

---

**BMCTRANS table**

The BMCTRANS table contains information that RECOVERY MANAGER and Log Master use for transaction recovery.

Table 112 on page 486 describes the contents of the BMCTRANS table. The table contains one row for each execution of Log Master (that is, one row for each log scan performed).

**Table 112: Contents of the BMCTRANS table**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8) NOT NULL</td>
<td>Transaction creator</td>
</tr>
<tr>
<td>TRANID</td>
<td>VARCHAR(18) NOT NULL</td>
<td>Transaction ID</td>
</tr>
<tr>
<td>STARTTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Transaction start time</td>
</tr>
<tr>
<td>PITRBA</td>
<td>CHAR(6) NOT NULL FORBIT DATA</td>
<td>RBA for point-in-time recovery</td>
</tr>
<tr>
<td>OUTDSNAME</td>
<td>VARCHAR(35) NOT NULL</td>
<td>Output data set prefix for SQL statements or the logical log</td>
</tr>
<tr>
<td>STATE</td>
<td>SMALLINT NOT NULL</td>
<td>Level of recovery analysis performed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 (only UNDO analysis has been performed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 through 9999 (UNDO and PIT analysis have been performed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Greater than 10000 (UNDO, PIT, and REDO analysis have been performed)</td>
</tr>
<tr>
<td>PITTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Timestamp for the PIT RBA</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>SEQNO</td>
<td>SMALLINT NOT NULL</td>
<td>Sequence number of the filter text</td>
</tr>
<tr>
<td>PITWKEST</td>
<td>FLOAT NOT NULL</td>
<td>Work estimate</td>
</tr>
<tr>
<td>FILTERLINE</td>
<td>VARCHAR(1040) NOT NULL</td>
<td>Text of the filter (may span more than one row)</td>
</tr>
<tr>
<td>UNDONUMROWSUPD</td>
<td>FLOAT</td>
<td>Number of unique rows (RIDs) that are selected by the filter of the log scan</td>
</tr>
<tr>
<td>UNDOSUBSEQUPDROWS</td>
<td>FLOAT</td>
<td>Total number of anomaly log records relating to one of the rows (RIDs) selected by the log scan</td>
</tr>
<tr>
<td>UNDOLOGRECROWS</td>
<td>FLOAT</td>
<td>Number of unique rows (RIDs) that are affected by an anomaly log record</td>
</tr>
<tr>
<td>UNDOJOBSTATUS</td>
<td>SMALLINT</td>
<td>Code indicating the status of an UNDO log scan:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 (no action taken)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 (Log Master execution started)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 2 (Log Master execution completed successfully with return code 0,4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 3 (Log Master execution completed unsuccessfully with return code 8,12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 4 (Log Master execution abnormally ended)</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REDOJOBSTATUS</td>
<td>SMALLINT</td>
<td>Code indicating the status of a REDO log scan:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 (no action taken)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 (Log Master execution started)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 2 (Log Master execution completed successfully with return code 0,4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 3 (Log Master execution completed unsuccessfully with return code 8,12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 4 (Log Master execution abnormally ended)</td>
</tr>
<tr>
<td>ENDTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Transaction end time</td>
</tr>
<tr>
<td>ACTION</td>
<td>SMALLINT</td>
<td>Code indicating what recovery, if any, has been performed on the transaction</td>
</tr>
</tbody>
</table>

**BMCUTIL table**

The BMCUTIL table contains information about utilities that are currently running or started.

Table 113 on page 489 describes the contents of the BMCUTIL table. The utilities use the table to control the use of utility IDs. Each BMC utility must have a unique ID for restart purposes. If you have more than one BMC utility installed, all of these utilities should share the same BMCUTIL table.

The utilities insert rows into the BMCUTIL table during the UTILINIT phase and update the table as the job status changes. The utilities delete rows from the BMCUTIL table during the UTILTERM phase.
Table 113: Contents of the BMCUTIL table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
</tbody>
</table>
| STATUS      | CHAR(1)   | Execution status of the utility:  
  - A (active, not executing command)  
  - I (initializing)  
  - P (pausing or pause-stopped)  
  - S (stopped)  
  - T (terminating)  
  - X (executing command)  
  *(DASD MANAGER PLUS)* The value for this column is always X. |
| UTILNAME    | CHAR(8)   | Name of the executing utility:  
  - CHECK  
  - COPY  
  - STATS  
  - LOAD  
  - RECOVER  
  - REORG  
  - UNLOAD |
| PHASE       | CHAR(8)   | Current phase of the utility  
  COPY PLUS does not use this column. |
| USERID      | CHAR(8)   | User ID executing the utility |
| SSID        | CHAR(4)   | DB2 subsystem where the utility is running |
### BMCUTIL table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTART</td>
<td>CHAR(1)</td>
<td>Restart option:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (not restart)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ P (RESTART(PHASE))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (RESTART)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>NOTEID</td>
<td>CHAR(8)</td>
<td>TSO user ID to be notified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td><em>(RECOVER PLUS and REORG PLUS)</em> Name of the database containing the table or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>index space for which the last checkpoint was taken</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value can be blank.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
<tr>
<td>SPNAME</td>
<td>CHAR(8)</td>
<td><em>(RECOVER PLUS and REORG PLUS)</em> Name of the table or index space for which</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the last checkpoint was taken.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value can be blank.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
<tr>
<td>SPSTATUS</td>
<td>CHAR(5)</td>
<td><em>(REORG PLUS)</em> Space status before the utility stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
<tr>
<td>COMMANDNO</td>
<td>SMALLINT</td>
<td>Not used (always 0)</td>
</tr>
<tr>
<td>COMMAND</td>
<td>VARCHAR(256)</td>
<td>First 256 characters of the utility command text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER PLUS, DASD MANAGER PLUS, and COPY PLUS do not use this column.</td>
</tr>
<tr>
<td>STATE</td>
<td>LONG VARCHAR</td>
<td>Utility state and sync information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>START_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Starting timestamp of the utility</td>
</tr>
</tbody>
</table>
Maintaining the BMCUTIL table

When a utility abends, rows might remain in the BMCUTIL table.

On rare occasions, you might need to take action to control expansion of the BMCUTIL table.

**To control expansion of the BMCUTIL table**

1. Use one of the following methods to delete rows from the BMCUTIL table:
   - Use the TERM restart parameter on the EXEC statement to delete rows from both the BMCUTIL and BMCSYNC tables. Do not delete any rows for instances of utilities that are awaiting restart.
   - Delete invalid rows in the BMCUTIL table. Do not delete any rows for instances of utilities that are awaiting restart.

   Then use the following SQL statement to delete rows from the BMCSYNC table:

   ```sql
   DELETE
   FROM creatorName.CMN_BMCSYNC
   WHERE UTILID NOT IN
   (SELECT UTILID FROM creatorName.CMN_BMCUTIL);
   ```

   **Note**
   
   The names of the BMCUTIL and BMCSYNC tables might have been changed at your site during installation.

BMCXCOPY table

The BMC utilities use the BMCXCOPY table to track registered copies.

Table 114 on page 492 describes the contents of the BMCXCOPY table, which contains information about the following types of registered copies:

- Indexes that COPY PLUS has copied:
  - COPY NO index copies
  - DSNUM $n$ index (nonpartitioned) copies
  - Incremental index copies
  - Index copies that are made at data set level
Instant Snapshots made by COPY PLUS that are not registered as Flash Copies in SYSCOPY with the BMC EXTENDED BUFFER MANAGER (XBM) product or BMC SNAPSHOT UPGRADE FEATURE (SUF) technology, and any standard copies made in association with the Instant Snapshot

- Online consistent copies
- Cabinet copies
- Encrypted copies

The BMCXCOPY table functions like SYSIBM.SYSCOPY except that IXNAME replaces TSNAME in BMCXCOPY. You must control authorization and access to users for BMCXCOPY through standard DB2 authorization.

If you have more than one BMC utility installed, all of these utilities should share the same BMCXCOPY table.

Table 114: Contents of the BMCXCOPY table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td>Name of the database</td>
</tr>
<tr>
<td>IXNAME</td>
<td>CHAR(8)</td>
<td>Name of the index space or table space for Instant Snapshots and associated copies</td>
</tr>
<tr>
<td>DSNUM</td>
<td>INTEGER</td>
<td>Data set number within the index or table space</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| ICTYPE      | CHAR(1)   | Operation type:  
- F (COPY FULL YES; for COPY PLUS, online consistent copies)  
- I (COPY FULL NO)  
- W (REORG LOG NO)  
- B (REBUILD INDEX)  
- P (POINT-IN-TIME RECOVERY)  
- C (for COPY PLUS version 7.3 and earlier, online consistent copies)  
- s (used by COPY PLUS to track system pages)  
- m (indicates that the table space was exported by the COPY PLUS EXPORT command or migrated by the RECOVER PLUS IMPORT command) |
| ICDATE      | CHAR(6)   | Date of the entry (yyymmdd) |
| START_RBA   | VARCHAR(10) | The relative byte location of a point in the DB2 recovery log  
The indicated point as follows:  
- For ICTYPE F, the starting point for all updates since the image copy was taken  
- For COPY_TYPE O, the minimum of the consistent point and the oldest inflight URID  
- *(RECOVERY MANAGER)* For ICTYPE C, the consistent log point for the copy  
  - RBA for non-data-sharing systems  
  - LRSN for data sharing systems |
<p>| FILESEQNO   | INTEGER   | Tape file sequence number of the copy |
| DEVTYPE     | CHAR(8)   | Type of device on which the copy resides |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBMREQD</td>
<td>CHAR(1)</td>
<td>Whether the row came from the basic machine-readable material (MRM) tape:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (NO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (YES)</td>
</tr>
<tr>
<td>DSNAME</td>
<td>CHAR(44)</td>
<td>Name of the data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If STYPE V, DSNAME is the name of the VSAM data component.</td>
</tr>
<tr>
<td>ICTIME</td>
<td>CHAR(6)</td>
<td>Time at which this row was inserted (hhmmss)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The insertion takes place after the completion of the operation that the row represents.</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>CHAR(1)</td>
<td>SHRLEVEL parameter on COPY if ICTYPE F:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ C (change)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ R (reference)</td>
</tr>
<tr>
<td>DSVOLSER</td>
<td>VARCHAR(1784)</td>
<td>Volume serial numbers of the data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Commas separate items in a list of 6-byte numbers. This column is blank if the data set is cataloged.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Date and time when the row was inserted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This column contains the date and time that are recorded in ICDATE and ICTIME. The use of TIMESTAMP over ICDATE and ICTIME is recommended, because later DB2 releases might not support the latter two columns.</td>
</tr>
<tr>
<td>ICBACKUP</td>
<td>CHAR(2)</td>
<td>Type of image copy contained in the data set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LB (data set contains local backup data)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RP (data set contains recovery system main data)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RB (data set contains recovery system backup data)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (data set contains local system main data or is not one of multiple copies)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ICUNIT</td>
<td>CHAR(1)</td>
<td>Media on which the image copy data set is stored:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ D (DASD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ T (tape)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (medium is neither tape nor DASD)</td>
</tr>
<tr>
<td>STYPE</td>
<td>CHAR(1)</td>
<td>Type of copy:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (for ICTYPE=F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ V (Instant Snapshot or a VSAM data set)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ e (encrypted copy)</td>
</tr>
<tr>
<td>PIT_RBA</td>
<td>VARCHAR(10)</td>
<td>Point-in-time recovery:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ X'00000000000000' (for ICTYPE=F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Consistent point (for COPY_TYPE=O)</td>
</tr>
<tr>
<td>GROUP_MEMBER</td>
<td>CHAR(8)</td>
<td>Data-sharing group member (the name of the SSID where the copy was made)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This column is blank if you are not using data sharing.</td>
</tr>
<tr>
<td>OTYPE</td>
<td>CHAR(1)</td>
<td>Type of object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ T (table)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ I (index)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ i (compressed index)</td>
</tr>
<tr>
<td>LOWDSNUM</td>
<td>INTEGER</td>
<td>Not used</td>
</tr>
<tr>
<td>HIGHDSNUM</td>
<td>INTEGER</td>
<td>Not used</td>
</tr>
<tr>
<td>COPYPAGESF</td>
<td>FLOAT(53)</td>
<td>Number of pages written to the copy data set</td>
</tr>
<tr>
<td>NPAGESF</td>
<td>FLOAT(53)</td>
<td>High-used RBA divided by the page size</td>
</tr>
<tr>
<td>CPAGESF</td>
<td>FLOAT(53)</td>
<td>Total number of changed pages</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Job name</td>
</tr>
<tr>
<td>AUTHID</td>
<td>CHAR(8)</td>
<td>Authorization ID</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OLDEST_VERSION</td>
<td>SMALLINT</td>
<td>When ICTYPE= B, F, I, S, W, or X, the version number of the oldest format of data for an object For other values of ICTYPE, the value is -1.</td>
</tr>
<tr>
<td>LOGICAL_PART</td>
<td>INTEGER</td>
<td>Logical partition number</td>
</tr>
<tr>
<td>LOGGED</td>
<td>CHAR(1)</td>
<td>Logging attribute of the table space:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (logged)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (not logged)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (row inserted prior to DB2 version 9) For a non-LOB table space or index space, blank indicates that the logging attribute is logged.</td>
</tr>
<tr>
<td>TTYPE</td>
<td>CHAR(8)</td>
<td>Row format for the table space or partition:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RRF (reordered row format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ BRF (basic row format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RBA/LRSN format for the space or partition:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ B (basic 6-byte format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ E (extended 10-byte format)</td>
</tr>
<tr>
<td>INSTANCE</td>
<td>SMALLINT</td>
<td>Instance number of the current base objects (table and index) The default value is 1.</td>
</tr>
<tr>
<td>RELCREATED</td>
<td>CHAR(1)</td>
<td>DB2 release that created the object If the release is earlier than Version 9, the value is blank.</td>
</tr>
<tr>
<td>COPY_TYPE</td>
<td>CHAR(1)</td>
<td>Type of copy:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ C (cabinet copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ O (online consistent copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ X (export copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ I (import copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NOTE_VALUE</td>
<td>CHAR(4)</td>
<td>Encoded value that quickly locates data for a specific space in a cabinet copy The default value is blank.</td>
</tr>
<tr>
<td>NOTE_TYPE</td>
<td>CHAR(1)</td>
<td>Type of NOTE (issued by COPY PLUS):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ A (ABS - tape)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ R (REL - disk)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ F (frame)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
<tr>
<td>OCC_COPY_RBA</td>
<td>VARCHAR(10)</td>
<td>Original START_RBA of an online consistent copy The default value is blank.</td>
</tr>
<tr>
<td>OCC_LOCKRULE</td>
<td>CHAR(1)</td>
<td>Locking rule for a table space (not used for indexes):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ A (for page level)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ R (for row level)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
<tr>
<td>OCC_SPACE_ALTERED</td>
<td>CHAR(1)</td>
<td>Whether the space was altered:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (altered)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (not altered)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
<tr>
<td>CAB_BLOCKS</td>
<td>INTEGER</td>
<td>Total number of frames written for a cabinet copy</td>
</tr>
<tr>
<td>EXPSSID</td>
<td>VARCHAR(8)</td>
<td>Source location SSID of the migration file (valid with COPY_TYPE = I)</td>
</tr>
<tr>
<td>EXPSLRSN</td>
<td>VARCHAR(10)</td>
<td>Indicates the SYNC AUTO point on the source (valid with COPY_TYPE = I and COPY_TYPE = X)</td>
</tr>
<tr>
<td>EXPTLRSN</td>
<td>VARCHAR(10)</td>
<td>Indicates the SYNC AUTO point on the target (valid with COPY_TYPE = I)</td>
</tr>
</tbody>
</table>
Maintaining the BMCXCOPY table

Periodically, you should review BMCXCOPY and delete old rows to control its expansion.

To control expansion of the BMCXCOPY table

1. To delete all rows from the BMCXCOPY table that are older than 30 days, run an SQL DELETE statement, using the following statement as an example:

```
DELETE FROM creatorName.CMN_BMCXCOPY
WHERE DAYS(CURRENT_TIMESTAMP) - DAYS(TIMESTAMP) > 30;
```
Generating control statements for DB2 or other software products

This appendix describes the control cards that you can generate by using the CNTLCARDS option.

Overview of generating control statements in UNLOAD PLUS

The UNLOAD PLUS for DB2 product can generate customized control statements during unload processing to define the unloaded sequential files for direct use by LOADPLUS, DB2, or other software products.

You can use the CNTLCARDS command option to specify the type of control statements that UNLOAD PLUS generates. To get control statements during unload processing, you must specify a SYSCNTL DD statement in your JCL. UNLOAD PLUS writes these control statements to the SYSCNTL data set. (You can specify the CNTLDDN option on the UNLOAD command to rename the default ddname of the data set that contains the control statements.) All control statements that UNLOAD PLUS writes to the SYSCNTL data set are echoed in SYSPRINT.

The control statement generation feature of UNLOAD PLUS currently supports the following software products:

- DB2 Version 10 or later
- LOADPLUS version 10.1 or later
- Easytrieve version 5.3
- FOCUS version 6
- NOMAD version 5.5
- SAS version 5
Inserting additional command options into utility control statements

In addition to using the CNTLCARDS command option to specify the type of control statements that UNLOAD PLUS generates, you can insert additional command options into the generated utility control statements.

Use one of the following string parameters to insert these additional options:

- The string parameter that you can specify with the CNTLCARDS command option inserts additional command options after the first statement of the generated control cards.

- The string parameter that you can specify with the INTO command option inserts additional command options after the INTO statement of the generated control cards.

You can specify one or more string parameters for each command option of up to 72 bytes each. These string parameters are valid when you are using the following CNTLCARDS options:

- DB2LOAD
- DB2
- BMCLOAD

For more information about these options, see “CNTLCARDS” on page 130.

The following example uses the string parameter with the CNTLCARDS BMCLOAD and INTO command options. In this example, UNLOAD PLUS generates LOAD utility control statements in the SYSCNTL data set with additional command options inserted after the LOAD statement and after the INTO statement.

```
BMC50102I   UNLOAD INFILE BMCCOPY
BMC50102I   CNTLCARDS BMCLOAD 'RESUME YES LOG NO NOCOPYPEND'
BMC50102I   UNLOADDN(SYSREC)
BMC50102I   ORDER NO
BMC50102I   SELECT *
BMC50102I   INTO 'REPLACE WHEN EMPLDATE >''2005-01-01'''
BMC50102I   FROM RDGTMS.TB10PART
```
The resulting LOAD command is as follows:

```
BMC51810I LOAD DATA INDDN SYSREC
BMC51809I RESUME YES LOG NO NOCOPYPEND
BMC51940I      EBCDIC CCSID(37,65534,65534)
BMC51811I      INTO TABLE RDGTMSTB1PART
BMC51809I REPLACE WHEN DATE > '2005-01-01'
BMC51815I     (SEQ_NUMBER POSITION(1:2) CHAR (2)
BMC51815I     ,PART_NUMBER POSITION(3:12) CHAR (10)
BMC51813I     ,VAR_CHAR_FLD1 POSITION(13:*) VARCHAR
BMC51813I     ,VAR_CHAR_FLD2 POSITION(*:* )VARCHAR
BMC51809I )
```

### Control statements to create and load DB2 tables

The control statement generation feature of UNLOAD PLUS supports the following options for DB2 load processing:

- Loading a table that was previously created (DB2LOAD or BMCLOAD option)
- Creating a new DB2 table and load it (DB2 option)
- Creating a new DB2 table without loading it (DB2DDL option)

For information about inserting additional command options into these control statements, see “Inserting additional command options into utility control statements” on page 500.

### DB2LOAD option

The DB2LOAD option allows you to construct a job that can unload data and load it directly with no intervention.

If you specify CNTLCARDS DB2LOAD, UNLOAD PLUS generates a LOAD utility statement to load data from the unloaded sequential file into an existing DB2 table, using either the BMC LOADPLUS product or the IBM DB2 LOAD utility.

The following figure illustrates this option:
If you specify a table name in the INTO NAME clause, UNLOAD PLUS uses that table name in the LOAD statement. If you do not supply a table name, UNLOAD PLUS uses the name of the table from which it unloaded the data.

You can use the INTO list of columns to specify column names for the table that is being created. If you do not supply a list of columns, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.
You can specify a string following the INTO statement to supply additional command options for the LOAD statement. For example, you might want to specify certain partitions to replace:

```
INTO 'PART 2 REPLACE'
```

For information about specifying additional command options with the CNTLCARDS DB2LOAD option, see “Inserting additional command options into utility control statements” on page 500.

### Adding new fields

If you added any new constant fields during unload processing, UNLOAD PLUS uses the format $CONSTn to assign field names, where n is 1 for the first field that UNLOAD PLUS creates, and is incremented by one for each additional field that it creates.

Any NULL columns that UNLOAD PLUS processed while unloading create a field named NULLn following the nullable field, where n is 1 for the first field created and is incremented by one for each additional field created. If this field contains a question mark (?) character, its associated field is null. Because these field names exist, the LOAD statement cannot specify SKIPFIELDS NO.

### DB2 option

If you specify CNTLCARDS DB2, UNLOAD PLUS generates appropriate CREATE TABLE DDL and LOAD utility statements.

Before the data can be reloaded, the CREATE TABLE DDL and the LOAD utility statement must be processed separately as shown in Figure 82 on page 504.

UNLOAD PLUS generates a LOAD TABLE utility statement after processing CREATE TABLE DDL. The BMC LOADPLUS product or the IBM DB2 LOAD utility use this statement to load data from the unloaded sequential file into the DB2 table that the CREATE TABLE statement names.
Figure 82: Using the DB2 option

1. UNLOAD PLUS
   - Unloaded data
   - DDL and LOAD control statements

2. User intervention
   - SPUFI or other tool
     - DB2 catalog

3. LOADPLUS or DB2 LOAD utility
   - DB2 table

Control statements to create and load DB2 tables
**INTO option**

You can use the INTO NAME clause to specify the table name that UNLOAD PLUS uses in the CREATE statement. If you do not supply a table name, UNLOAD PLUS uses the name of the unloaded table.

You can use the INTO list of columns to specify column names for the table that is being created. If you do not supply a list of columns, or if you do not specify the INTO command option, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

**Note**

The column definition retains the original column type in the case of DATE, TIME, or TIMESTAMP columns. The LOAD utility statement for these types is DATE, TIME, or TIMESTAMP EXTERNAL.

You can specify a string following the INTO statement to supply additional command options with the LOAD statement. For example, you might want to specify certain partitions to replace:

```
INTO 'PART 2 REPLACE'
```

For information about specifying additional command options with the CNTLCARDS DB2 option, see “Inserting additional command options into utility control statements” on page 500.

**Adding new fields**

If you added any new constant fields during unload processing, UNLOAD PLUS assigns field names using the format $CONSTn, where $ is 1 for the first field created and is incremented by one for each additional field that is created.

**BMCLOAD option for LOADPLUS for DB2**

Use the BMCLOAD option to construct a job that generates control statements specifically for the BMC LOADPLUS for DB2 product.

If you specify CNTLCARDS BMCLOAD, UNLOAD PLUS generates a LOAD utility statement to load data from the unloaded data sets into an existing DB2 table, using LOADPLUS. You can specify a string to insert additional command options into the control statements of LOADPLUS. You can specify multiple strings of up to 72 bytes each.

The following figure illustrates this option:
Additional options

This section describes the interaction between CNTLCARDS BMCLOAD and other UNLOAD PLUS options.

FORMAT

When you specify FORMAT CSV or FORMAT BMCLOAD, you should specify CNTLCARDS BMCLOAD too. For FORMAT BMCLOAD, if you do not specify CNTLCARDS BMCLOAD, UNLOAD PLUS overrides what you specified and uses CNTLCARDS BMCLOAD.
AUTOTAG

When you specify CNTLCARDS BMCLOAD and the following conditions exist, specify AUTOTAG YES to have UNLOAD PLUS generate one LOAD statement with multiple INTO clauses:

- You are using a single SYSCNTL data set.
- You specify multiple SELECT statements on tables that are in the same table space.

**Note**
If you specify FORMAT BMCLOAD, you cannot also specify AUTOTAG YES.

INTO

If you specify a table name in the INTO NAME clause, UNLOAD PLUS uses that table name in the LOAD statement. If you do not supply a table name, UNLOAD PLUS uses the name of the table from which it unloaded the data.

You can use the INTO statement to specify column names for the table that is being created. If you do not supply a list of columns, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

You can specify a string following the INTO statement to supply additional command options to LOADPLUS. For example, you might want to specify certain partitions to replace:

```
INTO 'PART 2:4 REPLACE'
```

For information about specifying additional command options with the CNTLCARDS BMCLOAD option, see “Inserting additional command options into utility control statements” on page 500.

Adding new fields

If you added any new constant fields during unload processing, UNLOAD PLUS uses the format $CONSTn to assign field names, where $n is 1 for the first field that UNLOAD PLUS creates and is incremented by one for each additional created field.

Any NULL columns that UNLOAD PLUS processed while unloading create a field named NULL$n following the nullable field, where $n is 1 for the first field created and is incremented by one for each additional field created. If this field contains a question mark (?) character, its associated field is null. Because these field names exist, the LOAD statement cannot specify SKIPFIELDS NO.
Dynamic allocation

If you dynamically allocate your unload data sets, these data sets are not associated with a DD that allows the subsequent LOAD step to use traditional control cards. LOADPLUS allows you to specify actual data set names in load control cards. UNLOAD PLUS generates INDSN(DSN,...,DSN) in load control cards when you specify ACTIVE YES for the primary unload data sets and you specify CNTLCARDS BMCLOAD.

If the dynamically allocated unload data set is a generation data group (GDG), UNLOAD PLUS assigns the explicit cataloged data set name.

DB2DDL option

If you specify CNTLCARDS DB2DDL, UNLOAD PLUS generates appropriate CREATE TABLE DDL.

Figure 84 on page 508 illustrates unloading of the data and creation of the DDL. You can specify the NAME clause to specify the table name optionally in the CREATE statement. If you do not supply a table name, UNLOAD PLUS uses the name of the unloaded table.

You can use the INTO list of columns to specify column names for the table that is being created. If you do not supply a list of columns or if you do not specify the INTO command option, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table that is being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

Figure 84: Using the DB2DDL option
Control statements for FOCUS processing

If you specify CNTLCARDS FOCUS, UNLOAD PLUS generates an appropriate external file definition statement for the FOCUS product from Information Builders.

UNLOAD PLUS uses the ddname of the unload file data set as the name of the FOCUS external file, and the name of the DB2 table that was unloaded for the FOCUS segment name.

The following figure illustrates this process:

**Figure 85: Using the FOCUS option**
If you specify a list of unloaded column names for the INTO command option, UNLOAD PLUS uses the list of column names for the FOCUS field description names. If you do not specify the INTO command option, UNLOAD PLUS uses the column names of the columns from which the data were unloaded. The data types will match the unloaded column types unless they are converted during the unload. In the case of DATE, TIME, or TIMESTAMP columns, the field definition will indicate a character column type of the appropriate length.

In the FOCUS definition statements, the ACTUAL keyword is set to the real field type and length created in the output record. For numeric field types, the USAGE keyword is set to the ACTUAL data type, and the length is set to an appropriate length to be able to print the field. For character field types, the maximum field length for FOCUS is 256 bytes.

When field sizes exceed the FOCUS maximums, you must modify the generated control statements in whatever way necessary to read these columns. If a character column size exceeds the 256-byte FOCUS maximum, the size generated on the statement is the total size of the field, and a comment statement is generated, noting that the field on the next statement must be modified.

To conform to FOCUS field naming conventions, UNLOAD PLUS uses the following processing to modify the DB2 column names:

1. FOCUS field names are limited to 12 characters. When generating control statements for processing by FOCUS, UNLOAD PLUS does not truncate DB2 column names that are longer than 12 characters, but UNLOAD PLUS does issue a warning message. **Note** You must check for names that exceed the FOCUS limit of 12 characters, and correct them if necessary before executing FOCUS.

2. The first character of the field name is checked for the characters A through Z. If it is not one of these characters, UNLOAD PLUS converts it to 'Z'. If the first character is a DB2 delimiter character such as " or ' (in other words, the field name is enclosed in SQL escape characters), it is not converted.

3. The remaining characters of the field name are checked for the characters A through Z, the numbers 0 through 9, and special characters _, @, $, and #. UNLOAD PLUS converts any character that is not one of these characters to '_'. If the first character is a DB2 delimiter character, the remainder of the name is not checked for invalid characters.

4. No FOCUS ALIAS names are prepared.

Any new constant fields added by UNLOAD PLUS during unload processing are assigned names using the format ZCONSTn, where n is 1 for the first field created and is incremented by one for each additional field created.
Any NULL columns processed by UNLOAD PLUS during unload processing will create a field named NULLn following the nullable field, where n is 1 for the first field created and is incremented by one for each additional field created. If this field contains a question mark (?) character, its associated field is null. Alternatively, you can use the NULLIF option to insert a value when the field is null and thus remove the extra byte.

FOCUS normally cannot process DB2 VARCHAR fields. To facilitate using these field types, UNLOAD PLUS allows you to specify FIXEDVARCHAR YES to convert these fields to fixed-length fields. When you specify FIXEDVARCHAR YES, UNLOAD PLUS defines a 2-byte count field followed by the fixed-length field.

Control statements for SAS processing

If you specify CNTLCARDS SAS, UNLOAD PLUS generates appropriate external file INPUT statements for the SAS product.

These statements can be used to read the unloaded sequential file for processing by SAS to produce reports or for other purposes.

The following figure illustrates this process:
If you specify a list of unloaded column names for the INTO command option, UNLOAD PLUS uses the list of column names for the SAS field description names. If you do not specify the INTO command option, UNLOAD PLUS uses the column names of the columns from which the data were unloaded. The data types will match the unloaded column types unless they are converted during the unload. In the case of DATE, TIME or TIMESTAMP columns, the field definition will indicate a character column type of the appropriate length.

If your DB2 column widths are unloaded into fields that exceed the 200-byte SAS maximum, you must modify the control statements generated by UNLOAD PLUS in whatever way necessary to read these fields. If a DB2 column size exceeds the 200-byte SAS maximum, the sequential file field size generated on the SAS statement will be the large field size, and a warning comment is generated on that statement.
In order to conform to SAS naming conventions, UNLOAD PLUS uses the following processing for each column name unloaded:

1. SAS names are limited to 8 characters. When generating control statements for processing by SAS, UNLOAD PLUS does not truncate column names or the DB2 column name, but UNLOAD PLUS does issue a warning message.

   **Note**
   
   You must check for names that exceed the SAS limit of 8 characters, and correct them if necessary before executing SAS.

2. The first character of the name is checked for the character ‘_’ and the characters A through Z. If it is not one of these characters, UNLOAD PLUS converts it to ‘_’. If the first character is a DB2 delimiter character such as " or ’ (in other words, the field name is enclosed in SQL escape characters), it is not converted.

3. The remaining characters of the name are checked for the characters A through Z, the numbers 0 through 9, or special character ‘_’. UNLOAD PLUS converts any character that is not one of these characters to ‘_’. If the first character is a DB2 delimiter character, the remainder of the name is not checked for invalid characters.

Variable character fields are assigned a count field name of VARLEN \( n \), where \( n \) is incremented for each usage as shown in the following example:

```plaintext
VARLEN1 IB2. @; /* LENGTH OF FIELD SVARCHAR */
IF VARLEN1'¬= 0 THEN INPUT
SVARCHAR $VARYING16. VARLEN1 @;
INPUT                     /* RESUME INPUT */
```

Any new constant fields added by UNLOAD PLUS during unload processing are assigned names using the format $CONSTn, where \( n \) is 1 for the first field created and is incremented by one for each additional field created.

Any NULL columns processed by UNLOAD PLUS during unload processing create a field named NULL\( n \) following the nullable field, where \( n \) is 1 for the first field created and is incremented by one for each additional field created. If this field contains a question mark (?) character, its associated field is null. Alternatively, you can use the NULLIF option to insert a value when the field is null and thus remove the extra byte.
Control statements to create and load SQL/DS tables

The control statement generation feature of UNLOAD PLUS supports the following options for SQL/DS load processing:

- Loading a table that was previously created (SQL/DS-LOAD option)
- Creating a new SQL/DS table and load it (SQL/DS option)
- Creating a new SQL/DS table without loading it (SQL/DS-DDL option)

SQL/DS-LOAD option

The SQL/DS-LOAD option allows you to construct a job that can unload data and then load it with minimal intervention into an SQL/DS table.

The following figure illustrates this option:
If you specify CNTLCARDS SQL/DS-LOAD, UNLOAD PLUS generates a DATA LOAD utility statement. This statement can be used to load data from the unloaded sequential file into an existing SQL/DS table on a VM system using the SQL/DS DATA LOAD utility.

If you specify a table name in the INTO NAME clause, UNLOAD PLUS uses that table name in the DATA LOAD statement. If you do not supply a table name, UNLOAD PLUS uses the name of the table from which the data was unloaded.

You can use the INTO list of columns to specify column names for the table that is being created. If you do not supply a list of columns or if you do not specify the INTO command option, UNLOAD PLUS uses the column names of the unloaded
columns. The data types of the columns in the table that is being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

If you have added any new constant fields during unload processing, UNLOAD PLUS assigns field names using the format $CONSTn, where $ is 1 for the first field created and is incremented by one for each additional field created.

Any NULL columns processed by UNLOAD PLUS during unload processing create a field named NULLn following the nullable field, where $ is 1 for the first field created and is incremented by one for each additional field created. If this field contains a question mark (?) character, its associated field is null.

### SQL/DS option

If you specify CNTLCARDS SQL/DS, UNLOAD PLUS generates appropriate CREATE TABLE DDL and DATA LOAD utility statements.

Before the data can be reloaded, the CREATE TABLE DDL and the DATA LOAD utility statement must be processed separately as shown in Figure 88 on page 517.

You can optionally specify the table name used in the CREATE statement by specifying the INTO NAME clause. If you do not supply a table name, UNLOAD PLUS uses the name of the unloaded table.

You can use the INTO list of columns to specify column names for the table that is being created. If you do not supply a list of columns or if you do not specify the INTO command option, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table that is being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

In the case of DATE, TIME, or TIMESTAMP columns, the column definition retains the original column type. The DATA LOAD utility statement for these types will be DATE, TIME, or TIMESTAMP EXTERNAL.

If you have added any new constant fields during unload processing, UNLOAD PLUS assigns field names using the format $CONSTn, where $ is 1 for the first field created and is incremented by one for each additional field created.

Following the CREATE TABLE DDL, UNLOAD PLUS generates a DATA LOAD TABLE utility statement. This statement is used to load data from the unloaded sequential file into the SQL/DS table named in the CREATE TABLE statement by using the SQL/DS DATA LOAD utility.
Figure 88: Using the SQL/DS option

1. DB2 table → UNLOAD PLUS
2. Unloaded data → DDL and DATA LOAD control statements
3. MVS → VM
4. DDL → DATA LOAD control statements
5. SQL/DS → SQL/DS catalog
6. SQL/DS DATA LOAD utility → SQL/DS table

Step 1
Step 2
Step 3

User Intervention
**SQL/DS-DDL option**

If you specify CNTLCARDS SQL/DS DDL, UNLOAD PLUS generates appropriate CREATE TABLE DDL.

The following figure illustrates the unloading of the data and the creation of the DDL:

*Figure 89: Using the SQL/DS-DDL option*

You can optionally specify the table name used in the CREATE statement by specifying the NAME clause. If you do not supply a table name, UNLOAD PLUS uses the name of the unloaded table.

You can use the INTO list of columns to specify column names for the table that is being created. If you do not supply a list of columns or if you do not specify the INTO command option, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table that is being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

**Control statements for Easytrieve processing**

If you specify CNTLCARDS EASYTRIEVE, UNLOAD PLUS generates appropriate data definition statements for use with the CA Easytrieve product from CA Technologies.
The following figure illustrates this process:

**Figure 90: Using the EASYTRIEVE option**

![Diagram showing the process of using the EASYTRIEVE option.]

UNLOAD PLUS uses the ddname of the unload file data set as the name of the Easytrieve file on the FILE statement.

If you specify a list of unloaded column names with the INTO command option, UNLOAD PLUS uses the list of column names for the Easytrieve file field names in the DEFINE statement. If you do not specify the INTO command option, UNLOAD PLUS uses the column names of the columns from which the data was unloaded. The data types will match the unloaded column types unless they are converted during the unload. For DATE, TIME, and TIMESTAMP columns, the field definition will indicate a character column type of the appropriate length.
In the Easytrieve DEFINE statements, the data format is set to the actual field type and length created in the output record. For character field types, the maximum field length for Easytrieve is 32,767 bytes. If any numeric field sizes exceed the Easytrieve maximums, you must modify the generated control statements in whatever way necessary for Easytrieve to read these columns.

To conform to Easytrieve field naming conventions, UNLOAD PLUS uses the following processing to potentially modify the DB2 column names:

1. Because Easytrieve field names are 40 characters in length, the INTO clause field name (or the DB2 column name if INTO is not used) is used as the field name.

2. The first position of the field name is checked for the characters A through Z and numbers 0 through 9. If it is not one of these characters, UNLOAD PLUS converts it to 'Z'. In this case, a message is issued to indicate that a field name has changed. If the first character is a DB2 delimiter character such as " or ' (in other words, the field name is enclosed in SQL escape characters), it is not converted.

Any new constant fields added by UNLOAD PLUS during unload processing are assigned names using the format ZCONSTn, where n is 1 for the first constant field added and is incremented by one for each additional constant field added.

Any NULL columns processed by UNLOAD PLUS during unload processing create a field named NULLn following the nullable field, where n is 1 for the first field created and is incremented by one for each additional field created. If this field contains a '?' character, its associated field is null. Alternatively, you can use the NULLIF option to insert a value when the field is null and thus remove the extra byte.

Easytrieve normally cannot process DB2 VARCHAR fields. To facilitate using these field types, UNLOAD PLUS allows you to specify FIXEDVARCHAR YES, converting these fields to fields of fixed size. When you specify FIXEDVARCHAR YES, UNLOAD PLUS defines a 2-byte count field followed by the fixed-length field.

Control statements to load Teradata tables

UNLOAD PLUS provides the following options for Teradata control statement generation:

- TERADATA-BULK
- TERADATA-FAST
- TERADATA-MULT
- TERADATA
The first three options are used in conjunction with a corresponding Teradata utility. The last option, TERADATA, generates all of the control statements provided individually by the other options, allowing you to choose which statements and utility to use. Each option, along with example control statements, is discussed on the following pages.

You can optionally specify the table name used in the CREATE statement by specifying the INTO NAME clause. If you do not supply a table name, UNLOAD PLUS uses the name of the unloaded table.

You can use the INTO list of columns to specify column names for the table that is being created. If any DB2 delimiter or the table or column name is used, UNLOAD PLUS retains any delimited characters. If you do not supply a list of columns or if you do not specify the INTO command option, UNLOAD PLUS uses the column names of the unloaded columns. The data types of the columns in the table that is being created match the unloaded column types unless UNLOAD PLUS converts them during the unload.

For TIME or TIMESTAMP data:

- If you request that TIME or TIMESTAMP columns be unloaded in EXTERNAL format, UNLOAD PLUS defines the Teradata table columns as CHAR(n), where n is the length of the column.
- If you request that a TIME column be unloaded in INTERNAL format, UNLOAD PLUS defines the Teradata table column as BYTE(4).
- If you request that a TIMESTAMP column be unloaded in INTERNAL format, UNLOAD PLUS defines the Teradata table column as BYTE(10).

The column definition is included in the define section of the control cards, which defines input records to Teradata load programs.

For DATE data:

- If you request that a DATE column be unloaded in EXTERNAL format, UNLOAD PLUS defines the Teradata table column as DATE. However, you should ensure that the EXTERNAL format output of UNLOAD PLUS reflects Teradata’s DATE FORMAT. You can use the UNLOAD PLUS DATEFMT option to ensure that the formats are compatible. The definition section of the control cards defines DATE field input as CHAR(n), where n is the length of the column.
- If you request that a DATE column be unloaded in INTERNAL format, UNLOAD PLUS defines the Teradata table column as BYTE(4). This column definition is included in the define section of the control cards, which defines input records to Teradata load programs.
If you have added any new constant fields during unload processing, UNLOAD PLUS assigns field names using the format $CONSTn, where $n is 1 for the first field created and is incremented by one for each additional field created.

Any NULL columns processed by UNLOAD PLUS during unload processing create a field named NULLn following the nullable field, where $n is 1 for the first field created and is incremented by one for each additional field created. If this field contains a '?' character, its associated field is null. Alternatively, you can use the NULLIF option to insert a value when the field is null and thus remove the extra byte.

**TERADATA-BULK option**

TERADATA-BULK generates the control statements necessary to create a Teradata table and load a large amount of data into it using the Teradata Bulk Data Load utility.

The CREATE DDL statements (DBC/SQL) and the Bulk Data Load statements must be processed separately as shown in Figure 92 on page 524.

When the TERADATA-BULK option is specified, UNLOAD PLUS creates control statements similar to those in the following figure for use with the Bulk Data Load utility.

**Figure 91: Example control statements for Teradata Bulk Data Load utility**

```sql
CREATE TABLE tableName
( colName1 colType length) NOT NULL
, colName2 colType length)
);

LOGON/or USERID PASSWORD;
*
DEFINE
  colName1 (colType length)
, colName2 (colType length)
DDNAME = ddname;
*
INDICATORS OFF;
*

INSERT INTO tableName VALUES
( :colName1
, :colName2
);

LOGOFF;
```

When the unload has completed, edit the data set containing the control statements just produced, extract and modify the CREATE TABLE DDL, then use this DDL to define the table to Teradata.
When the table to be loaded is defined to Teradata, modify the LOGON statement as required by your installation, and use these statements as input to the Bulk Data Load utility.

When UNLOAD PLUS creates DDL for tables with more than 50 columns, ALTER statements are generated for columns 51 and higher. In cases where the Teradata limit of 256 columns is reached, all statements are generated and a warning message is issued.

Due to Teradata restrictions, when more than 50 columns are to be loaded, you must modify the control statements to process multiple utility executions.
Figure 92: Using the TERADATA-BULK option

1. **Step 1**: UNLOAD PLUS
   - DB2 table
   - Unloaded data

2. **User intervention**
   - Teradata control statements
   - Bulk Data Load statements
   - DBC/SQL DDL

3. **Step 2**: Teradata
   - Teradata Bulk Data Load

4. **Step 3**: Teradata table
**TERADATA-FAST option**

The TERADATA-FAST option generates the statements required to create a Teradata table and load it using the Teradata Fast Data Load utility.

The following figure illustrates this option:

**Figure 93: Using the TERADATA-FAST option**

![Diagram of TERADATA-FAST option](image)

Because this utility is always used to load a new table, the CREATE statement is imbedded before the DEFINE statement for the input data set.
When the TERADATA-FAST option is specified, UNLOAD PLUS creates control statements similar to those in the following figure for use with the Fast Data Load utility:

**Figure 94: Example control statements for Teradata Fast Data Load utility**

```sql
LOGON 0/USERID PASSWORD;
 *
 DROP TABLE tableName
 CREATE TABLE tableName
 ( colName1 colType length) NOT NULL
 , colName2 colType length);
 *
 BEGIN LOADING tableName;
 DEFINE
  colName1 (colType length)
 , colName2 (colType length)
 DDNAME = ddname;
 *
 INSERT INTO tableName VALUES
 ( :colName1
 , :colName2
 );
 END LOADING;
 LOGOFF;
```

After the unload has completed, edit the statements just produced and modify the LOGON statement as required by your installation. Determine if you want the table dropped, and, if so, remove the "*" from the DROP statement so that it will execute.

Next, review and modify the CREATE TABLE DDL, if necessary and then use these statements as input to the Fast Data Load utility.

When UNLOAD PLUS creates DDL for tables with more than 50 columns, ALTER statements are generated for columns 51 and higher. In cases where the Teradata limit of 256 columns is reached, all statements are generated and a warning message is issued.

Due to Teradata restrictions, when more than 50 columns are to be loaded, you must modify the control statements to process multiple utility executions.

**TERADATA-MULT option**

The TERADATA-MULT option generates the statements needed to create a Teradata table and load it using the Teradata Multiload utility.

The following figure illustrates this option:
When the TERADATA-MULT option is specified, UNLOAD PLUS creates control statements similar to those in the following figure to be used with the Multiload utility:

**Figure 96: Example control statements for the Teradata Multiload utility**

```
.LOGON 0/USERID PASSWORD:
/* */
CREATE TABLE tableName
  ( colName1 colType length) NOT NULL
  , colName2 colType length)
);

/* */
.LAYOUT ddname;
.FIELD colName1 inputPosition colType length);
```
After the unload has completed, edit the statements just produced and modify the LOGON statement as required by your installation.

Next, review and modify the CREATE TABLE DDL, if necessary and then use these statements as input to the Multload utility.

When UNLOAD PLUS creates DDL for tables with more than 50 columns, ALTER statements are generated for columns 51 and higher. In cases where the Teradata limit of 256 columns is reached, all statements are generated and a warning message is issued.

Due to Teradata restrictions, when more than 50 columns are to be loaded, you must modify the control statements to execute multiple utility executions.

**TERADATA option**

The TERADATA option generates all forms of the various CREATE and LOAD control statements provided in the individual TERADATA-BULK, -FAST and -MULT options.

You decide which statements to use with the appropriate Teradata utility. For information about these utilities, see the appropriate Teradata documentation from NCR.

**Control statements for NOMAD processing**

If you specify CNTLCARDS NOMAD, UNLOAD PLUS generates appropriate data definition statements for use with the NOMAD product from Select Business Solutions.

The following figure illustrates this process:
UNLOAD PLUS uses the ddname of the unload file data set as the name of the NOMAD file on the FILE statement.

If you specify a list of unloaded column names with the INTO command option, UNLOAD PLUS uses the list of column names for the NOMAD file field names in the DEFINE statement. If you do not specify the INTO command option, UNLOAD PLUS uses the column names of the columns from which the data was unloaded. The data types will match the unloaded column types unless they are converted during the unload. In the case of DATE, TIME, and TIMESTAMP columns, the field definition will indicate a character column type of the appropriate length.

In the NOMAD DEFINE statements, the data format is set to the actual field type and length created in the output record. For character field types, the maximum field
length for NOMAD is 32,767 bytes. If any numeric field sizes exceed the NOMAD maximums, you must modify the generated control statements so that NOMAD can read these columns.

To conform to NOMAD field naming conventions, UNLOAD PLUS uses the following process to potentially modify the DB2 column names:

1. Because NOMAD field names are 40 characters in length, the INTO clause field name (or the DB2 column name if INTO is not used) is used as the field name.

2. The first position of the field name is checked for the characters A through Z and numbers 0 through 9. If it is not one of these characters, UNLOAD PLUS converts it to 'Z'. In this case, a message is issued to indicate that a field name has changed. If the first character is a DB2 delimiter character such as " or ' (in other words, the field name is enclosed in SQL escape characters), it is not converted.

Any new constant fields added by UNLOAD PLUS during unload processing are assigned names using the format ZCONST$n$, where $n$ is 1 for the first constant field added and is incremented by one for each additional constant field added.

Any NULL columns processed by UNLOAD PLUS during unload processing create a field named NULL$n$ following the nullable field, where $n$ is 1 for the first field created and is incremented by one for each additional field created. If this field contains a '?' character, its associated field is null. Alternatively, you can use the NULLIF option to insert a value when the field is null and thus remove the extra byte.

NOMAD normally cannot process DB2 VARCHAR fields. To facilitate using these field types, UNLOAD PLUS allows you to specify FIXEDVARCHAR YES to convert these fields to fields of fixed size. When you specify FIXEDVARCHAR YES, UNLOAD PLUS defines a 2-byte count field followed by the fixed-length field.
UNLOAD PLUS user exits

This appendix describes user-written exit routines that you can use to customize your unload processing.

Overview of UNLOAD PLUS user exits

UNLOAD PLUS allows you to further customize your unload processing through user-written exit routines.

UNLOAD PLUS supports exit routines written in COBOL II, LE COBOL, SAS C, LE C, and Assembler. The exit routine is invoked when you supply the module name and program type with the UNLOADEXIT option of the UNLOAD command statement.

UNLOAD PLUS passes every output record to your routine before the record is written to the unload data set. The exit routine can inspect, modify, or discard records before UNLOAD PLUS writes them to the unload data set. If sorting is performed, the record is passed to the exit before the sort, but after the sort key is built.

Requirements and restrictions for UNLOAD PLUS user exits

UNLOAD PLUS user exits have the following requirements:

- The library in which this exit resides must be APF-authorized and included in your system’s LINKLIST or in your JOBLIB or STEPLIB.

- For COBOL II and C programs, the appropriate runtime libraries must be authorized and in your LINKLIST, JOBLIB, or STEPLIB.

- For LE COBOL and LE C programs, the appropriate IBM Language Environment runtime libraries must be authorized and in your LINKLIST, JOBLIB, or STEPLIB.
Your routine must be reentrant.

The following restrictions apply when using user-written exit routines with UNLOAD PLUS:

- If you specify FORMAT BMCLOAD, you cannot also use user-written exit routines. UNLOAD PLUS ignores the UNLOADEXIT option when you also specify FORMAT BMCLOAD.

- MAXSORTS is set to 1 when you specify LE COBOL, COBOL II, C, or LE C user exit routines.

- Do not use a user-written exit routine when unloading multiple tables.

- Do not use a user-written exit routine to modify the length of a record passed to the exit.

- UNLOAD PLUS passes 128-byte areas for the table creator and table name parameters to user-written exit routines. If you use an exit that depends on the table creator or table name that UNLOAD PLUS passes, ensure that your exit allows for 128-byte creator and table names.

Using sample UNLOAD PLUS user exits

Source code for the sample UNLOAD PLUS user exits is distributed in the UNLOAD PLUS HLQ.LLQSAMP library (where HLQ is the high-level qualifier specified during installation and LLQ is the low-level qualifier or prefix set during installation).

The following table lists the sample exit names and the command option to use to invoke the sample exit:

<table>
<thead>
<tr>
<th>User exit name</th>
<th>Description</th>
<th>UNLOAD command option to invoke</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADUEUEUE</td>
<td>Assembler sample exit</td>
<td>UNLOADEXIT ADUEUEUE ASM</td>
<td>“Sample Assembler user exit” on page 541</td>
</tr>
<tr>
<td>ADUEXIC</td>
<td>C (standard) sample exit</td>
<td>UNLOADEXIT ADUEXIC C</td>
<td>“Sample C user exit” on page 555</td>
</tr>
<tr>
<td>ADUEXIL</td>
<td>LE C sample exit</td>
<td>UNLOADEXIT ADUEXIL LE_C</td>
<td>“Sample LE C user exit” on page 571</td>
</tr>
<tr>
<td>ADUEXITC</td>
<td>LE COBOL sample exit</td>
<td>UNLOADEXIT ADUEXITC LE_COBOL</td>
<td>“COBOL II and LE COBOL user exits: sample 1” on page 586</td>
</tr>
</tbody>
</table>
User exit name | Description | UNLOAD command option to invoke | Reference
---|---|---|---
ADUEXTC2 | COBOL II sample exit | UNLOADEXIT ADUEXTC2 COBOL2 | “COBOL II and LE COBOL user exits: sample 2” on page 588

**To use the sample user exits**

1. Copy the appropriate member as the base code for your exit routine.
2. Compile, or assemble, and link your resulting exit into an authorized library so that it can be loaded during execution.

**Parameter data and pseudo-SQLDA**

Each type of user exit can access a parameter data structure that contains the following information:

- Address and miscellaneous data items that can be used in the exit
- Message areas for user messages
- An address for a pseudo-SQLDA control block that contains the data type of the column that UNLOAD PLUS passes to the exit.

This pseudo-SQLDA is similar to the IBM SQLDA. The fields of the pseudo-SQLDA are described in “UNLOAD PLUS pseudo-SQLDA” on page 534. For pseudo-SQLDA for each language, see the following references:

<table>
<thead>
<tr>
<th>Language</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler</td>
<td>“Pseudo-SQLDA DSECT” on page 539</td>
</tr>
<tr>
<td>C</td>
<td>“C pseudo-SQLDA structure” on page 553</td>
</tr>
<tr>
<td>COBOL II and LE COBOL</td>
<td>“COBOL pseudo-SQLDA record” on page 584</td>
</tr>
</tbody>
</table>

**Function codes and return codes for UNLOAD PLUS user exits**

UNLOAD PLUS communicates with your exit by passing a function code and parameters to the exit and receiving a return code from the exit.
For a C exit, the value of the function is the first element in the structure ADUEXITP (see “C exit parameter structure” on page 550). For a COBOL exit, the value of the function code is contained in the first field of the EXITPARMS record (see “Exit parameter record” on page 582). For an Assembler exit, register 0 contains a function code value and register 1 contains the address of an unload exit parameter block, ADUEXITP (see “Exit parameter block DSECT” on page 538).

On return from the exit, the return code instructs UNLOAD PLUS what to do. For an Assembler exit, the return code is the contents of register 15. For a COBOL exit, the return code is set in the RETURN-CODE variable. For C and LE C exits, the RETURN code is the value returned by the return function. The return code varies depending on the function code.

The following table describes the function codes and the appropriate return codes:

<table>
<thead>
<tr>
<th>Function code</th>
<th>Return code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 Write this record.</td>
</tr>
<tr>
<td></td>
<td>4 Discard this record.</td>
</tr>
<tr>
<td></td>
<td>Any other code terminates the utility.</td>
</tr>
<tr>
<td>1</td>
<td>0 Process this table with the exit.</td>
</tr>
<tr>
<td></td>
<td>4 Do not use the exit for this table.</td>
</tr>
<tr>
<td></td>
<td>Any other code terminates the utility.</td>
</tr>
<tr>
<td>2</td>
<td>Return codes are ignored.</td>
</tr>
</tbody>
</table>

“COBOL pseudo-SQLDA record” on page 584 illustrates the COBOL II and LE COBOL EXITPARM record relationships, and “Pseudo-SQLDA DSECT” on page 539 illustrates the Assembler parameter control block relationships.

**UNLOAD PLUS pseudo-SQLDA**

For all user exits, one of the parameters passed to the exit is an address or pointer for a pseudo-SQLDA structure.

The pseudo-SQLDA structure describes each table and the columns being unloaded. For pseudo-SQLDA for each language, see the following references:

<table>
<thead>
<tr>
<th>Language</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler</td>
<td>“Pseudo-SQLDA DSECT” on page 539</td>
</tr>
</tbody>
</table>
The pseudo-SQLDA structure consists of two portions: the SQLDA portion and the SQLVAR portion. The SQLDA portion of the pseudo-SQLDA contains the fields of the SQLDA. This portion appears once for each DB2 table being unloaded. Table 117 on page 535 summarizes the field names, data types, and usage notes for each field in the SQLDA portion of the structure.

The SQLVAR portion of the pseudo-SQLDA contains the fields of the SQLVAR. This portion is repeated for each column in the DB2 table being unloaded. Table 118 on page 536 summarizes the field names, data types, and usage notes for each field in this portion of the structure.

Within the SQLVAR portion of the pseudo-SQLDA is a field for the SQLTYPE. The value of this field depends on the type of data in the column and whether or not it allows null values. Table 119 on page 536 describes the valid SQLTYPE field values for each data type supported.

<table>
<thead>
<tr>
<th>Value</th>
<th>Data type</th>
<th>Indicator variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>384/385</td>
<td>Date</td>
<td>No/yes</td>
</tr>
<tr>
<td>388/389</td>
<td>Time</td>
<td>No/yes</td>
</tr>
<tr>
<td>392/393</td>
<td>Timestamp</td>
<td>No/yes</td>
</tr>
<tr>
<td>448/449</td>
<td>Variable-length character string</td>
<td>No/yes</td>
</tr>
<tr>
<td>452/453</td>
<td>Fixed-length character string</td>
<td>No/yes</td>
</tr>
<tr>
<td>456/457</td>
<td>Long variable length character string</td>
<td>No/yes</td>
</tr>
<tr>
<td>464/465</td>
<td>Variable-length graphic string</td>
<td>No/yes</td>
</tr>
<tr>
<td>468/469</td>
<td>Fixed-length graphic string</td>
<td>No/yes</td>
</tr>
<tr>
<td>472/473</td>
<td>Long variable length graphic string</td>
<td>No/yes</td>
</tr>
<tr>
<td>480/481</td>
<td>Floating-point</td>
<td>No/yes</td>
</tr>
<tr>
<td>484/485</td>
<td>Decimal</td>
<td>No/yes</td>
</tr>
<tr>
<td>496/497</td>
<td>Large integer</td>
<td>No/yes</td>
</tr>
<tr>
<td>500/501</td>
<td>Small integer</td>
<td>No/yes</td>
</tr>
</tbody>
</table>
### Table 118: Fields of the SQLVAR portion of the pseudo-SQLDA

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLDAID</td>
<td>CHAR(8)</td>
<td>&quot;Eye catcher&quot; for storage dumps containing 'SQLDAX'</td>
</tr>
<tr>
<td>SQLDABC</td>
<td>INTEGER</td>
<td>Length of the SQLDA, equal to SQLN*44+16</td>
</tr>
<tr>
<td>SQLN</td>
<td>SMALLINT</td>
<td>Number of occurrences of SQLVAR</td>
</tr>
<tr>
<td>SQLD</td>
<td>SMALLINT</td>
<td>Total number of columns described by occurrences of SQLVAR</td>
</tr>
</tbody>
</table>

### Table 119: Valid SQLTYPE field values for pseudo-SQLDA DSECT

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLTYPE</td>
<td>SMALLINT</td>
<td>For Initialization Function Code 1 and Processing Function Code 0, contains the data type of the column and whether it allows null values. See Table 117 on page 535 for a description of the data type codes.</td>
</tr>
<tr>
<td>SQLLEN</td>
<td>SMALLINT</td>
<td>For Initialization Function Code 1 and Processing Function Code 0, defines the external length of a value from the column as follows: Date: DB2 installation defined Time: DB2 installation defined Timestamp: 26 (bytes) Data Type: Content Character: Length attribute in bytes Graphic: Length attribute in double-byte characters Decimal: byte 1 = precision; byte 2 = scale Float: 4 (bytes) for single precision; 8 for double precision Smallint: 2 (bytes) Integer: 4 (bytes)</td>
</tr>
<tr>
<td>SQLDATA</td>
<td>Pointer</td>
<td>For Processing Function Code 0 only, contains the address of the data field in the record</td>
</tr>
<tr>
<td>SQLIND</td>
<td>Pointer</td>
<td>For Processing Function Code 0 only, contains the address of an associated indicator variable, if there is one; otherwise not used</td>
</tr>
<tr>
<td>SQLNAME</td>
<td>VARCHAR (30)</td>
<td>For Initialization Function Code 1 and Processing Function Code 0, contains the name of the column</td>
</tr>
</tbody>
</table>
The following topics provide a sample exit parameter block and exit that you can use to write a user exit in assembler:
Exit parameter block DSECT

This topic illustrates the Assembler exit parameter block DSECT.

Figure 99: Assembler exit parameter block

* ADUEXITP DSECT, PARMS PASSED TO EXIT
  XPFUNC DS F 0 = PROCESS, 1 = INIT, 2=TERMINATE
  XPSQLDA@ DS F A(SQLDA) FOR THIS TABLE
  XPTABLE@ DS F A(TABLE NAME BEING UNLOADED)
* THE ABOVE POINTS TO 128 BYTE CREATOR, FOLLOWED BY 128 BYTE NAME
  DS H RESERVED FOR UNLOAD PLUS
  XPREF# DS H BLOCK REFERENCE#
  XPSSID@ DS F A(SSID) 4 BYTES
  XPUSER@ DS F A(USERID) 8 BYTES
  XPUTID@ DS F A(UTILITY ID) 16 BYTES
  DS 6F RESERVED FOR UNLOAD PLUS
*
** USER AREA
* XPUSRMSG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT,
  IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED.
* XPUSRMSG@ POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT
  MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE
  PRINTED AFTER XPUSRMSG (IF ANY).
  FORMAT OF THE MESSAGE BUFFER AT THIS ADDRESS IS:
  #MSGS DS H NUMBER OF LINES, 0 = NO PRINT
  MSGSIZE DS H SIZE OF EACH LINE (MAX = 100)
  MSGTEXT DS CL(#MSGS*MSGSIZE) USER MESSAGES
  XPUSERW@ DS F USER WORK AREA ADDRESS
  XPUSERF1 DS F USER FIELD
  XPUSERF2 DS F USER FIELD
  XPUSERF3 DS F USER FIELD
  XPUSERF4 DS F USER FIELD
  XPUSERM@ DS F USER MESSAGE BUFFER ADDRESS
  XPUSERMS CL100 USER MESSAGE BUFFER TOTAL SIZE
  XPUSRM5 DS CL100 USER SINGLE MESSAGE AREA
  XPFLAGS DS X VARIOUS FLAGS FOR EXIT
  XPFDEBUG EQU X’01’ DEBUG IS ON
  DS X
*
** USER WORK SPACE
The following table explains the major DSECT fields and their uses:

**Table 120: Assembler user exit parameter block DSECT fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPSQLDA@</td>
<td>Address of the pseudo-SQLDA for this table as shown in “Pseudo-SQLDA DSECT” on page 539. For each table that UNLOAD PLUS unloads, this control block describes the column names, data types, and pointers to the column data within the output record being unloaded. This control block is identical in format to a DB2 SQLDA, and is used in much the same way.</td>
</tr>
<tr>
<td>XPTABLE@</td>
<td>Pointer to the name of the table being unloaded. This name field allows for a 128-byte creator name, followed by a 128-byte table name.</td>
</tr>
<tr>
<td>XPREF#</td>
<td>Exit reference number. Each table being unloaded gets its own copy of a user parameter block and an SQLDA. The XPREF# is incremented to provide a unique reference number.</td>
</tr>
<tr>
<td>XPSSID@</td>
<td>Pointer to a 4-byte field containing the SSID value</td>
</tr>
<tr>
<td>XPUSER@</td>
<td>Pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility</td>
</tr>
<tr>
<td>XPUTID@</td>
<td>Pointer to the 16-byte utility ID</td>
</tr>
<tr>
<td>XUSRMNG</td>
<td>A 100-byte message field to use when issuing a simple message. On return from any function call, if this field is nonblank, UNLOAD PLUS issues message BMC51635I identifying the exit name, exit reference number, and this message text. This field is blank after processing.</td>
</tr>
<tr>
<td>XUSERM@</td>
<td>Pointer to a user-defined message buffer that may contain multiple messages of equal length. Any messages to which this address points are printed after the XUSRMNG message. The format of the message buffer pointed to is</td>
</tr>
<tr>
<td>XPUSERM$</td>
<td>Must be set to the size of the user message buffer pointed to by XUSERM@</td>
</tr>
</tbody>
</table>

**Pseudo-SQLDA DSECT**

This topic illustrates the Assembler exit pseudo-SQLDA DSECT.
The pseudo-SQLDA DSECT consists of two portions:

- The first portion of the pseudo-SQLDA DSECT contains the fields of the SQLDA. This portion appears once for each DB2 table being unloaded.

- The second portion of the pseudo-SQLDA DSECT contains the fields of the SQLVAR. This portion will be repeated for each column in the DB2 table being unloaded.

  Within the SQLVAR portion of the pseudo-SQLDA DSECT is a field for the SQLTYPE. The value of this field depends on the type of data in the column and whether or not it allows null values.

The tables in “UNLOAD PLUS pseudo-SQLDA” on page 534 summarize the field information for this structure.

The following figure illustrates the Assembler parameter control block relationships:
Sample Assembler user exit

This topic provides a sample user exit that is written in Assembler.

**Figure 102: Sample Assembler user exit**

```assembly
ADUEUEUE TITLE 'ADUEUEUE - UNLOAD PLUS USER EXIT USER EXAMPLE - V13'
ADUEUEUE CSECT
ADUEUEUE AMODE 31
ADUEUEUE RMODE 24
**********************************************************************
*                         D I S C L A I M E R                        *
**********************************************************************
*  THIS IS A SAMPLE UNLOAD PLUS USER EXIT. THIS EXIT WOULD BE USED  *
```
IN ORDER TO INSPECT AND/OR MANIPULATE UNLOAD DATA RECORDS PRIOR TO THEIR BEING WRITTEN TO THE OUTPUT DATASET.

THIS EXIT WILL ONLY BE INVOKED WHEN IT IS SPECIFICALLY NAMED IN THE 'UNLOADEXIT --NAME--' PARAMETER.

NOTE: PLEASE REVIEW THE DOCUMENTATION IN THE REFERENCE MANUAL, AND THE FOLLOWING USAGE NOTES PRIOR TO IMPLEMENTING THIS EXIT.

PLEASE CALL BMC PRODUCT SUPPORT WITH ANY QUESTIONS YOU MAY HAVE IN THIS AREA.

PHONE: 1-800-537-1813

**********************************************************************
SPACE
**********************************************************************

* ADUEUEUE IS CALLED AT 3 POINTS IN PROCESSING A TABLE UNLOAD.
* WHEN INVOKED, R0 CONTAINS A FUNCTION CODE.
* R1 CONTAINS THE ADDRESS OF A USER EXIT BLOCK DESCRIBED BY THE
  ADUEXIT P DSECT.  FIELDS FOR YOUR USE IN THIS BLOCK ARE DESCRIBED
  IN THE DSECT.  THE USER EXIT BLOCK IS UNIQUE FOR EACH TASK INVOKING
  THE EXIT AND IS NOT SHARABLE BY MULTIPLE TASKS.
* THE MAIN FIELD OF INTEREST IS THE PSEUDO-SQLDA POINTER XPSQLDA@
  WHICH POINTS TO AN "SQLDA" PREPARED FOR EACH SELECTED UNLOAD TABLE.
  THIS SQLDA HAS THE SAME FORMAT AND CONTENTS OF A DB2 SQLDA, WITH THE
  MAIN EXCEPTION BEING THAT THE DESCRIPTIONS OF THE DATA FIELDS ARE
  THE OUTPUT DESCRIPTIONS (AFTER CONVERSIONS IF ANY), AND THE DATA
  POINTERS ARE POINTING TO THE OUTPUT RECORD OFFSETS IN THE RECORD
  ABOUT TO BE WRITTEN (FUNCTION CALL 0 ONLY).
* FUNCTION CODES:

  X'00' = PROCESS RECORD
  THIS CALL IS MADE AFTER A RECORD IS PREPARED FOR WRITING.
  ALL FIELDS ARE CONVERTED READY FOR OUTPUT.  THE SQLDA
  PROVIDES THE FIELD TYPES AND RECORD POSITIONS OF THE DATA.
  THE UNLOAD EXIT PARM BLOCK PASSED WITH THIS FUNCTION IS
  UNIQUE FOR EACH UNLOAD TASK INVOKING THE EXIT AND IS NOT
  SHARABLE AMONG MULTIPLE TASKS.
  R15 RETURN CODES FROM PROCESS:
  0 = ACCEPT THIS RECORD
  4 = DISCARD THIS RECORD
  ANYTHING ELSE = TERMINATE THE RUN.

  X'01' = INITIALIZATION CALL
  THIS CALL IS MADE DURING TABLE UNLOAD INITIALIZATION.  IT
  HAPPENS ONCE PER TABLE TO BE UNLOADED TO ALLOW SELECTION OF
  WHETHER OR NOT TO PROCESS THE TABLE WITH THE EXIT.
  R15 RETURN CODES FROM INITIALIZE:
  0 = ACTIVATE EXIT FOR RECORDS FROM THIS TABLE
  4 = DON'T USE THE EXIT FOR THIS TABLE.
  ANYTHING ELSE = TERMINATE THE RUN.
X'02' = TERMINATION/CLEANUP CALL

THIS CALL IS MADE JUST PRIOR TO TERMINATION OF THE UNLOAD TO ALLOW YOU TO PERFORM ANY CLEANUP FUNCTIONS NECESSARY.

RETURN CODES FOR TERMINATE ARE IGNORED.

FOR ANY EXIT FUNCTION, THE EXIT MAY INSERT MESSAGE TEXT IN THE MESSAGE AREA FOR PRINTING UPON RETURN.

ANOTHER AREA OF INTEREST IS THE COMMON AREA DATA BLOCK.  ITS STORAGE IS GETMAIN'D DURING THE INITIALIZATION CALL AND IS SHARABLE BY EACH TASK INVOKING THE EXIT.  AS A RESULT, A LOCKING MECHANISM MUST BE USED TO SERIALIZE PROCESSING(FUNCTION CALL 0 ONLY) WITH THE LOCK BEING RELEASED AFTER EACH RECORD IS PROCESSED.  THIS COMMON AREA ALLOWS OUTPUT OF SELECTED RECORDS TO A DATASET WRITTEN BY THE EXIT. USAGE OF AN AREA LIKE THIS IS NOT REQUIRED UNLESS MODIFICATION OF DATA, OUTPUT TO A DATASET, ETC. IS PLANNED FOR THE EXIT AND A MULTITASKING ENVIRONMENT IS IN EFFECT.

**********************************************************************
EJECT**********************************************************************

* INTERNAL REGISTER USAGE
**********************************************************************

* 0 - ON ENTRY TO MODULE = FUNCTION CODE = COPIED TO R2
* 1 - ON ENTRY TO MODULE = A(ADUEXITP BLOCK) COPIED TO R10
* 2 - FUNCTION CODE HOLDER UNTIL BRANCH CODE
* 3 -
* 4 - A(COMMON AREA) GETMAIN AREA
* 5 - A(DCB) USED FOR OUTPUT OF SELECTED RECORDS BY THE EXIT
* 6 -
* 7 -
* 8 -
* 9 -
* 10 - A(ADUEXITP DSECT BLOCK)
* 11 - A(WORKAREA) CONTAINING THE SAVE AREA, TABLE CREATOR AND TABLE NAME.  THIS AREA IS GETMAIN'D ON EACH ENTRY AND FREEMAIN'D ON EACH EXIT TO ALLOW EXIT INVOCATION IN A MULTITASKING ENVIRONMENT.
* 12 - BASEREG
* 13 - SAVEAREA
* 14 -
* 15 -

**********************************************************************

R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15

Appendix D  UNLOAD PLUS user exits 543
* ADUXITP DEFINES THE UNLOAD USER EXIT PARM BLOCK
* YOU MAY NOT MODIFY THE FIELDS IN FRONT OF THE USER AREA
* NOTE THAT THE UNLOAD USER EXIT PARM BLOCK IS UNIQUE FOR EACH
* INVOCATION OF THE EXIT. ANY ADDRESSES STORED INTO/MODIFICATIONS
* MADE TO THE USER PORTION OF THIS Control BLOCK ARE NOT SHARABLE
* IN A MULTITASKING ENVIRONMENT.

** USER AREA

* XPUSRMSG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT,
  IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED.
* XPUSRMS@ POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT
  MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE
  PRINTED AFTER XPUSRMSG (IF ANY).
* FORMAT OF THE MESSAGE BUFFER AT THIS ADDRESS IS:
  #MSGS DS H NUMBER OF LINES, 0 = NO PRINT
  MSGSIZE DS H SIZE OF EACH LINE (MAX = 100)
  MSGTEXT DS CL(#MSGS*MSGSIZE) USER MESSAGES

** USER WORK SPACE

* XPUSER DS 0F USER WORK AREA
** MACRO GENERATED SQLDA AREA SAME AS EXEC SQL INCLUDE SQLDA
* THE SQLDA AREA IS ALSO UNIQUE FOR EACH INVOCATION OF THE EXIT.
* ANY MODIFICATIONS MADE TO DATA IN THIS AREA ARE NOT SHARABLE
* AMONG TASKS IN A MULTITASKING ENVIRONMENT.

SQLDA DSECT
SQLDAID DS CL8 ID
SQLDABC DS F BYTE COUNT
**SQLN**  DS    H                   TOTAL VARS
**SQLD**  DS    H                   PERTINENT VARS
**SQLVAR**  DS    0F                  BEGIN VARS
**SQLDSIZ**  EQU   *-SQLDA             SIZE OF FIXED PART
**SQLVARN**  DSECT ,                   NTH VARIABLE
**SQLTYPE**  DS    H                   TYPE CODE
**SOLLLEN**  DS    0H                  NAME LENGTH
**SOLPVERS**  DS    X                   DEC PRECISION
**SOLCSNEG**  DS    X                   DEC SCALE
**SOLDATA**  DS    A                   ADDR OF DATA IN OUTPUT RECORD
**SOLIND**  DS    A                   ADDR OF NULL FIELD BYTE (? IF NULL)
**SQLNAME**  DS    H,CL30              DESCRIBE NAME
**ORG**  SQLNAME
**SOLNAM$**  DS    H                   SIZE OF COL NAME
**SOLNAM**  DS    CL30                NAME ALONE
**SOLVSIZ**  EQU   *-SQLVARN

**********************************************************************
* LOCAL PROGRAM WORK AREA DSECT - UNIQUE AREA THAT IS NOT SHARABLE
* DURING MULTIPLE INVOCATIONS (MULTITASKING ENVIRONMENT) -
* POINTED TO BY R11
**********************************************************************
**WORKAREA**  DSECT ,
**SAVE**  DS    18F                LOCAL SAVE AREA
**TBCREAT**  DS    CL128              TABLE CREATOR
**TNAMES**  DS    CL128              TABLE NAME
**SPARES**  DS    XL926
**WRKAREA$**  EQU   *-WORKAREA

**********************************************************************
* COMMON PROGRAM WORK AREA DSECT - SHARED AREA WHOSE USE MUST BE
* SERIALIZED DURING MULTIPLE INVOCATIONS (MULTITASKING ENVIRONMENT)
* THIS AREA IS GETMAIN'D DURING THE INITIALIZATION CALL. ITS
* ADDRESS IS STORED IN THE XPUSERW@ FIELD OF ADUEXITP AND IS COPIED
* TO EACH ADUEXITP BLOCK BY UNLOAD PLUS PRIOR TO ANY PROCESS CALLS.
* THIS AREA IS POINTED TO BY R4.
**********************************************************************
**COMNAREA**  DSECT ,
**COMNLOKW**  DS    F                  COMMON AREA LOCKWORD
**COMN#FRE**  EQU   0                  ..COMMON AREA UNLOCKED
**COMN#HLD**  EQU   1                  ..COMMON AREA LOCKED
*  
**COMNREC#**  DS    F                  OUTPUT RECORD COUNT
**COMNDBLW**  DS    D                  DOUBLE WORD
**COMNUNPK**  DS    CL16               UNPACK AREA
**COMNDATE**  DS    CL10               DATE HOLDING AREA
*  
**COMN#FND**  DS    X'80'              ..GAMES_BEHIND COLUMN FOUND
**COMN#KEP**  DS    X'40'              ..PROCESS/DONT WRITE RECORD
**COMN#DIS**  DS    X'20'              ..DONT PROCESS/WRITE RECORD
**COMN#OFF**  DS    X'00'              ..FLAG RESET
**DS**    XL3                SPARE
**DS**    4F                 SPARE
*  
**COMNMSGA**  DS    0F                 PROCESSING MESSAGE AREA
**COMNMSG1**  DS    CL28               MESSAGE AREA 1
**COMNMSG2**  DS    CL18               MESSAGE AREA 2
**COMNMSG3**  DS    CL9                MESSAGE AREA 3
**COMNMSG4**  DS    CL45               MESSAGE AREA PAD
**COMNMSG5**  DS    CL80               DISCARD RECORD AREA

Appendix D  UNLOAD PLUS user exits 545
COMNOPNL OPEN (0),MF=L           LIST FORM OF OPEN
    DS    4F                       SPARE
*
COMNCLS CLS (0),MF=L           LIST FORM OF CLOSE
    DS    4F                       SPARE
*
COMNODCB DCB DSORG=PS,MACRF=PM,DDNAME=WHATEVER
    DS    0F                       FULLWORD ALIGN
COMNODC$ EQU *-COMNODCB           DCB LENGTH
COMNAREL EQU *-COMNAREA           COMMON AREA LENGTH
EJECT
PRINT NOGEN
DCBD  DSORG=(PS,PO),DEVD=(DA,TA)
PRINT GEN
EJECT
**********************************************************************
* PROGRAM START
**********************************************************************
ADUEUEUE CSECT
    LA    R15,4     >>>>>    4 = DONT PROCESS CURRENT TABLE <<<<<
    BSM   0,R14     >>>>>    JUST RETURN TO CALLER          <<<<<
*
    STM   R14,R12,12(R13)          SAVE CALLERS REGS
    LR    R12,R15                  R12 IS MY BASEREG
    USING  ADUEUEUE,R12             ESTABLISH ADDRESSABLITY
*
    LR    R2,R0                    SAVE FUNCTION CODE
    LR    R10,R1                   GET A(PASSED BLOCK)
    USING  ADUEXITP,R10             MAP IT
*
    GETMAIN RC,LV=WRKAREA$,LOC=BELOW  GET WORK AREA
    LTR   R15,R15                  OK?
    BNZ   INIT9900                 ..NO, ERROR
    LR    R11,R1                   R11 = A(USER WORK AREA)
    USING WORKAREA,R11             ADDRESS WORKAREA DSECT
    LA    R15,SAVE                 MY SAVE AREA@
    ST    R15,8(,R13)              SAVE IN CALLERS SAVE AREA@ IN MINE
    LR    R13,R15                  SET OURS CURRENT
    EJECT
*
**********************************************************************
* CHECK THE FUNCTION CODE AND BRANCH TO PROCESSING ROUTINE
**********************************************************************
CHKF0000 DS    0H
    L     R15,XPUTID@              GET A(UTILITY ID)
    CLC   =C'DEBUG',0(R15)         DOES USER WANT DEBUG MESSAGES?
    BNE   INIT0010                 ..NO, SKIP MESSAGE PROCESSING
    OI    XPFLAGS,XPFDEBUG         ..YES, SET DEBUG FLAG
*
INIT0010 DS    0H
    L     R15,XPUTID@              GET A(UTILITY ID)
    CLC   =C'DEBUG',0(R15)         DOES USER WANT DEBUG MESSAGES?
    BNE   INIT0010                 ..NO, SKIP MESSAGE PROCESSING
    OI    XPFLAGS,XPFDEBUG         ..YES, SET DEBUG FLAG
*
L R3,XPTABLE@ GET A(TABLE NAME BEING UNLOADED)
MVC TBCREAT,O(R3) COPY TABLE CREATOR
MVC TBNNAME,128(R3) COPY TABLE NAME
CLC TBNNAME,BBALLTBL CORRECT TABLE?
BNE RETC0004 ..NO, DONT PROCESS

GETMAIN RC, LV=COMNAREL, GET COMMON AREA FOR IO
    LOC=BELOW BELOW 16M
LTR R15,R15 OK?
BNZ INIT9900 ..NO, ISSUE ERROR
ST R1,XPUSERW@ SAVE GETMAIN ADDRESS
LR R4,R1 SET UP COMNAREA DSECT
USING COMNAREA,R4 ADDRESSABILITY

MVI COMNAREA,X'00' INITIALIZE GETMAIN AREA
L R1,=A(COMNAREL-1) LENGTH OF COMMON AREA-1
LA R0,COMNAREA+1 TARGET @ FOR MVCL
LA R14,COMNAREA FROM @ FOR MVCL
LA R15,1 SET LENGTH = 1
MVCL R0,R14 PERFORM MOVE
MVC COMNODCB(COMNODC$),OUTDCB INITIALIZE COMMON OUTPUT DCB
LA R5,OUTDCB OUTPUT DCB
USING IHADCB,R5 SET DCB ADDRESSABILITY
MVI COMNOPNL,X'80' SET END OF LIST

OPEN ((5),(OUTPUT)), OPEN OUTPUT DATASET
    MF=(E,COMNOPNL) MAKE IT REENTRANT

TM DCBOFLGS,DCBOFOPN SUCCESSFUL?
BNO INIT9910 ..NO, ISSUE ERROR

TM XPFLAGS,XPFDEBUG IS DEBUG ON?
BNO RETC0000 ..NO, NO MESSAGES

MVC XPUSRMSG(L'IGOT2INI),IGOT2INI SAY I GOT HERE
MVC XPUSRMSG+L'IGOT2INI(18),0(R3) I GOT THIS TABLE
B RETC0000 PROCESS THIS TABLE

* ERROR CONDITIONS

INIT9900 DS 0H GETMAIN ERROR
MVC XPUSRMSG(L'MGETMERR),MGETMERR
B RETC0008 TERMINATE THE EXECUTION

* ERROR CONDITIONS

INIT9910 DS 0H OPEN ERROR
MVC XPUSRMSG(L'MOPENERR),MOPENERR
B RETC0008 TERMINATE THE EXECUTION

EJECT

**********************************************************************
* PROCESS A RECORD PRIOR TO WRITE
* THIS SECTION INTERROGATES A RECORD CONTAINING TEAM STANDINGS IN
* MAJOR LEAGUE BASEBALL. RECORDS FOR TEAMS THAT WERE 10 OR MORE
* GAMES BEHIND PRIOR TO JULY 1, 1994 ARE WRITTEN TO A SEPARATE
* DATASET(SYSEXIT) BY THE EXIT INSTEAD OF BEING WRITTEN TO THE
* UNLOAD DATASET(SYSREC) BY UNLOADPLUS.
**********************************************************************

PROC0000 DS 0H
ICM R4,B'1111',XPUSERW@ GET A(COMMON AREA)
BZ PROC9900 NO COMMON AREA - ERROR

PROC0010 DS 0H
LA R14,COMN#FRE SERIALIZE ON COMMON AREA
LA R15,COMN#HLD
CS R14,R15,COMNLOKW ATTEMPT LOCK
BE PROC0020 GOT IT, CONTINUE
CALLDISP , RELEASE CPU
B PROC0010 TRY AGAIN

Appendix D  UNLOAD PLUS user exits 547
PROC0020 DS OH
  L R8,XPSQOLDA@ GET SQLDA ADDRESS
  USING SOLDA,R8
  LH R9,SOLN GET NUMBER OF COLUMNS IN RECORD
  LA R8,SOLDSIZ(R8) PASS FIXED AREA
  USING SOLVARN.R8

PROC0030 DS OH
  COLC SOLNAM,DTCLNAME DATE COLUMN?
  BNE PROC0040 ..NO. CHECK FOR GAMES BEHIND
  L R7,SQLDATA GET A(DATE FIELD)
  MVC COMNRECA(R8),0(R7) SAVE ENTIRE RECORD
  MVC COMNDATE,0(R7) SAVE DATE VALUE
  B PROC0050 NOW FIND GAMES BEHIND FIELD

PROC0040 DS OH
  COLC SOLNAM,GBCLNAME GAMES BEHIND DATA?
  BNE PROC0050 ..NO, CHECK NEXT FIELD
  OI COMNFLAG,COMN#FND INDICATE COLUMN FOUND
  L R7,SQLDATA GET A(GAMES BEHIND FIELD)
  CLC 0(5,R7),=C' 10.0' 10 OR MORE GAMES BEHIND?
  BL PROC0060 ..NO, PROCESS/DONT WRITE
  CLC COMNDATE,DATECHEK ..YES, PRIOR JULY 1, 1994?
  BL PROC0070 ..YES, DONT PROCESS/WRITE
  B PROC0060 ..NO, PROCESS/DONT WRITE

PROC0050 DS OH
  LA R8,SQLVSIZ(R8) ..NO, POINT TO NEXT FIELD
  BCT R9,PROC0030 LOOP
  B PROC9900 GAMES BEHIND NOT FOUND - FATAL

PROC0060 DS OH
  OI COMNFLAG,COMN#KEP SET KEEP FLAG
  B PROC9000 GO FINISH UP

PROC0070 DS OH
  OI COMNFLAG,COMN#DIS SET DISCARD FLAG
  LA R5,COMNODCB GET A(OUTPUT DCB)
  LA R7,COMNRECA GET A(OUTPUT RECORD)
  LA R14,*+6 SET AMODE=24
  BSM 0,R14
  PUT (5),(7) WRITE RECORD TO SYSEXIT
  LA R14,*+10 SET AMODE=31
  O R14.=X'80000000'
  BSM 0,R14

PROC9000 DS OH
  L R1,COMNREC# GET RECORD COUNT
  LA R1,1(R1) INCREMENT
  ST R1,COMNREC# SAVE NEW RECORD COUNT
  TM XPFILS,XPDEBUG IS DEBUG ON?
  BNO PROC9050 ..NO, SKIP MESSAGE PROCESSING
  CVD R1,COMNDBLW CONVERT RECORD COUNT TO DECIMAL
  MVI COMMUNPK,X'40' BLANK 1ST BYTE OF UNPK AREA
  MVC COMMUNPK+1(L'COMMUNPK-1),COMMUNPK AND PROPAGATE
  MVI COMMUNMSA,X'40' BLANK 1ST BYTE OF MSGAREA
  MVC COMMUNMSA+1(COMMUNMSA+1),COMMUNMSA AND PROPAGATE
  UNPK COMMUNPK,COMNDBLW UNPACK RECORD COUNT
  OI COMMUNPK+15,X'FO' TURN OFF SIGN
  MVC COMMUNMSA1(COMMUNMSA1),IGOT2PRO MOVE MESSAGE TEXT
MVC COMNMSG2(COMNMS2$-2),COMNUNPK MOVE RECORD NUMBER
TM COMNFLAG,COMN#KEP KEEP THIS RECORD?
BNO PROC9010 ..NO, CHECK FOR DISCARD
MVC COMNMSG3(COMNMS3$),ACCEPT .YES, SIGNAL ACCEPTED
* PROC9010 DS OH
TM COMNFLAG,COMN#DIS DISCARD THIS ONE?
BNO PROC9020 ..NO, MOVE COMPLETE MESSAGE
MVC COMNMSG3(COMNMS3$),DISCARD ..YES, SIGNAL ACCEPTED
* PROC9020 DS OH
MVC XPUSRMSG(COMNMSG$),COMNMSGA MOVE ENTIRE MESSAGE
* PROC9050 DS OH
TM COMNFLAG,COMN#KEP LET UNLOAD PROCESS THIS ONE??
BO RETC0000 ..YES
TM COMNFLAG,COMN#DIS ..NO, VERIFY DISCARD
BO RETC0004 DISCARD, DONT PROCESS
B RETC0000 DONT KNOW, PROCESS IN UNLOAD
* ERROR CONDITION
* PROC9900 DS OH
MVC XPUSRMSG(L'MFATLERR),MFATLERR SOMETHING IS WRONG
B RETC0008 TERMINATE PROCESSING
EJECT
**********************************************************************
* SET RETURN CODE AND EXIT POINTS
**********************************************************************
* RETC0000 DS OH
LA R3,0 SAVE RETURN CODE
B RET9000 NORMAL PROCESSING
* RETC0004 DS OH
LA R3,4 SAVE RETURN CODE
B RET9000 DONT PROCESS/DISCARD
* RETC0008 DS OH
LA R3,8 SAVE RETURN CODE
B RET9000 TERMINATE UNLOAD+ EXEC
RET9000 DS OH
ICM R1,B'1111',XPUSERW@ GET A(COMMON AREA)
BZ RET9010 NO COMMON, SKIP RESETS
MVC COMNFLAG,=A(COMN#OFF) RESET FLAGS
MVC COMNLOKW,=A(COMN#FRE) RELEASE LOCK ON COMMON AREA
* RET9010 DS OH
L R13,SAVE+4 FREEMAIN RC,LV=WRKAREA$,A=(R11) FREE LOCAL WORK AREA
LR R15,R3 RESTORE RETURN CODE
L R14,12(.R13) RETURN
LM R0,R12,20(R13)
BSM 0,R14 EJECT
**********************************************************************
* CLEANUP (FUNCTION CALL = 2)
**********************************************************************
CLEANO00 DS OH
TM XPFLAGS,XPFDEBUG IS DEBUG ON?
BNO CLEANO10 ..NO, SKIP MESSAGE
MVC XPUSRMSG(L'IGOT2CLN),IGOT2CLN SAY I GOT HERE
* CLEANO10 DS OH
ICM R4,B'1111',XPUSERW@ GET A(COMMON AREA)
BZ CLEANO20 NO @, NO CLOSE OR FREEMAIN

Appendix D  UNLOAD PLUS Assembler user exits 549
UNLOAD PLUS C user exit

The following topics provide a sample exit parameter structure and exit that you can use to write a user exit in C:

- “C exit parameter structure” on page 550
- “C pseudo-SQLDA structure” on page 553
- “Sample C user exit” on page 555

C exit parameter structure

This topic illustrates the C exit parameter structure.
The ADUEXITP structure is contained in a header file, aduxith0.h.

**Figure 103: C user exit parameter structure**

```
struct ADUEXITP
{
    /********************************************************************
    /* ADUEXITP DEFINES THE UNLOAD USER EXIT PARM BLOCK              */
    /* YOU MAY NOT MODIFY THE FIELDS IN FRONT OF THE USER AREA        */
    /* (ACTUALLY YOU MAY, BUT YOU WON'T LIKE WHAT HAPPENS)            */
    /********************************************************************
    int xpfunc; /* 0 = PROCESS, 1 = INIT, 2=TERMINATE */
    void *xpsqldaA_; /* A(SQLDA) FOR THIS TABLE */
    void *xptableA_; /* A(TABLE NAME BEING UNLOADED) */
    /* THE ABOVE POINTS TO 128 BYTE CREATOR, FOLLOWED BY 128 BYTE NAME*/
    char xpflags; /* VARIOUS FLAGS */
    #define XPFDEBUG 0x01 /* DEBUG IS ON */
    char _f0; /* */
    short xprefP_; /* BLOCK REFERENCE # */
    void *xpssidaA_; /* A(SSID) */
    void *xpuserA_; /* A(USERID) */
    void *xputidaA_; /* A(UTILITY ID) */
    int _f1(|6|); /* */
    /* */
    /* USER AREA */
    /* */
    /* XPUSRMSG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT*/
    /* IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED. */
    /* */
    /* XPUSRMS@ POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT*/
    /* MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE PRINTED AFTER XPUSRMSG (IF ANY). */
    /* FORMAT OF THE MESSAGE BUFFER AT THIS ADDRESS IS: */
    /* */
    /* */
    #define XPFLAGS xpflags
    #define XPFUNC xpfunc
    #define XPREFP_ xprefP_
    #define XPRESRV xpresrv
    #define XPSQLDAA_ xpsqldaA_
    #define XPSSIDA_ xpssidaA_
    #define XTABLEA_ xptableA_
    #define XUSERA_ xpuserA_
    #define XUSERF1 xpuserf1
    #define XUSERF2 xpuserf2
    #define XUSERF3 xpuserf3
    #define XUSERF4 xpuserf4
    #define xusermz xusermz
    void *xpusermA_; /* USER MESSAGE BUFFER ADDRESS */
    int xpusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
    CL100 xpusrmsg; /* USER SINGLE MESSAGE AREA */
    XL2 xpresrv; /* RESERVED */
    char xpusrpad(|842|); /* PADDING */
    #define XPK 1024 /* */
    void *xpuserwA_; /* USER WORK AREA ADDRESS */
    int xpuserf1; /* USER FIELD */
    int xpuserf2; /* USER FIELD */
    int xpuserf3; /* USER FIELD */
    int xpuserf4; /* USER FIELD */
    void *xpusermA_; /* USER MESSAGE BUFFER ADDRESS */
    int xpusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
    CL100 xpusrmsg; /* USER SINGLE MESSAGE AREA */
    XL2 xpresrv; /* RESERVED */
    char xpusrpad(|842|); /* PADDING */
    #define XPK 1024 /* */
    void *xpusermA_; /* USER MESSAGE BUFFER ADDRESS */
    int xpusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
    CL100 xpusrmsg; /* USER SINGLE MESSAGE AREA */
    XL2 xpresrv; /* RESERVED */
    char xpusrpad(|842|); /* PADDING */
    #define XPK 1024 /* */
};
```

Appendix D  UNLOAD PLUS user exits 551
The following table explains major parameter structure elements and their uses:

### Table 121: C user exit parameter fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>xpsqldaA_</em></td>
<td>Pointer to the pseudo-SQLDA for this table as shown in “C pseudo-SQLDA structure” on page 553.</td>
</tr>
<tr>
<td></td>
<td>For each table unloaded, this record describes the column names, data types, and pointers to the column data within the output record being unloaded. This record is identical in format to a DB2 SQLDA, and is used in much the same way.</td>
</tr>
<tr>
<td><em>xptableA_</em></td>
<td>Pointer to the name of the table being unloaded</td>
</tr>
<tr>
<td></td>
<td>This name field allows for a 128-byte creator name, followed by a 128-byte table name.</td>
</tr>
<tr>
<td>xprefP_*</td>
<td>Exit reference number</td>
</tr>
<tr>
<td></td>
<td>Each table being unloaded gets its own copy of a user parameter block and an SQLDA. The XPREF# is incremented to provide a unique reference number.</td>
</tr>
<tr>
<td><em>xpssidA_</em></td>
<td>Pointer to a 4-byte field containing the SSID value</td>
</tr>
<tr>
<td><em>xpuserA_</em></td>
<td>Pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility</td>
</tr>
<tr>
<td><em>xputidA_</em></td>
<td>Pointer to the 16-byte utility ID</td>
</tr>
</tbody>
</table>
**Field** | **Description**
--- | ---
CL100 xpusrm | A 100-byte message field to use when issuing a simple message. On return from any function call, if this field is nonblank, UNLOAD PLUS issues message 51635I identifying the exit name, exit reference number, and this message text. This field is blank after processing.
*xpusermA_* | Pointer to a user-defined message buffer that can contain multiple messages of equal length. Any messages pointed to by this address are printed after the XPUSRM MSG message.
xpuserm$ | Must be set to the size of the user message buffer pointed to by *xpusermA_*

## C pseudo-SQLDA structure

This topic illustrates the C exit pseudo-SQLDA.

**Figure 104: C pseudo-SQLDA record**

```
struct SQLDA
{
  CL8 sqldaid;            /* ID                                      */
  int sqldabc;            /* BYTE COUNT                              */
  short sqln;             /* TOTAL VARS                              */
  short sqld;             /* PERTINENT VARS                          */
  int sqlvar(|0|);        /* BEGIN VARS                              */
  #define SQLDSIZ 16        /* SIZE OF FIXED PART                      */
};

#define SQLD        sqld
#define SQLDABC     sqldabc
#define SQLDAID     sqldaid
#define SQLN        sqln
#define SQLVAR      sqlvar(|0|)
```

The pseudo-SQLDA structure consists of two portions:

- The first portion of the pseudo-SQLDA structure contains the fields of the SQLDA. This portion appears once for each DB2 table being unloaded.
The second portion of the pseudo-SQLDA structure contains the fields of the SQLVAR. This portion is repeated for each column in the DB2 table being unloaded.

Within the SQLVAR portion of the pseudo-SQLDA structure is a field for the SQLTYPE. The value of this field depends on the type of data in the column and whether or not it allows null values.

The tables in “UNLOAD PLUS pseudo-SQLDA” on page 534 summarize the field information for this structure.

The following figure illustrates the C parameter control block relationships:

Figure 105: Major C user exit pointer relationships
Sample C user exit

This topic provides a sample user exit that is written in C.

**Figure 106: Sample C user exit**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "aduxith0.h"      prototypes and #defines
#include "aduxith1.h"      aduexitp

#define ADU_MVS
/*defines
*/
#if !defined(_XL2)
typedef struct
{
#if defined ADU_MVS
    unsigned BF : 16;
#else
    unsigned BF1 : 8;
    unsigned BF2 : 8;
#endif
} XL2;
#endif

#elif !defined(_CL100)

/* prototypes */
__asm void aduxitr1(int,struct LVAREA*);
#endif

#include "aduxith1.h"      aduexitp

#if !defined(_XL2)
#define _XL2
typedef struct
{
    #ifndef ADU_MVS
        unsigned BF : 16;
    #else
        unsigned BF1 : 8;
        unsigned BF2 : 8;
    #endif
    } XL2;
#endif

#if !defined(_CL100)

```
struct ADUEXITP  
{  
    /*********************************************************************
    /* ADUEXITP DEFINES THE UNLOAD USER EXIT PARM BLOCK                  */
    /* YOU MAY NOT MODIFY THE FIELDS IN FRONT OF THE USER AREA           */
    /* (ACTUALLY YOU MAY, BUT YOU WON'T LIKE WHAT HAPPENS)               */
    /*********************************************************************
    int xpfunc; /* 0 = PROCESS, 1 = INIT, 2=TERMINATE */
    void *xpsqldaA_; /* A(SQDA) FOR THIS TABLE */
    void *xptableA_; /* A(TABLE NAME BEING UNLOADED) */
    /* THE ABOVE POINTS TO 128 BYTE CREATOR, FOLLOWED BY 128 BYTE NAME*/
    char xpflags; /* VARIOUS FLAGS */
    #define XPFDEBUG 0x01 /* DEBUG IS ON */
    char _f0; /* */
    short xprefP_; /* BLOCK REFERENCE # */
    void *xpssidA_; /* A(SSID) 4 BYTES*/
    void *xpuserA_; /* A(USERID) 8 BYTES*/
    void *xputidA_; /* A(UTILITY ID) 16 BYTES*/
    int _f1(6); /* */
    /* */
    /* USER AREA */
    /* */
    /* XPUSRMSG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT.*/
    /* IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED. */
    /* */
    /* XPUSRMSA POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT*/
    /* MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE */
    /* PRINTED AFTER XPUSRMSG (IF ANY). */
    /* FORMAT OF THE MESSAGE BUFFER AT THIS ADDRESS IS: */
    /* */
    #define MSGS DS H NUMBER OF LINES, 0 = NO PRINT*/
    #define MSGSIZE DS H SIZE OF EACH LINE (MAX = 100)*
    #define MSGTEXT DS CL(#MSGS*MSGSIZE) USER MESSAGES*/
    /* */
    void *xpuserWA_; /* USER WORK AREA ADDRESS */
    int xpuserf1; /* USER FIELD */
    int xpuserf2; /* USER FIELD */
    int xpuserf3; /* USER FIELD */
    int xpuserf4; /* USER FIELD */
    void *xpusermA_; /* USER MESSAGE BUFFER ADDRESS */
    int xpusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
    CL100 *xpusrmsg; /* USER SINGLE MESSAGE AREA */
    /* */
    XL2 xpresrv; /* RESERVED V12*/
    /* */
    char xpresrv(|842|); /* PADDING */
    #define XPK 1024 /* */
};

#define _CL100
typedef char CL100(|100|);
#endif

#define XPFLAGS                         xpflags
#define XPFUNC                          xpfunc
#define XPRSV                           xpresrv
#define XPSQDAA_                        xpsqldaA_
#define XPSSIDA_                        xpsiddA_
#define XPTABLEA_                       xptableA_
#define XPUSERA_                        xpuserA_
#define XPUSERF1                        xpuserf1
#define XPUSERF2                        xpuserf2
#define XPUSERF3                        xpuserf3
#define XPUSERF4                        xpuserf4

UNLOAD PLUS for DB2 Reference Manual
#define xpusermz            xpusermz
#define XPUSERMA_           xpusermA_
#define XPUSERWA_           xpuserwA_
#define XPUSRMSG           xpusrmsg
#define XPUSRPAD           xpusrpad
#define XPUTIDA_           xputidA_

/*
SYMBOL    OFFSET     SIZE     TYPE      C-TYPE       C-NAME
ADUEXITP  000000    000000    X         char         xpflags
XPFLAGS   000000    000000    F         int          xfunc
XP忽然   000000    000000    H         short        xprefp_
XPRESRV   000000    000000    XL2       XL2          xpresrv
XPSSLDA@  000000    000000    A         void *       xpsqldaA_
XPSSID@   000000    000000    A         void *       xpssidA_
XPTABLE@  000000    000000    A         void *       xptableA_
XPUSER@   000000    000000    A         void *       xpuserA_
XPUSERF1  000000    000000    F         int          xpuserf1
XPUSERF2  000000    000000    F         int          xpuserf2
XPUSERF3  000000    000000    F         int          xpuserf3
XPUSERF4  000000    000000    F         int          xpuserf4
XPUSERMZ  000000    000000    F         int          xpusermz
XPUSERM@  000000    000000    A         void *       xpusrpad
XPUSERW@  000000    000000    A         void *       xpresrv
XPUSRMSG  000000    000000    XL2       XL2          xpusrmsg
XPUSRPAD  000000    000000    A         void *       xpusrpad
XPUTIDA_  000000    000000    A         void *       xputidA_
*/

unless "aduxith2.h"  lvarea /*
/* */

#if !defined(_XL292)
define _XL292
typedef char XL292(|292|);
#endif

#if !defined(_CL128)
define _CL128
typedef char CL128(|128|);
#endif

#if !defined(_CL8)
define _CL8
typedef char CL8(|8|);
#endif

#if !defined(_CL6)
define _CL6
typedef char CL6(|6|);
#endif

#if !defined(_PL2)
define _PL2
typedef struct
#define ADU_MVS
    unsigned BF :16;
#define else
    unsigned BF1 :8;
#endif
    unsigned BF2 :8;
#endif

 Appendix D  UNLOAD PLUS user exits 557
struct LVAREA
{
    /*********************************************************************/
    /* LOCAL VARIABLES WORK AREA DSECT                                */
    /*********************************************************************/
    int lvsave(|18|);        /* LOCAL SAVE AREA */
    PL2 lvkeepP_;           /* RECORDS KEPT */
    PL2 lvdiscP_;           /* RECORDS DISCARDED */
    #define LVBLANKA 76       /* START OF BLANKING AREA */
    CL6 lvkeep;             /* UNPKED KEEPERS */
    CL6 lvdisc;             /* UNPKED DISCARDS */
    CL128 lvtbcrea;         /* TABLE CREATOR */
    CL128 lvtbname;         /* TABLE NAME */
    #define LVBLANKZ 38       /* SIZE OF BLANKING AREA */
    XL292 lvspares;         /* */
    #define LVAREAZ 406       /* */
};

#define LVDISC lvdisc
#define LVDISC# lvdiscP_
#define LVKEEP lvkeep
#define LVKEEP# lvkeepP_
#define LVSAVE lvsave
#define LVSPARES lvspares
#define LVTBCREA lvtbcrea
#define LVTBNAME lvtbname

/*---------------------------------------*/
#include "aduxith3.h"   /* SQLDA Header */
/*---------------------------------------*/
#if !defined(_CL8)
#define _CL8
#endif

struct SQLDA
{
    CL8 sqldaid;            /* ID */
    int sqldabc;            /* BYTE COUNT */
    short sqln;             /* TOTAL VARS */
    short sqld;             /* PERTINENT VARS */
    int sqldvar(|10|);      /* BEGIN VARS */
    #define SQLDSIZ 16        /* SIZE OF FIXED PART */
};

#define SQLD sqld
#define SQLDABC sqldabc
#define SQLDAID sqldaid
#define SQLN sqln
#define SQLVAR sqlvar(|0|)

/*
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
SQLD 00000E 000002 H short sqld
SQLDA 000000 000000
SQLDABC 000008 000004 F int sqldabc
SQLDAID 000000 000008 CL8 CL8 sqldaid
SQLN 00000C 000002 H short sqln
SQLVAR 000010 000000 0F int (|0|) sqlvar
*/

/*---------------------------------------------*/
/* #include "aduxith4.h" SQLVARN HEADER START */
/*---------------------------------------------*/
#if !defined(_CL30)
define _CL30
typedef char CL30(|30|);
#endif

struct SQLVARN
{
  short sqltype;          /* TYPE CODE     */
  #ifdef ADU_MVS
  short sqlllen(|0|);      /* NAME LENGTH   */
  #else
  short sqlllen;
  #endif
  char sqlprcsn;          /* DEC PRECISION */
  char sqlscale;          /* DEC SCALE     */
  void *sqldata;          /* ADDR OF VAR   */
  void *sqlind;           /* ADDR OF IND   */
  union
  {
    struct
    {
      short sqlname;      /* DESCRIBE NAME */
      CL30 _f0;           /*               */
    } _s0;
    struct
    {
      short sqlnamz;      /* SIZEOF COLNAME*/
      CL30 sqlnam;        /* NAME ALONE    */
    } _s1;
    SQLVSIZ 44  /*               */
  };
};

/*--------------------*/
/* end SQLVARN HEADER */
/*--------------------*/
#define SQLDATA sqldata
#define SQLIND sqlind
#define SQLLEN sqlllen(|0|)
#define SQLNAM _s1.sqlnam
#define sqlnamz _s1.sqlnamz
#define SQLNAME _s0.sqlname
#define SQLPRCSN sqlprcsn
#define SQLSCALE sqlscale
#define SQLTYPE sqltype
/*
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
SQLDATA 000004 000004 A void * sqldata
SQLIND 000008 000004 A void * sqlind
SQLLEN 000002 000000 0H short (|0|) sqllen
SQLNAM 00000E 00001E CL30 CL30 _s1.sqlnam
sqlnamz 00000C 000002 H short _s1.sqlnamz
SQLNAME 00000C 000002 H short _s0.sqlname
SQLPRCSN 000002 000001 X char sqlprcsn
SQLSCALE 000003 000001 X char sqlscale
SQLTYPE 000000 000000 H short sqltype
SQLVARN 000000 000000 */

static int BadFuncCode(struct ADUEXITP *pzAduexitp);

#define MAXFUNC 2
#define PROCESS 0
#define INITIAL 1
#define CLEANUP 2

static char
szAccept (||) = "ACCEPTED"
, szDiscard (||) = "DISCARDED"
, szMbadfunc (||) = "BAD FUNCTION CODE RECEIVED FROM MAIN"
, szMgetmerr (||) = "GETMAIN ERROR GETTING USER STORAGE"
, szMgetmer2 (||) = "GETMAIN ERROR GETTING MSG STORAGE"
, szIgot2ini (||) = "COLUMNS AVAILABLE FROM TABLE"
, szIgot2pro (||) = "PROCESSING REGISTRATION ID:"
, szIgot2cin (||) = "PROCESSING IS COMPLETED"
;

int aduexic( long lFcode , struct ADUEXITP *pzAduexitp ){
    int lRc;
    /*-----------------------*
    * Main Entry Point      *
    *-----------------------*/

    lRc = 0;
    switch (lFcode){
    case PROCESS:
        lRc = Process(pzAduexitp);
        break;
    case INITIAL:
        lRc = Initial(pzAduexitp);
        break;
    case CLEANUP:
        lRc = Cleanup(pzAduexitp);
        break;
    default:
        return BadFuncCode(pzAduexitp);
    }
    return lRc;
}

/*----------------------------------------* *
* INTERNAL ROUTINES *
*/
* INTERNAL . ROUTINES *
* INTERNAL ...... ROUTINES *
* INTERNAL .. ROUTINES *
* INTERNAL ROUTINES *
* *
*-------------------------------------------------------------------*/
/*------------------------------------*
* handle bad function call            *
*------------------------------------*/
static int BadFuncCode(struct ADUEXITP *pzAduexitp){
    memset(pzAduexitp->XPUSRMSG ,' ' ,sizeof(pzAduexitp->XPUSRMSG));
    memcpy(pzAduexitp->XPUSRMSG ,szMbadfunc ,sizeof(szMbadfunc));
    return 8;
}
/*------------------------------------*
* Processing call                     *
*------------------------------------*/
static int Process(struct ADUEXITP *pzAduexitp){
    short sCols;
    char *pstrMsg;
    struct SQLDA *pzSqllda;
    struct SQLVARN *pzSqlvarn;
    if ( (pstrMsg=pzAduexitp->XPUSERMA_)!=NULL ){
        *(short*)pstrMsg = 0;
    }
    /* Say I got here */
    memcpy(pzAduexitp->XPUSRMSG ,szIgot2pro ,sizeof(szIgot2pro));
    pzSqllda = pzAduexitp->XPSQLDAA_;    
    sCols = pzSqllda->SQLN;
    pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqllda)+SQLDSIZ);
    while(sCols){
        if ( memicmp(pzSqlvarn->SQLNAM ,"REGISTRATION" ,12)==0 ){               
            memcpy(pzAduexitp->XPUSRMSG+sizeof(szIgot2pro) ,pzSqlvarn->SQLDATA  ,8);
            if ( (memicmp(pzSqlvarn->SQLDATA ,"NCC-1700" ,8)>=0)   
                 && (memicmp(pzSqlvarn->SQLDATA ,"NCC-1711" ,8)<=0) ){        
                /* We're starting over with columns */
                sCols = pzSqllda->SQLN;
                pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqllda)+SQLDSIZ);
                while(sCols){
                    if ( memicmp(pzSqlvarn->SQLNAM ,"NAME ",5)==0){
                        if ( (pzSqlvarn->SQLIND == NULL)  
                            || (pzSqlvarn->SQLIND != NULL) 
                            && *((char*)pzSqlvarn->SQLIND)=='?') ){                
                            memcpy(pzAduexitp->XPUSRMSG  
                                       +sizeof(pzAduexitp->XPUSRMSG)-21  
                                       ,"STARSHIP NAME IS NULL" ,21);
                            return 4; /* Q U I T ! */
                    } else {
                        aduxitr1(2,pzAduexitp->XPUSERWA_); /* bump keep count */
                        return 0; /* Q U I T ! */
                    }
                }
            }
        }
        pzSqlvarn++;
        sCols--;
    }
memcpy(pzAduexitp->XPUSRMSG ,"NAME COLUMN NOT FOUND" ,21);
return 8;  /* Q U I T ! */
}  
else {
    aduxitr1(1,pzAduexitp->XPUSERWA_); /* bump discard count */
    memcpy(pzAduexitp->XPUSRMSG
      +sizeof(pzAduexitp->XPUSRMSG)
      -sizeof(szDiscard)
      ,szDiscard
      ,sizeof(szDiscard));
    return 4;  /* Q U I T ! */
}
}

pzSqlvarn++;
sCols--;
}
memcpy(pzAduexitp->XPUSRMSG ,"REGISTRATION COLUMN NOT FOUND" ,29);
return 8;  /* Q U I T ! */
}

/*------------------------*
*   Exit Initialization   *
*------------------------*/

static int Initial(struct ADUEXITP *pzAduexitp){

short sCols, sMsgs;
char *pstrMsg;
struct LVAREA *pzLvarea;
struct SOLDA *pzSqlda;
struct SQLVARN *pzSqlvarn;

struct TBNAME {
    char strCreator (|8|)  ;
    char strName    (|18|) ;
} *pzTb;

if ( memcmp(pzAduexitp->XPUTIDA_ ,"JST" ,3) != 0){
    return 4;
}  else if (memcmp(((char*)pzAduexitp->XPTABLEA_)+8
                   ,"STARSHIP" ,8) != 0){
    return 4;
}

/* allocate user workarea */
if ( (pzLvarea=calloc(1,LVAREAZ))==NULL){
    return GetmainError(1,pzAduexitp);
}

pzAduexitp->XPUSERWA_ = pzLvarea; /* remember for use later */
aduxitr1(0,pzLvarea);
memset(pzLvarea->LVKEEP ,',',sizeof(LVBLANKZ));
pzTb = pzAduexitp->XPTABLEA_;  
memcpy(pzLvarea->LVTBCREA .pzTb->strCreator .sizeof(pzLvarea->LVTBCREA));
memcpy(pzLvarea->LVTBNAME .pzTb->strName    .sizeof(pzLvarea->LVTBNAME));
/* Say I am here, and have this table */
memcpy(pzAduexitp->XPUSRMSG ,szIgot2ini ,sizeof(szIgot2ini));
memcpy(pzAduexitp->XPUSRMSG+sizeof(szIgot2ini)
        ,pzAduexitp->XPTABLEA_}
.sizeof(struct TBNAME));

pzSqllda = pzAduexitp->XPSQLDAA_;  
SMsps = pzSqllda->SQLN * .sizeof(pzAduexitp->XPUSRMSG);  
SMsps += 4+256;   /* adjust for prefix and pad */  
pzAduexitp->xpusermz = (int)SMsps;  
if ( (pzAduexitp->XPUSERMA_=calloc(1,SMsps))==NULL){  
   return GetmainError(2,pzAduexitp);  
}

SCols = pzSqllda->SQLN;  
pzSqlvarn = (struct SQLVARN*)((char*)pzSqllda)+SQLDSIZ;  
pstrMsg = pzAduexitp->XPUSERMA_;  
pstrMsg += 4;  
SMsps = 0;  
while(sCols){  
   memset(pstrMsg ," ",.sizeof(pzAduexitp->XPUSRMSG));  
   memcpy(pstrMsg ,pzSqlvarn->SQLNAM ,.sizeof(pzSqlvarn->SQLNAM));  
   SMsps++;  
   pstrMsg += .sizeof(pzAduexitp->XPUSRMSG);  
   pzSqlvarn++;  
   sCols--;  
}
/* memcpy(pzAduexitp->XPUSERMA_ ,&SMsps ,.sizeof(SMsps)); */  
*(((short*)pzAduexitp->XPUSERMA_)+1) = .sizeof(pzAduexitp->XPUSRMSG);  
return 0;

/*-----------------------------*  
* Exit termination processing  *  
*-----------------------------*/
  
static int Cleanup(struct ADUEXITP *pzAduexitp){  
    short SMsps;  
    char *pstrMsg;  
    struct LVAREA *pzLvarea;  
    pzLvarea = pzAduexitp->XPUSERWA_;  
    memcpy(pzAduexitp->XPUSRMSG ,szIgot2cln ,.sizeof(szIgot2cln));  
    SMsps = 0;  
    pstrMsg = pzAduexitp->XPUSERMA_;  
    pstrMsg += 4;   /* bump past gunk */  
    aduxitr1(3,pzAduexitp->XPUSERWA_); /* unpack counts */  
    memset(pstrMsg ," ",.sizeof(pzAduexitp->XPUSRMSG));  
    memcpy(pstrMsg ,pzLvarea->LVDISC ,.sizeof(pzLvarea->LVDISC));  
    memcpy(pstrMsg+sizeof(pzLvarea->LVDISC)+1  
      "DID NOT QUALIFY AS HEAVY CRUISERS"  
      .33));  
    pstrMsg += .sizeof(pzAduexitp->XPUSRMSG);  
    SMsps++;  
    memset(pstrMsg ," ",.sizeof(pzAduexitp->XPUSRMSG));  
    memcpy(pstrMsg ,pzLvarea->LVKEEP ,.sizeof(pzLvarea->LVKEEP));  
    memcpy(pstrMsg+sizeof(pzLvarea->LVKEEP)+1  
      "QUALIFIED AS HEAVY CRUISERS"  
      .27));  
    pstrMsg += .sizeof(pzAduexitp->XPUSRMSG);  

Appendix D  UNLOAD PLUS user exits 563
sMsgs++;
memset(pstrMsg ,',', sizeof(pzAduexitp->XPUSRMSG));
memcpy(pstrMsg+sizeof(pzLvarea->LVKEEP)+1 ,"** ALL HEAVY CRUISERS NOW UNLOADED **
,37);
*((short*)pzAduexitp->XPUSERMA_) = sMsgs;
/* free message block if we acquired one */
if (pzAduexitp->XPUSERMA_ != NULL){
  free(pzAduexitp->XPUSERMA_);
  pzAduexitp->XPUSERMA_ = NULL;
}
/* free main workarea */
free(pzAduexitp->XPUSERWA_);
pzAduexitp->XPUSERWA_ = NULL;
return 0;
}

static int GetmainError(int lType ,struct ADUEXITP *pzAduexitp){
  memset(pzAduexitp->XPUSRMSG ,',', sizeof(pzAduexitp->XPUSRMSG));
  if (lType == 1){
    memcpy(pzAduexitp->XPUSRMSG ,szMgetmerr ,sizeof(szMgetmerr));
  } else {
    memcpy(pzAduexitp->XPUSRMSG ,szMgetmer2 ,sizeof(szMgetmer2));
  }
  return 8;
}

/*
.delegate
*   Error in acquiring local storage *
*-----------------------------------*
*/

static int GetmainError(int lType ,struct ADUEXITP *pzAduexitp){
  memset(pzAduexitp->XPUSRMSG ,',', sizeof(pzAduexitp->XPUSRMSG));
  if (lType == 1){
    memcpy(pzAduexitp->XPUSRMSG ,szMgetmerr ,sizeof(szMgetmerr));
  } else {
    memcpy(pzAduexitp->XPUSRMSG ,szMgetmer2 ,sizeof(szMgetmer2));
  }
  return 8;
}

/*
.delegate
*   DISCLAIME R
*--------------------
*/

This is a sample UNLOAD PLUS user exit written in SAS C.

This exit would be used in order to inspect and/or manipulate unload
data records prior to their being written to the sort or output
dataset.

This exit will only be invoked when it is specifically named
in the 'UNLOADEXIT ADUEXIC C' parameter.

Note: please review the documentation in the reference manual,
and the following usage notes prior to implementing this exit.

please call BMC Product Support with any questions you may
have in this area.

Phone: 1-800-841-2031 outside Texas
1-713-240-8800 in Texas

************************************************************************
As always it is tough to come up with a good example for an exit that is intended to be application oriented. For the purposes of this example, a DB2 table exists containing data on Federation starships.

The DB2 table name is STARSHIP, containing the following columns:

- REGISTRATION CHAR(16) NOT NULL,
- NAME VARCHAR(64))

The purpose of this exit is to screen the starships by registration number to filter out any non-heavy cruiser class starships.

In the star fleet, there were 12 heavy cruiser class starships with registration numbers between NCC-1700 and NCC-1711. This code only allows these records to be unloaded. (This filtering could be done with unload control statements, but since this data would appear on a report that could be stolen by the Romulans, starfleet feels it is more secure to hide the logic for selection in a program)

In addition, this screening is only done if the unload utility id begins with the characters 'JST'.

The name must also be checked to ensure that it is not null since this is a nullable field (clerical errors do occur occasionally!)

The bottom line of all this is that this example shows how to:

1. Inspect the user exit block and pseudo SQLDA
2. Getmain storage needed that is preserved during the unload
3. Get main storage for multiple user messages
4. Issue single and multiple messages for DB2 UNLOAD PLUS to issue
5. Find and look at unload table column names
6. Find and inspect data prior to being unloaded
7. Determine null fields
8. Print records being processed with messages
9. Accept and discard records from UNLOAD PLUS processing

ADUEXIC is called at 3 points in processing a table unload.

When invoked, r0 contains a function code.

r1 contains the address of a user exit block described by the ADUEXITP dsect copied to your program from macro member @aduxprm. fields for your use in this block are described in the dsect.

The main field of interest is the pseudo-sqlda pointer xpsqlda@ which points to an "sqlda" prepared for each selected unload table. this sqlda has the same format and contents of a db2 sqlda, with the main exception being that the descriptions of the data fields are the output descriptions (after conversions if any), and the data pointers are pointing to the output record offsets in the record about to be written (function call 0 only).

function:

Function code 0 = process record

this call is made after a record is prepared for writing.
all fields are converted ready for output. the sqlda provides the field types and record positions of the data.

return codes from process:

   0 = accept this record
   4 = discard this record
   anything else = terminate the run.

Function code 1 = initialization call

this call is made during table unload initialization. it happens once per table to be unloaded to allow selection of whether or not to process the table with the exit.

return codes from initialize:

   0 = activate exit for records from this table
   4 = don't use the exit for this table.
   anything else = terminate the run.

Function code 2 = termination/cleanup call

this call is made just prior to termination of the unload, to allow you to perform any cleanup functions necessary.

return codes for terminate are ignored.

for any exit function, the exit may insert message text in the message area for printing upon return.

UNLOAD PLUS LE C user exit

The following topics provide a sample exit parameter structure, exit pseudo-SQLDA structure, and exit that you can use to write a user exit in LE C:

- “LE C exit parameter structure” on page 566
- “Pseudo-SQLDA structure” on page 569
- “Sample LE C user exit” on page 571

LE C exit parameter structure

This topic illustrates the LE C exit parameter structure.
The ADUEXITP structure is contained in a header file, aduxith0.h.

Figure 107: LE C user exit parameter structure

```
struct ADUEXITP {
    /*********************************************************************/
    /* ADUEXITP DEFINES THE UNLOAD USER EXIT PARM BLOCK               */
    /* YOU MAY NOT MODIFY THE FIELDS IN FRONT OF THE USER AREA         */
    /* (ACTUALLY YOU MAY, BUT YOU WON'T LIKE WHAT HAPPENS)             */
    /*********************************************************************/
    int xpfunc; /* 0 = PROCESS, 1 = INIT, 2=TERMINATE */
    void *xpsqldaA_; /* A(SQLDA) FOR THIS TABLE */
    void *xptableA_; /* A(TABLE NAME BEING UNLOADED) */
    /* THE ABOVE POINTS TO 128 BYTE CREATOR, FOLLOWED BY 128 BYTE NAME*/
    char xpflags; /* VARIOUS FLAGS */
    #define XPFDEBUG 0x01 /* DEBUG IS ON */
    char f0; /* */
    short xprefP_; /* BLOCK REFERENCE # */
    void *xpssidA_; /* A(SSID) */
    void *xpuserA_; /* A(USERID) */
    void *xputidA_; /* A(UTILITY ID) */
    int _f16; /* */
    /* */
    /* USER AREA */
    /* */
    /* XPUSRMSG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT,*/
    /* IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED. */
    /* */
    /* XPUSRMSG@ POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT*/
    /* MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE PRINTED AFTER XPUSRMSG (IF ANY). */
    /* FORMAT OF THE MESSAGE BUFFER AT THIS ADDRESS IS: */
    /* */
    /* */
    /* #MSGS DS H  NUMBER OF LINES, 0 = NO PRINT */
    /* MSGSIZE DS H  SIZE OF EACH LINE (MAX = 100) */
    /* MSGTEXT DS CL(#MSGS*MSGSIZE)  USER MESSAGES */
    /* */
    void *xpuserwA_; /* USER WORK AREA ADDRESS */
    int xpuserf1; /* USER FIELD */
    int xpuserf2; /* USER FIELD */
    int xpuserf3; /* USER FIELD */
    int xpuserf4; /* USER FIELD */
    void *xpusermA_; /* USER MESSAGE BUFFER ADDRESS */
    int xpusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
    CL100 xpusrmsg; /* USER SINGLE MESSAGE AREA */
    /* */
    XL2 xpresrv; /* RESERVED */
    /* */
    char xpresrvpad[842]; /* PADDING */
    #define XPK 1024 /* */
};

#define XPFLAGS xpflags
#define XFUNC xpfunc
#define XREFP xprefP_
#define XRESRV xpresrv
#define XSQLDAA xpsqldaA_
#define XSSIDA xptableA_
#define XTABLEA xptableA_
#define XUSERA xpuserA_
#define XUSERF1 xpuserf1_
#define XUSERF2 xpuserf2_
#define XUSERF3 xpuserf3_
#define XUSERF4 xpuserf4_
#define XUSERM xpusermz
```
The following table describes the major parameter structure elements and their uses:

Table 122: LE C user exit parameter fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*xpsqldaA_</td>
<td>Pointer to the pseudo-SQLDA for this table as shown in “Pseudo-SQLDA structure” on page 569</td>
</tr>
<tr>
<td></td>
<td>For each table unloaded, this record describes the column names, data types, and pointers to the column data within the output record being unloaded. This record is identical in format to a DB2 SQLDA, and is used in much the same way.</td>
</tr>
<tr>
<td>*xtableA_</td>
<td>Pointer to the name of the table being unloaded</td>
</tr>
<tr>
<td></td>
<td>This name field allows for a 128-byte creator name, followed by a 128-byte table name.</td>
</tr>
<tr>
<td>xprefP_</td>
<td>Exit reference number</td>
</tr>
<tr>
<td></td>
<td>Each table being unloaded gets its own copy of a user parameter block and an SQLDA. The XPREF# is incremented to provide a unique reference number.</td>
</tr>
<tr>
<td>*xpssidA_</td>
<td>Pointer to a 4-byte field containing the SSID value</td>
</tr>
<tr>
<td>*xpuserA_</td>
<td>Pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility</td>
</tr>
<tr>
<td>*xputidA_</td>
<td>Pointer to the 16-byte utility ID</td>
</tr>
</tbody>
</table>
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL100 xpusrmsg</td>
<td>A 100-byte message field to use when issuing a simple message. On return from any function call, if this field is nonblank, UNLOAD PLUS issues message 51635I identifying the exit name, exit reference number, and this message text. This field is blank after processing.</td>
</tr>
<tr>
<td>*xpusermA_</td>
<td>Pointer to a user-defined message buffer that can contain multiple messages of equal length. Any messages pointed to by this address are printed after the XPUSRMSG message.</td>
</tr>
<tr>
<td>xpuserm$</td>
<td>Must be set to the size of the user message buffer pointed to by *xpusermA_.</td>
</tr>
</tbody>
</table>

---

### Pseudo-SQLDA structure

This topic illustrates the UNLOAD PLUS LE C exit pseudo-SQLDA.

#### Figure 108: LE C pseudo-SQLDA record

```c
struct SQLDA {
    CL8 sqlaid;            /* ID */
    int sqldabc;            /* BYTE COUNT */
    short sqln;             /* TOTAL VARS */
    short sqld;             /* PERTINENT VARS */
    #define SQLDSIZ 16        /* SIZE OF FIXED PART */
};

#define SQLD sqld
#define SQLDABC sqldabc
#define SQLDAID sqldaid
#define SQLN sqln
#define SQLVAR sqlvar[0]
```

| SYMBOL / OFFSET / SIZE / TYPE   / C-TYPE / C-NAME |
|---------------------------------|---------|------------|
| SOLD / 00000E / 000002 / H / short / sqld |
| SQLDA / 000000 / 000000 / CL8 / CL8 / sqlaid |
| SQLDABC / 000008 / 000004 / F / int / sqldabc |
| SQLDAID / 000000 / 000008 / CL8 / CL8 / sqlaid |
| SQLN / 00000C / 000002 / H / short / sqln |
| SQLVAR / 000010 / 000000 / OF / int [0] / sqlvar |

The pseudo-SQLDA structure consists of two portions:

- The first portion of the pseudo-SQLDA structure contains the fields of the SQLDA. This portion appears once for each DB2 table being unloaded.
The second portion of the pseudo-SQLDA structure contains the fields of the SQLVAR. This portion is repeated for each column in the DB2 table being unloaded.

Within the SQLVAR portion of the pseudo-SQLDA structure is a field for the SQLTYPE. The value of this field depends on the type of data in the column and whether or not it allows null values.

The tables in “UNLOAD PLUS pseudo-SQLDA” on page 534 summarize the field information for this structure.

The following figure illustrates the LE C parameter control block relationships:

Figure 109: Major LE C user exit pointer relationships
Sample LE C user exit

This topic provides a sample user exit that is written in LE C.

Figure 110: Sample LE C user exit

```c
/* *********************** IDENTIFICATION AND DESCRIPTION *********************** */
/* PGM: ADUXIL */
/* Description: Example BMC UNLOAD user exit */
/* Please review documentation at the end of this member */
/* Note that all non-standard C headers are included at the front of */
/* this code. */
/* */
/* **************************************************/

#pragma runopts (PLIST(MVS))
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*-----------------------------------------------------*/
/* #include "aduxith0.h" prototypes and #defines */
/*-----------------------------------------------------*/

#define ADU_MVS
/* */
/* defines */
/* */
#ifdef ADU_MSC
#define __asm
#define __BYREF
#else
#define __BYREF @
#endif
/* */
/* prototypes */
/* */
#pragma linkage(ADUXITR1,OS)
/*-----------------------------------------------------*/
/* #include "aduxith1.h" aduxitp */
/*-----------------------------------------------------*/
#if !defined(_XL2)
define _XL2
typedef struct
  {#ifdef ADU_MVS
    unsigned BF : 16;
  #else
    unsigned BF1 : 8;
    unsigned BF2 : 8;
  #endif
  } XL2;
#endif
```
#if !defined(_CL100)
#define _CL100
typedef char CL100100;
#endif

struct ADUEXITP
{

} /* ADUEXITP DEFINES THE UNLOAD USER EXIT PARM BLOCK */
/* YOU MAY NOT MODIFY THE FIELDS IN FRONT OF THE USER AREA */
/* (ACTUALLY YOU MAY, BUT YOU WON'T LIKE WHAT HAPPENS) */
/* ADUEXITP DEFINES THE UNLOAD USER EXIT PARM BLOCK */
/* YOU MAY NOT MODIFY THE FIELDS IN FRONT OF THE USER AREA */
/* (ACTUALLY YOU MAY, BUT YOU WON'T LIKE WHAT HAPPENS) */

int xpfunc; /* 0 = PROCESS, 1 = INIT, 2=TERMINATE */
void *xpsqldaA_; /* A(SQLDA) FOR THIS TABLE */
void *xptableA_; /* A(TABLE NAME BEING UNLOADED) */
/* THE ABOVE POINTS TO 128 BYTE CREATOR, FOLLOWED BY 128 BYTE NAME*/
char xpflags; /* VARIOUS FLAGS */
#define XPFDEBUG 0x01 /* DEBUG IS ON */
char _f0; /* */
short xprefP_; /* BLOCK REFERENCE # */
void *xpssidA_; /* A(SSID) */
void *xpuserA_; /* A(USERID) */
void *xputidA_; /* A(UTILITY ID) */
int _f16; /* */
/* */
/* USER AREA */
/* */
/* XPUUSRMG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT, */
/* IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED */
/* */
/* XPUUSRMG@ POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT*/
/* MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE PRINTED AFTER XPUUSRMG (IF ANY). */
/* FORMAT OF THE MESSAGE BUFFER AT THIS ADDRESS IS: */
/* */
/* #MSGs DS H NUMBER OF LINES, 0 = NO PRINT*/
/* MSGSIZE DS H SIZE OF EACH LINE (MAX = 100]*)
/* MSGTEXT DS CL(#MSGs*MSGSIZE) USER MESSAGES*/
/* */
void *xpuserwa_; /* USER WORK AREA ADDRESS */
int xpuserf1; /* USER FIELD */
int xpuserf2; /* USER FIELD */
int xpuserf3; /* USER FIELD */
int xpuserf4; /* USER FIELD */
void *xpusermA_; /* USER MESSAGE BUFFER ADDRESS */
int xpusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
CL10D xpusrmsg; /* USER SINGLE MESSAGE AREA */
/* */
/* XL2 xpresrv; /* RESERVED V12*/
/* */
char xpusrpad[842]; /* PADDING */
#define XPK 1024 /* */

};

#define XPFLAGS xpflags
#define XPFUNC xpfunc
#define XPREFP xprefP_
#define XPRESRV xpresrv
#define XPSQLDAA xpsqldaA_
#define XPSSIDAA xpssidA_
#define XPTABLEAA xptableA_
#define XPUSERA xpuserA_
#define XPUSERF1 xpuserf1
#define XPUSERF2 xpuserf2
#define XPUSERF3 xpuserf3

UNLOAD PLUS LE C user exit
572 UNLOAD PLUS for DB2 Reference Manual
/*
SYMBOL         OFFSET     SIZE     TYPE      C-TYPE       C-NAME
ADUEXITP       000000    000000
XPFLAGS        00000C    000001    X         char         xpflags
XPFUNC         000000    000004    F         int          xpfunc
XPREF#         00000E    000002    H         short        xprefP_
XPRESRV        0000B4    000002    XL2       XL2          xpresrv
XPSQLDA@       000004    000004    A         void *       xpsqldaA_
XPSSID@        000010    000004    A         void *       xpssidA_
XPTABLE@       000008    000004    A         void *       xptableA_
XPUSER@        000014    000004    A         void *       xpuserA_
XPUSERF1       000038    000004    F         int          xpuserf1
XPUSERF2       00003C    000004    F         int          xpuserf2
XPUSERF3       000040    000004    F         int          xpuserf3
XPUSERF4       000044    000004    F         int          xpuserf4
XPUSERMZ       000048    000004    A         void *       xpusermA_
XPUSERM2       00004C    000004    F         int          xpusermz
XPUSERM@       000048    000004    A         void *       xptableA_
XPUSERW@       000034    000004    A         void *       xpuserwA_
XPUSRMSG       000050    000064    CL100     CL100        xpusrmsg
XPUSRPAD       000086    00034A    (1024-... char [842] xpusrpad
XPUTIDA@       000018    000004    A         void *       xputidA_
*/

/*-----------------------------------------*/
/* #include "aduxith2.h"      lvarea       */
/*-----------------------------------------*/

#if !defined(_XL292)
#define _XL292
typedef char XL292292¨;
#endif

#if !defined(_CL128)
#define _CL128
typedef char CL12818¨;
#endif

#if !defined(_CL8)
#define _CL8
typedef char CL88¨;
#endif

#if !defined(_CL6)
#define _CL6
typedef char CL66¨;
#endif

#if !defined(_PL2)
#define _PL2
typedef struct
{
  #ifdef ADU_MVS
  unsigned BF :16;
  #else
  unsigned BF1 :8;
  unsigned BF2 :8;
  #endif
} PL2;
*/
**UNLOAD PLUS LE C user exit**

```c
#include "aduxith3.h"   SQLDA Header

# ifdef

struct LVAREA
{
  /***************************************************************************/
  /* LOCAL VARIABLES WORK AREA DSECT*/
  /***************************************************************************/
  int lvsavel8;  /* LOCAL SAVE AREA*/
  PL2 lvkeepP_; /* RECORDS KEPT*/
  PL2 lvdiscP_; /* RECORDS DISCARDED*/
  # define LVBLANKA 76 /* START OF BLANKING AREA*/
  CL6 lvkeep; /* UNPKED KEEPERS*/
  CL6 lvdisc; /* UNPKED DISCARDS*/
  CL128 lvtbcrea; /* TABLE CREATOR*/
  CL128 lvtbname; /* TABLE NAME*/
  # define LVBLANKZ 38 /* SIZE OF BLANKING AREA*/
  XL292 lvspares; /*
  # define LVAREAZ 636 /*
  */
  # define LVDISC lvdisc
  # define LVDISCP_ lvdiscP_
  # define LVKEEP lvkeep
  # define LVKEEPP_ lvkeepP_
  # define LVSAVE lvsave
  # define LVSPARES lvspares
  # define LVTBCREA lvtbcrea
  # define LVTBNNAME lvtbname

  /***************************************************************************/
  /* SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME*/
  LVAREA 000000 000000
  LVDISC 000052 000006 CL6 CL6 lvdisc
  LVDISC# 00004A 000002 PL2 PL2 lvdiscP_
  LVKEEP 00004C 000006 CL6 CL6 lvkeep
  LVKEEP# 000048 000002 PL2 PL2 lvkeepP_
  LVSAVE 000000 000048 18F int [18] lvsave
  LVSPARES 000072 000124 XL292 XL292 lvspares
  LVTBCREA 000058 000008 CL128 CL128 lvtbcrea
  LVTBNNAME 000060 000012 CL128 CL128 lvtbname
  /***************************************************************************/

  /***************************************************************************/
  # if ! defined (_CL8)
  # define _CL8
typedef char CL88`;
  # endif

  struct SQLDA
  {
    CL8 sqldaid; /* ID*/
    int sqldabc; /* BYTE COUNT*/
    short sqln; /* TOTAL VARS*/
    short sqld; /* PERTINENT VARS*/
    # define SQLDSIZ 16 /* SIZE OF FIXED PART*/
  };

  # define SQLD sqld
  # define SQLDABC sqldabc
```
```c
#define SQLDAID sqldaid
#define SQLN sqln
#define SQLVAR sqlvar[0]

/*
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
SQLD 00000E 000002 H short sqld
SQLDA 000000 000000
SQLDABC 000008 000004 F int sqldabc
SQLDAID 000000 000008 CL8 CL8 sqldaid
SQLN 00000C 000002 H short sqln
SQLVAR 000010 000000 OF int [0] sqlvar
*/

/*---------------------------------------------*/
/* #include "aduxith4.h" SQLVARN HEADER START */
/*---------------------------------------------*/
#if !defined(_CL30)
#define _CL30
typedef char CL3030¨;
#endif

struct SQLVARN
{
short sqltype; /* TYPE CODE */
char sqlprcsn; /* DEC PRECISION */
char sqlscale; /* DEC SCALE */
void *sqldata; /* ADDR OF VAR */
void *sqlind; /* ADDR OF IND */
struct
{
short sqlnamz; /* SIZEOF COLNAME*/
CL30 sqlnam; /* NAME ALONE */
}_s1;
};

/*--------------------*/
/* end SQLVARN HEADER */
/*--------------------*/
#define SQLDATA sqldata
#define SQLIND sqlind
#define SQLLEN sqllen[0]
#define SQLNAME _s1.sqlnam
#define sqlnamz _s1.sqlnamz
#define SQLNAME _s0.sqlname
#define SQLPRCSN sqlprcsn
#define SQLSCALE sqlscale
#define SQLTYPE sqltype

/*
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
SQLDATA 000004 000004 A void * sqldata
SQLIND 000008 000004 A void * sqlind
SQLLEN 000000 000000 OH short [0] sqllen
SQLNAME 00000E 00001E CL30 CL30 _s1.sqlnam
sqlnamz 00000C 000002 H short _s1.sqlnamz
SQLNAME 000000 000000 H short _s0.sqlname
SQLPRCSN 000002 000001 X char sqlprcsn
SQLSCALE 000003 000001 X char sqlscale
SQLTYPE 000000 000002 H short sqltype
SQLVARN 000000 000000
*/
```

---

**Appendix D**  UNLOAD PLUS user exits 575
static int BadFuncCode(struct ADUEXITP *pzAduexipt);

#define MAXFUNC 2
#define PROCESS 0
#define INITIAL 1
#define CLEANUP 2

static char
    szAccept   [] = "ACCEPTED",
    .szDiscard [] = "DISCARDED",
    .szMbadfunc [] = "BAD FUNCTION CODE RECEIVED FROM MAIN",
    .szMgetmerr [] = "GETMAIN ERROR GETTING USER STORAGE",
    .szMgetmer2 [] = "GETMAIN ERROR GETTING MSG STORAGE",
    .szIgot2ini [] = "COLUMNS AVAILABLE FROM TABLE",
    .szIgot2pro [] = "PROCESSING REGISTRATION ID:"
    .szIgot2cln [] = "PROCESSING IS COMPLETED"
;

int main ( int argc, char *argv[] )
{
    int lRc;
    long lFcode;
    long *rc_ptr;
    long *hold_ptr;
    struct ADUEXITP *pzAduexipt;

    lFcode = *(long *)argv[1];
    hold_ptr = (long *) (*(long *)argv[2]);
    pzAduexipt = (void *)hold_ptr;
    rc_ptr = (long *)argv[3];
    *rc_ptr = 0;

    /*---------------------*
     *   Main Entry Point   *
     *---------------------*/

    lRc = 0;
    switch(lFcode){
        case PROCESS:
            lRc = Process(pzAduexipt);
            break;
        case INITIAL:
            lRc = Initial(pzAduexipt);
            break;
        case CLEANUP:
            lRc = Cleanup(pzAduexipt);
            break;
        default:
            lRc = BadFuncCode(pzAduexipt);
    }
    *rc_ptr = lRc;
    return lRc;
}

/*-------------------------------------------------------------------*
*                                                                    *
*                          INTERNAL ROUTINES                          *
*                       INTERNAL ROUTINES                             *
*                     INTERNAL .... ROUTINES                        *
*                       INTERNAL ROUTINES                             *
*-------------------------------------------------------------------*/

UNLOAD PLUS LE C user exit
INTERNAL ROUTINES

*------------------------------------*
* handle bad function call            *
*------------------------------------*
static int BadFuncCode(struct ADUEXITP *pzAduexitp){
    memset(pzAduexitp->XPUSRMSG ,' ' ,sizeof(pzAduexitp->XPUSRMSG));
    memcpy(pzAduexitp->XPUSRMSG ,szMbadfunc ,sizeof(szMbadfunc));
    return 8;
}

*------------------------------------*
* Processing call                     *
*------------------------------------*
static int Process(struct ADUEXITP *pzAduexitp){
    short sCols;
    char *pstrMsg;
    struct SQLDA *pzSqlnda;
    struct SQLVARN *pzSqlvarn;
    if ( (pstrMsg=pzAduexitp->XPUSERMA_)!=NULL ){
        *(short*)pstrMsg = 0;
    }
    /* Say I got here */
    memcpy(pzAduexitp->XPUSRMSG ,szIgot2pro ,sizeof(szIgot2pro));
    pzSqlnda = pzAduexitp->XPSQLDAA_; 
    sCols = pzSqlnda->SQLN;
    pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqlnda)+SQLDSIZ);
    while(sCols){
        if ( memcmp(pzSqlvarn->SQLNAM ,"REGISTRATION" ,12)==0 ){
            memcpy(pzAduexitp->XPUSRMSG+sizeof(szIgot2pro),
            pzSqlvarn->SQLDATA,8);
            if ( (memcmp(pzSqlvarn->SQLDATA ,"NCC-1700" ,8)>=0) 
            && (memcmp(pzSqlvarn->SQLDATA ,"NCC-1711" ,8)<=0) ){
                /* We're starting over with columns */
                sCols = pzSqlnda->SQLN;
                pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqlnda)+SQLDSIZ);
                while(sCols){
                    if ( memcmp(pzSqlvarn->SQLNAM ,"NAME ",5))==0){
                        if ( (pzSqlvarn->SQLIND == NULL) ||
                        (*((char*)pzSqlvarn->SQLIND)=='?')){
                            memcpy(pzAduexitp->XPUSRMSG 
                            +sizeof(pzSqlvarn->SQLDATA) ,
                            "STARSHIP NAME IS NULL",
                            21);
                            return 4; /* Q U I T ! */
                        } else { 
                            aduxitr1(2,pzAduexitp->XPUSERWA_); /* bump keep count */
                            return 0; /* Q U I T ! */
                        }
                    } else { 
                        aduxitr1(2,pzAduexitp->XPUSERWA_); /* bump keep count */
                        return 0; /* Q U I T ! */
                    }
                }
                return 8; /* Q U I T ! */
            }
        }
    }
}

UNLOAD PLUS LE C user exit
Appendix D  UNLOAD PLUS user exits 577
} else {
  aduxitr1(1,pzAduexitp->XPUSERWA_); /* bump discard count */
  memcpy(pzAduexitp->XPUSRMSG
         +sizeof(pzAduexitp->XPUSRMSG)
         -sizeof(szDiscard)
         ,szDiscard
         ,sizeof(szDiscard));
  return 4; /* Q U I T ! */
}

pzSqlvarn++;
sCols--;
memcpy(pzAduexitp->XPUSRMSG ,"REGISTRATION COLUMN NOT FOUND" ,29);
return 8; /* Q U I T ! */
}

/*------------------------*
*   Exit Initialization   *
*------------------------*/
static int Initial(struct ADUEXITP *pzAduexitp){
  short sCols, sMsgs;
  char *pstrMsg;
  struct LVAREA *pzLvarea;
  struct SQLDA *pzSqlda;
  struct SQLVARN *pzSqlvarn;
  struct TBNAME {
    char strCreator [8]    ;
    char strName    [18] ;
  } *pzTb;

  if ( memcmp(pzAduexitp->XPUTIDA_ ,"JST" ,3) != 0){
    return 4;
  } else if (memcmp(((char*)pzAduexitp->XPTABLEA_)+8
    ,"STARSHIP" ,8) != 0){
    return 4;
  }

  /* allocate user workarea */
  if ( (pzLvarea=calloc(1,LVAREAZ))==NULL){
    return GetmainError(1,pzAduexitp);
  }
  pzAduexitp->XPUSERWA_ = pzLvarea; /* remember for use later */
  aduxitr1(0,pzLvarea);
  memset(pzLvarea->LVKEEP , ' ' ,sizeof(LVBLANKZ));
  pzTb = pzAduexitp->XPTABLEA_;
  memcpy(pzLvarea->LVTBCREA ,pzTb->strCreator,
        sizeof(pzLvarea->LVTBCREA));
  memcpy(pzLvarea->LVTBNAME ,pzTb->strName,
        sizeof(pzLvarea->LVTBNAME));
  /* Say I am here, and have this table */
  memcpy(pzAduexitp->XPUSRMSG ,szIgot2ini ,sizeof(szIgot2ini));
  memcpy(pzAduexitp->XPUSRMSG+sizeof(szIgot2ini)
         ,pzTb->strName ,30);AUS
```c
pzSqllda = pzAduexitp->XPSQLDA_;  
smgs = pzSqllda->SQLN * sizeof(pzAduexitp->XPUSRMSG);  
smgs += 4+256;  /* adjust for prefix and pad */  
pzAduexitp->xpsqlma = (int)smgs;  
if ( (pzAduexitp->XPUSERMA_=calloc(1,smgs))==NULL){  
    return GetmainError(2,pzAduexitp);  
}

sCols = pzSqllda->SQLN;  
pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqllda)+SQLDSIZ);  
pstrMsg = pzAduexitp->XPUSERMA_;  
pstrMsg += 4;  
sMsgs = 0;  
while(sCols){  
    memset(pstrMsg ,' ' ,sizeof(pzAduexitp->XPUSRMSG));  
    memcpy(pstrMsg ,pzSqlvarn->SQLNAM ,sizeof(pzSqlvarn->SQLNAM));  
    sMsgs++;  
    pstrMsg += sizeof(pzAduexitp->XPUSRMSG);  
    pzSqlvarn++;  
    sCols--;  
}  

memcpy(pzAduexitp->XPUSERMA_ ,&sMsgs ,sizeof(sMsgs));  
*(short*)pzAduexitp->XPUSERMA_+1 = sizeof(pzAduexitp->XPUSRMSG);  
return 0;  

/*--------------------------------*  
|   Exit termination processing   |  
*--------------------------------*/  
static int Cleanup(struct ADUEXITP *pzAduexitp){  
    short sMsgs;  
    char *pstrMsg;  
    struct LVAREA *pzLvarea;  
    pzLvarea = pzAduexitp->XPUSERWA_;  
    memcpy(pzAduexitp->XPUSRMSG ,szIgot2cln ,sizeof(szIgot2cln));  
    sMsgs = 0;  
    pstrMsg = pzAduexitp->XPUSERMA_;  
    pstrMsg += 4;  /* bump past gunk */  
    aduxitr1(3,pzAduexitp->XPUSERWA_);  /* unpack counts */  
    memset(pstrMsg ,"" ,sizeof(pzAduexitp->XPUSRMSG));  
    memcpy(pstrMsg ,pzLvarea->LVDISC ,sizeof(pzLvarea->LVDISC));  
    memcpy(pstrMsg+sizeof(pzLvarea->LVDISC)+1  
        "DID NOT QUALIFY AS HEAVY CRUISERS"  
        ,33);  
    pstrMsg += sizeof(pzAduexitp->XPUSRMSG);  
    sMsgs++;  
    memcpy(pstrMsg ,"" ,sizeof(pzAduexitp->XPUSRMSG));  
    memcpy(pstrMsg ,pzLvarea->LVKEEP ,sizeof(pzLvarea->LVKEEP));  
    memcpy(pstrMsg+sizeof(pzLvarea->LVKEEP)+1  
        "QUALIFIED AS HEAVY CRUISERS"  
        ,27);  
    pstrMsg += sizeof(pzAduexitp->XPUSRMSG);  
    sMsgs++;  
```
UNLOAD PLUS LE C user exit

```c
memset(pstrMsg , ' ' , sizeof(pzAduxitp->XPUSRMSG));
memcpy(pstrMsg+sizeof(pzLvarea->LVKEEP)+1
   "*** ALL HEAVY CRUISERS NOW UNLOADED **",
   37);
*((short*)pzAduxitp->XPUSERMA_) = sMsgs;
/* free message block if we acquired one */
if (pzAduxitp->XPUSERMA_ != NULL)
   free(pzAduxitp->XPUSERMA_);
pzAduxitp->XPUSERMA_ = NULL;
/* free main workarea */
free(pzAduxitp->XPUSERWA_);
pzAduxitp->XPUSERWA_ = NULL;
return 0;
}
/*-------------------------------------*
|   Error in acquiring local storage  |
*-------------------------------------*/
static int GetmainError(int lType ,struct ADUEXITP *pzAduxitp){
    memset(pzAduxitp->XPUSRMSG , ' ' , sizeof(pzAduxitp->XPUSRMSG));
    if (lType == 1){
        memcpy(pzAduxitp->XPUSRMSG , szMgetmerr , sizeof(szMgetmerr));
    } else {
        memcpy(pzAduxitp->XPUSRMSG , szMgetmer2 , sizeof(szMgetmer2));
    }
    return 8;
}
/*
***************************
|   DISCLAIMER           |
***************************
This is a sample UNLOAD PLUS user exit written in LE C.
This exit would be used in order to inspect and/or manipulate unload
data records prior to their being written to the sort or output
dataset.
This exit will only be invoked when it is specifically named
in the 'UNLOADEXIT ADUXIL LE_C' parameter.
Note: please review the documentation in the reference manual,
and the following usage notes prior to implementing this exit.
please call BMC Product Support with any questions you may
have in this area.
Phone: 1-800-841-2031    outside Texas
1-713-240-8800    in Texas
*******************************
|   EXITS CENARIO            |
*******************************
```

As always it is tough to come up with a good example for an exit that is intended to be application oriented. For the purposes of this example, a DB2 table exists containing data on Federation starships.

The DB2 table name is STARSHIP, containing the following columns:

```
REGISTRATION CHAR(16) NOT NULL,
NAME VARCHAR(64)
```

The purpose of this exit is to screen the starships by registration number to filter out any non-heavy cruiser class starships.

In the star fleet, there were 12 heavy cruiser class starships with registration numbers between NCC-1700 and NCC-1711. This code only allows these records to be unloaded. (This filtering could be done with unload control statements, but since this data would appear on a report that could be stolen by the Romulans, starfleet feels it is more secure to hide the logic for selection in a program)

In addition, this screening is only done if the unload utility id begins with the characters 'JST'.

The name must also be checked to ensure that it is not null since this is a nullable field (clerical errors do occur occasionally!)

The bottom line of all this is that this example shows how to:

1. Inspect the user exit block and pseudo SQLDA
2. Get main storage needed that is preserved during the unload
3. Get main storage for multiple user messages
4. Issue single and multiple messages for DB2 UNLOAD PLUS to issue
5. Find and look at unload table column names
6. Find and inspect data prior to being unloaded
7. Determine null fields
8. Print records being processed with messages
9. Accept and discard records from UNLOAD PLUS processing

***************
*     N O T E S     *
***************

ADUXEIL is called at 3 points in processing a table unload.

When invoked, r0 contains a function code.

r1 contains the address of a user exit block described by the ADUXEITP dsect copied to your program from macro member @aduxprm. fields for your use in this block are described in the dsect.

The main field of interest is the pseudo-sqlda pointer xpsqlda which points to an "sqlda" prepared for each selected unload table. this sqlda has the same format and contents of a db2 sqlda, with the main exception being that the descriptions of the data fields are the output descriptions (after conversions if any), and the data pointers are pointing to the output record offsets in the record about to be written (function call 0 only).

function:

```
Function code 0 = process record
```

this call is made after a record is prepared for writing. all fields are converted ready for output. the sqlda provides the field types and record positions of the data.
return codes from process:
   0 = accept this record
   4 = discard this record
   anything else = terminate the run.

Function code 1 = initialization call
   this call is made during table unload initialization. it
   happens once per table to be unloaded to allow selection of
   whether or not to process the table with the exit.
   return codes from initialize:
   0 = activate exit for records from this table
   4 = don’t use the exit for this table.
   anything else = terminate the run.

Function code 2 = termination/cleanup call
   this call is made just prior to termination of the unload.
   to allow you to perform any cleanup functions necessary.
   return codes for terminate are ignored.

   for any exit function, the exit may insert message text in the
   message area for printing upon return.

*/

**UNLOAD PLUS COBOL II and LE COBOL user exits**

Two COBOL II and LE COBOL sample user exits are provided in the UNLOAD PLUS sample.

The first user exit (ADUEXITC) is a simple example that produces a report of data read and unloaded by UNLOAD PLUS. It illustrates using the passed parameter block and retrieving the unloaded data fields within the output record. The second user exit (ADUEXTC2) builds on the first one, illustrating a comparison of rows passed from UNLOAD PLUS with records read from an external data set.

**Exit parameter record**

This topic illustrates the UNLOAD PLUS COBOL exit parameter record.
The exit parameter record is the same for both sample exits.

**Figure 111: COBOL user exit parameter record**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT-SQLDA-POINTER</td>
<td>Pointer to the pseudo-SQLDA for this table as shown in “COBOL pseudo-SQLDA record” on page 584. For each table unloaded, this record describes the column names, data types, and pointers to the column data within the output record being unloaded. This record is identical in format to a DB2 SQLDA and is used in much the same way.</td>
</tr>
<tr>
<td>EXIT-TB-NAME-POINTER</td>
<td>Pointer to the name of the table being unloaded. This name field allows for a 128-byte creator name, followed by a 128-byte table name.</td>
</tr>
<tr>
<td>EXIT-REFERENCE-NUM</td>
<td>Exit reference number. Each table being unloaded gets its own copy of a user parameter block and an SQLDA. The XPARF# is incremented to provide a unique reference number.</td>
</tr>
<tr>
<td>EXIT-SSID-POINTER</td>
<td>Pointer to a 4-byte field containing the SSID value</td>
</tr>
<tr>
<td>EXIT-USERID-POINTER</td>
<td>Pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility</td>
</tr>
<tr>
<td>EXIT-UTILID-POINTER</td>
<td>Pointer to the 16-byte utility ID</td>
</tr>
</tbody>
</table>
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT-USER-MESSAGE</td>
<td>A 100-byte message field to use when issuing a simple message. On return from any function call, if this field is nonblank, UNLOAD PLUS issues message BMC51635I identifying the exit name, exit reference number, and this message text. This field is blank after processing.</td>
</tr>
<tr>
<td>EXIT-USERMSG</td>
<td>Pointer to a user-defined message buffer that can contain multiple messages of equal length. Any messages pointed to by this address are printed after the XPUUSRMSG message.</td>
</tr>
<tr>
<td>EXIT-USERMSG-SIZE</td>
<td>Must be set to the size of the user message buffer pointed to by XPUSERM@</td>
</tr>
</tbody>
</table>

### COBOL pseudo-SQLDA record

This topic illustrates the exit pseudo-SQLDA DSECTs for the two COBOL sample user exits.

**Figure 112: COBOL pseudo-SQLDA record: sample 1**

```cobol
01 SQLDA.
   02 SQLDAX PIC X(8).
   02 SQLDABC PIC S9(8) COMPUTATIONAL.
   02 SQLN PIC S9(4) COMPUTATIONAL.
   02 SQLD PIC S9(4) COMPUTATIONAL.
   02 SQLVAR OCCURS 1 TO 300 TIMES DEPENDING ON SQLN.
      03 SQLOCTYPE PIC S9(4) COMP.
      03 SQLLEN PIC S9(4) COMP.
      03 SQLDATA POINTER.
      03 SQLIND.
         04 SQLNAMEL PIC S9(4) COMP.
         04 SQLNAMEC PIC X(30).
```

**Figure 113: COBOL pseudo-SQLDA record: sample 2**

```cobol
01 SQLDA.
   02 SQLDAX PIC X(8).
   02 SQLDABC PIC S9(8) COMPUTATIONAL.
   02 SQLN PIC S9(4) COMPUTATIONAL.
   02 SQLD PIC S9(4) COMPUTATIONAL.
   02 SQLVAR OCCURS 1 TO 300 TIMES DEPENDING ON SQLN INDEXED BY I.
      03 SQLTYPE PIC S9(4) COMP.
      03 SQLLEN PIC S9(4) COMP.
      03 SQLDATA POINTER.
      03 SQLIND.
      03 SQLINDN REDEFINES SQLIND PIC S9(9) COMP.
      03 SQLNAME.
         04 SQLNAMEL PIC S9(4) COMP.
         04 SQLNAMEC PIC X(30).
```

The pseudo-SQLDA DSECT consists of two portions:

- The first portion of the pseudo-SQLDA DSECT contains the fields of the SQLDA. This portion appears once for each DB2 table being unloaded.
The second portion of the pseudo-SQLDA DSECT contains the fields of the SQLVAR. This portion is repeated for each column in the DB2 table being unloaded.

Within the SQLVAR portion of the pseudo-SQLDA DSECT is a field for the SQLTYPE. The value of this field depends on the type of data in the column and whether or not it allows null values.

The tables in “UNLOAD PLUS pseudo-SQLDA” on page 534 summarize the field information for this structure.

The following figure illustrates the COBOL parameter control block relationships:

Figure 114: Major COBOL user exit pointer relationships
This topic provides a sample user exit that is written in COBOL.

Figure 115: Sample COBOL user exit: sample 1

```
*------------------------
IDENTIFICATION DIVISION.
*------------------------
*
* THIS IS A SAMPLE OF A COBOL EXIT FOR UNLOAD+ FOR DB2.
*
PROGRAM-ID. ADEXITC.
DATE-COMPILED.
*---------------------
ENVIRONMENT DIVISION.
*---------------------
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-Control.
*-----------------
DATA DIVISION.
*-----------------
FILE SECTION.
WORKING-STORAGE SECTION.
*------------------------
* WORKAREAS AND VARIABLES
*------------------------
01 INIT-MSG.
   10 FILLER PIC X(13) VALUE "UNLOAD INFO: ".
   10 FILLER PIC X(03) VALUE "TB=".
   10 IM-CREATOR PIC X(08).
   10 FILLER PIC X(01) VALUE ".".
   10 IM-NAME PIC X(18).
   10 FILLER PIC X(06) VALUE ",SSID=".
   10 IM-SSID PIC X(04).
   10 FILLER PIC X(08) VALUE ",USERID=".
   10 IM-USERID PIC X(08).
   10 FILLER PIC X(08) VALUE ",UTILID=".
   10 IM-UTID PIC X(16).
01 TERM-MSG.
   10 FILLER PIC X(16) VALUE "CLEANUP DONE. ".
   10 FILLER PIC X(21) VALUE "PROCESSED ROW COUNT: ".
   10 TERM-ROW-COUNT PIC ZZ,ZZZ,ZZ9.
*01 WORK-STUFF.
*   10 ROW-COUNT PIC S9(9) VALUE 0.
LINKAGE SECTION.
01 EXITPARMS.
   02 EXIT-FUNCTION-CODE PIC S9(9) BINARY.
     88 PROCESS-CALL VALUE 0.
     88 INITIALIZE-CALL VALUE 1.
     88 CLEANUP-CALL VALUE 2.
   02 EXIT-SQLDA-POINTER POINTER.
```
PROCEDURE DIVISION USING EXITPARMS.

EVALUATE TRUE
   WHEN PROCESS-CALL PERFORM PROCESS-RECORD
      THRU PROCESS-RECORD-EXIT
   WHEN INITIALIZE-CALL PERFORM INITIALIZE-THE-EXIT
      THRU INITIALIZE-THE-EXIT-EXIT
   WHEN CLEANUP-CALL PERFORM CLEANUP-THE-EXIT
      THRU CLEANUP-THE-EXIT-EXIT
END-EVALUATE.

GOBACK.

*---------------------
PROCESS-RECORD.
*---------------------

MOVE ZERO TO RETURN-CODE.
COBOL II and LE COBOL user exits: sample 2

This topic provides another sample user exit that is written in COBOL.

Figure 116: Sample COBOL user exit: sample 2

IDENTIFICATION DIVISION.

* THIS IS A SAMPLE OF A COBOL EXIT FOR UNLOAD+ FOR DB2.
* THIS IS A SIMPLE COMPARASION OF ROWS PASSED FROM UNLOAD+
* AND RECORDS READ FROM A SEQUENTIAL DATA SET TO WRITE AN
* OUTPUT RECORD. AS EACH ROW IS RECEIVED FROM THE TABLE
* A SEQUENTIAL FILE IS READ IN PARALLEL.
* IF THE SEQUENTIAL RECORD INDICATOR
  "COMES FROM INPUT TABLE" IS FOUND THE OUTPUT
* RECORD IS WRITTEN FROM THE SEQUENTIAL DATA SET,
* OTHERWISE THE OUTPUT RECORD WRITTEN FROM THE TABLE.
*
PROGRAM-ID.  ADEXTC2.
DATE-COMPiled.

*---------------------
ENVIRONMENT DIVISION.
*---------------------

CONFIGURATION SECTION.

INPUT-OUTPUT SECTION.

FILE-Control.

  SELECT COBOLOUT ASSIGN TO UT-S-COBOLOUT.
  SELECT COBOLIN ASSIGN TO UT-S-COBOLIN.

*-------------------------
DATA DIVISION.
*-------------------------

FILE SECTION.

FD  COBOLOUT
    RECORD CONTAINS 80 CHARACTERS
    RECORDING MODE IS F
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS OUTREC.

  01  OUTREC  PIC X(80).

FD  COBOLIN
    RECORD CONTAINS 80 CHARACTERS
    RECORDING MODE IS F
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS INREC.

  01  INREC  PIC X(80).

WORKING-STORAGE SECTION.

* WORKAREAS AND VARIABLES
*-------------------------

  01  INIT-MSG.
     10  FILLER  PIC X(13) VALUE 'UNLOAD INFO: '.
     10  FILLER  PIC X(03) VALUE 'TB='.
     10  IM-CREATOR  PIC X(08).
     10  FILLER  PIC X(01) VALUE '.'.
     10  IM-NAME  PIC X(18).
     10  FILLER  PIC X(06) VALUE ',SSID='.
     10  IM-SSID  PIC X(04).
     10  FILLER  PIC X(06) VALUE ',USERID='.
     10  IM-USERID  PIC X(08).
     10  FILLER  PIC X(06) VALUE ',UTILID='.
     10  IM-UTID  PIC X(16).

  01  OUTDATA.
     10  P-ID  PIC 9(4).
     10  P-NAME.
        49  P-NAME-LEN  PIC 9(4).
        49  P-NAME-TEXT  PIC X(9).
     10  P-DEPT  PIC 9(4).
UNLOAD PLUS COBOL II and LE COBOL user exits

10 P-JOB PIC X(5).
10 P-YEARS PIC 9(4).
10 P-SALARY PIC 9(5)V9(2).
10 P-COMM PIC 9(5)V9(2).
10 P-FILLER PIC X(21) VALUE 'DB2 TABLE'.

01 INDATA.
  10 I-ID PIC 9(4).
  10 I-NAME.
     49 I-NAME-LEN PIC 9(2).
     49 I-NAME-TEXT PIC X(11).
  10 I-DEPT PIC 9(4).
  10 I-JOB PIC X(5).
  10 I-YEARS PIC 9(4).
  10 I-SALARY PIC 9(5)V9(2).
  10 I-COMM PIC 9(5)V9(2).
  10 I-COMES-FROM PIC X(21).
  10 I-FILLER PIC X(15).

01 TERM-MSG.
  10 FILLER PIC X(16) VALUE 'CLEANUP DONE.  '.
  10 FILLER PIC X(21) VALUE 'PROCESSED ROW COUNT:  '.
  10 TERM-ROW-COUNT PIC ZZ,ZZZ,ZZ9.

01 WORK-STUFF.
  10 ROW-COUNT PIC S9(9) VALUE 0.

*-----------------------
| LINKAGE SECTION.     |
*-----------------------

01 EXTPARMS.
  02 EXIT-FUNCTION-CODE PIC S9(9) BINARY.
   88 PROCESS-CALL VALUE 0.
   88 INITIALIZE-CALL VALUE 1.
   88 CLEANUP-CALL VALUE 2.
  02 EXIT-SQDA-POINTER POINTER.
  02 EXIT-TB-NAME-POINTER POINTER.
  02 FILLER PIC XX.
  02 EXIT-REFERENCE-NUM PIC S9(4) BINARY.
  02 EXIT-SSID-POINTER POINTER.
  02 EXIT-USERID-POINTER POINTER.
  02 EXIT-UTILID-POINTER POINTER.
  02 FILLER PIC X(24).
  02 EXIT-USER-POINTER POINTER.
  02 EXIT-USER1 PIC S9(9) BINARY.
  02 EXIT-USER2 PIC S9(9) BINARY.
  02 EXIT-USER3 PIC S9(9) BINARY.
  02 EXIT-USER4 PIC S9(9) BINARY.
  02 EXIT-USERMSG-POINTER POINTER.
  02 EXIT-USERMSG-SIZE PIC S9(9) BINARY.
  02 EXIT-USER-MESSAGE PIC X(100).
  02 EXIT-FLAGS PIC X.
  02 FILLER PIC X.
  02 FILLER PIC X(214).
  02 FILLER PIC X(100).

01 TABLE-INFO.
  13 TABLE-CREATOR PIC X(128).
  13 TABLE-NAME PIC X(128).
  01 SSID PIC X(04).
  01 USERID PIC X(08).
  01 UTILITY-ID PIC X(16).

01 SQlda.
  02 SQLDAX PIC X(8).
  02 SQLDABC PIC S9(8) COMPUTATIONAL.
02 SQLN     PIC S9(4) COMPUTATIONAL.
02 SQLD     PIC S9(4) COMPUTATIONAL.
02 SQLVAR   OCCURS 1 TO 300 TIMES DEPENDING ON SQLN
            INDEXED BY I.
  03 SQLTYPE      PIC S9(4) COMP.
  03 SQLLEN       PIC S9(4) COMP.
  03 SQLDATA      POINTER.
  03 SQLIND       POINTER.
  03 SQLINDN REDEFINES SQLIND PIC S9(9) COMP.
  03 SQLNAME.
    04 SQLNAMEL     PIC S9(4) COMP.
    04 SQLNAMEC     PIC X(30).
01 NUMID          PIC S9(4) USAGE COMP.
01 NAME.
  49 NAME-LEN    PIC S9(4) USAGE COMP.
  49 NAME-TEXT   PIC X(9).
01 DEPT           PIC S9(4) USAGE COMP.
01 JOB            PIC X(5).
01 YEARS          PIC S9(4) USAGE COMP.
01 SALARY         PIC S9(5)V9(2) USAGE COMP-3.
01 COMM           PIC S9(5)V9(2) USAGE COMP-3.
01 NULLBYTE       PIC X.
01 RECORD-DATA-WORK.
  10 DATA-FIELD   PIC X(20).

*-------------------
PROCEDURE DIVISION USING EXITPARMS.
*-------------------
EVALUATE TRUE
WHEN PROCESS-CALL PERFORM PROCESS-RECORD
    THRU PROCESS-RECORD-EXIT
WHEN INITIALIZE-CALL PERFORM INITIALIZE-THE-EXIT
    THRU INITIALIZE-THE-EXIT-EXIT
WHEN CLEANUP-CALL PERFORM CLEANUP-THE-EXIT
    THRU CLEANUP-THE-EXIT-EXIT
END-EVALUATE.
GOBACK.

*--------------------
PROCESS-RECORD.
*--------------------
SET ADDRESS OF SQLDA TO EXIT-SQLDA-POINTER.
PERFORM BUILD-PRINT-LINE THRU BUILD-PRINT-LINE-EXIT
    VARYING I FROM 1 BY 1
    UNTIL I GREATER THAN SQLN.
READ COBOLIN INTO INDATA.
IF I-COMES-FROM = 'COMES FROM INPUT FILE'
    THEN MOVE INDATA TO OUTDATA.
WRITE OUTREC FROM OUTDATA.
MOVE '' TO I-COMES-FROM.
MOVE 'DB2 TABLE' TO P-FILLER.
MOVE NAME-TEXT TO EXIT-USER-MESSAGE.
ADD 1 TO ROW-COUNT.
PROCESS-RECORD-EXIT.
EXIT.
*--------------------
INITIALIZE-THE-EXIT.
*--------------------

SET ADDRESS OF TABLE-INFO TO EXIT-TB-NAME-POINTER.
SET ADDRESS OF SSID TO EXIT-SSID-POINTER.
SET ADDRESS OF USERID TO EXIT-USERID-POINTER.
SET ADDRESS OF UTILITY-ID TO EXIT-UTILID-POINTER.

MOVE TABLE-CREATOR TO IM-CREATOR.
MOVE TABLE-NAME TO IM-NAME.
MOVE SSID TO IM-SSID.
MOVE USERID TO IM-USERID.
MOVE UTILITY-ID TO IM-UTID.
MOVE INIT-MSG TO EXIT-USER-MESSAGE.

MOVE ZERO TO ROW-COUNT.
* TO NOT USE THIS PROGRAM, MOVE 4 TO RETURN-CODE.

OPEN OUTPUT COBOLOUT.
OPEN INPUT COBOLIN.

INITIALIZE-THE-EXIT-EXIT.
EXIT.

*--------------------
CLEANUP-THE-EXIT.
*--------------------

MOVE ROW-COUNT TO TERM-ROW-COUNT.

CLOSE COBOLOUT.
CLOSE COBOLIN.
MOVE TERM-MSG TO EXIT-USER-MESSAGE.

CLEANUP-THE-EXIT-EXIT.
EXIT.

*--------------------
BUILD-PRINT-LINE.
*--------------------

* PREPARE TEST FOR THE NULL INDICATOR

IF SQLINDN(I) EQUAL 0
THEN
  SET ADDRESS OF NULLBYTE TO NULL
ELSE
  SET ADDRESS OF NULLBYTE TO SQLIND(I).

* PROCESS EACH COLUMN BY NUMBER AS CALLED BY THE PERFORM/VARYING

IF I = 1
THEN
  SET ADDRESS OF NUMID TO SQLDATA(I)
  MOVE NUMID TO P-ID.
IF I = 2
THEN
  SET ADDRESS OF NAME TO SQLDATA(I)
  MOVE NAME TO P-NAME.
IF I = 3
THEN
  SET ADDRESS OF DEPT TO SQLDATA(I)
  MOVE DEPT TO P-DEPT.
IF I = 4
THEN
SET ADDRESS OF JOB TO SQLDATA(I)
MOVE JOB TO P-JOB.
IF I = 5
THEN
SET ADDRESS OF YEARS TO SQLDATA(I)
MOVE YEARS TO P-YEARS.
IF I = 6
THEN
SET ADDRESS OF SALARY TO SQLDATA(I)
MOVE SALARY TO P-SALARY.
IF I = 7
THEN
SET ADDRESS OF COMM TO SQLDATA(I).
BUILD-PRINT-LINE-EXIT.
EXIT.
Symbols

&ATTACH variable 186, 450
&DATE variable 186, 450
&DAY variable 186, 450
&DB variable
  description 186, 450
  resolving when using DIRECT NO 188
&HOUR variable 186, 450
&JDATE variable 186, 450
&JDAY variable 186, 450
&JOBNAME variable 186, 450
&MIN variable 186, 450
&MINUTE variable 186, 450
&MONTH variable 186, 450
&OBID variable
  description 186, 450
  resolving when using DIRECT NO 188
&PART variable
  description 186, 450
  examples 296, 373
  resolving when using DIRECT NO 188
&POSNUM variable 186, 450
&SEC variable 186, 450
&SECOND variable 186, 450
&SELNUM variable
  description 186, 450
  example 296
&SSID variable 108, 186, 450
&STEPNAME variable 186, 450
&TIME variable 186, 450
&TS variable
  description 186, 450
  description 186, 450
  resolving when using DIRECT NO 188
&TYPE variable 186, 450
&USERID variable 186, 450
&UTIL variable 186, 450
&YEAR variable 186, 450

$532SOPT configuration job 417
$730DOP2 configuration job 417
$ADUOPTS macro 418
$ADUOUTP macro 417
$AUPSMAC macro 417
$NO_UNITS$ keyword 183, 449
$ORTPARM DD statement 308

A

abends
  JES3 35, 307
    recovery from 171
    using STOP UTILITY for 173
    using TERMINATE UTILITY for 173
above-the-bar storage 45
accelerator server 182
access, controlling
  in UNLOAD PLUS 43
access, shared 478
accessibility of objects 25, 103
ACF2. See CA ACF2
ACFORTSS option 18
ACHKP (auxiliary CHECK pending) status 49
ACS (Automatic Class Selection) routines 304
activating dynamic allocation 122, 442
ACTIVE option 122
ADU$OPTS options module 417
ADUUMAIN module 272
algorithms for calculating space for data sets 304
aliases, specifying in FROM option 211
ALL keyword
  of FORCE option 180, 430
allocating data sets

Index
copy 279
DYLIN 278
dynamically 294
estimating size with ANALYZE option 123, 422
output 286
sort (SORTWK) 281
SORTWK 281
SYSCNTL 284
SYSREC 286
SYSRED 286
VSAM 281
allocating directory, for referenced file data sets 190
analytics server 24
ANALYZE option
command 123
installation 422
using to calculate data set size 302
AND option of FORMAT CSV option 73
APF authorization requirements 276, 417, 531
application defaults module 273
application-period temporal tables
CNTLCARDS BMCLOAD 133
CNTLCARDS DB2 132
CNTLCARDS DB2DDL 135
CNTLCARDS DB2LOAD 131
ASC keyword of ORDER BY option 219
ASCII
data translation 261
exit routines 77
option 158
ASM keyword of UNLOADEXIT option 169
assembler user exit routine
example 538
specifying 169
authorization mechanisms 42
authorizations
APF authorization 276, 531
DB2 and data set
UNLOAD PLUS 43
DISPLAY 42
exit routines 77
for canceling threads 42
for data sets 42
mechanisms, description 42
OPNDB2ID option 436
RACF 42
UNLOAD PLUS 43
READ privileges 42
required to run UNLOAD PLUS 42
system 43
UNLOAD PLUS 43
verifying 42
AUTO keyword
of DIRECT option 101
of FORMAT XML option 152
of RECFM option 128, 436
AUTOTAG option 145
auxiliary CHECK pending (ACHKP) status 49
availability, data 25, 103

B
backing up BMC tables 469
BASIC keyword of DSNTYPE option 189, 453
batch jobs
building 271
executing 308
batch pipe data 64
batch threads, number used by UNLOAD PLUS 45
benefits of UNLOAD PLUS 21
BIGINT data type 237
binary data 240, 244
BINARY data type 240
binary files, specifying 230, 233
BINARY VARYING data type 240
binary XML data 230, 233
BINARYXML keyword
of CHAR data type 230
of VARCHAR data type 233
bitemporal tables
CNTLCARDS BMCLOAD 133
CNTLCARDS DB2 132
CNTLCARDS DB2DDL 135
CNTLCARDS DB2LOAD 131
FORMAT BMCLOAD 67
BLOB data type 244
BLOBF keyword of character data types 230, 233
block size, of copy data sets 404, 432
BMC internal format, output in 150
BMC utilities
    displaying status 471
    running concurrently 46, 484
    terminating 471
BMCDICT table
    considerations 473
    contents 472
    maintaining 473
BMCFORCE data set 278
BMCHIST installation option, BMCHIST table 474
BMCHIST table
    backing up 469
    contents 474
    COPY PLUS considerations 476
    maintenance 477
    querying 471
    RECOVER PLUS considerations 476
    updating using HISTORY 432
BMCLGRNX table 477
BMCLOAD keyword
    of CNTLCARDS option 133, 295, 389
    of FORMAT option 150
BMCSORT
    controlling 405
    data set that contains messages 307
    description 38
    ENUMROWS to pass number of rows to 141
    memory usage 405, 439
    performance tuning 405
    SMCORE option 439
    sort work data sets, allocating 281
    SORTDEVT option 142
    SORTNUM option 142, 440
    version requirement 41
BMCSORT installation option 463
BMCSYNC table
    backing up 469
    cleaning up RECOVER PLUS UNLOADKEYS 483
    considerations 481
    contents 478
    LOB data considerations 481
    maintaining 483
    purpose 46
    running utilities concurrently 46, 484
    updating 434
    XML data considerations 481
BMCTRANS table 486
BMCUNLOAD option (in LOADPLUS) 389
BMCUTIL table
    backing up 469
    cabinet copies 113
    contents 491
    Instant Snapshot copies 118
    maintaining 498
    online consistent copies 118
    querying 471
    buffer usage
    controlling 404
    IBUFFS option 404, 432
    specifying for INFILE option data sets 404
    UBUFFS option 405, 441
BYTASK keyword of ORDER YES option 139

C
C language
    sample user exit 550
    UNLOADEXIT option 169
CA ACF2 security product 42
CA Top Secret security product 42
cabinet copies, unloading from 110
cache, sufficient for SUF 53
caching, DASD 410
calculating
data set size during dynamic allocation 302
estimated number of rows for a table space 125
nonpartitioned sort work space 282
SORTWK data sets size 282
space for SMS-managed data sets 304
calculating SORTWK data set size 282
canceling a job 309
canceling threads
  authorizations required 42
  FORCE option 179, 430
  FORCE_AT option 431
  FORCE_RPT option 432
  reporting 181, 278, 432
  specifying when to cancel 180, 431
carriage return characters, in FORMAT CSV options 73
catalog of DB2, unloading from 63
CCSID option
  description 159
  performance considerations 401
  relation to FORMAT BMCLOAD option 70
CCSIDs (code character set identifiers) 158, 160
CENTURY option
  command 175
  installation 423
  of DATE data type 241
  of TIMESTAMP data type 242
CHANGE_CONSISTENT option 423
CHANGEQUIESCE option 424
changes to the product 14
CHAR data type 230
character constants
  specifying with CONSTRULES 425
  string rules 217
  translation 264
character data
  collating sequence 139, 219
  conversions (translation) supported 262
  data types 230, 232
  translation 261
characters, multiple-byte 73
characters, special 73
classes, SMS 195, 304, 456
cleaning up unused data sets 173
CLOB data type 244
CLOBF keyword of character data types 231, 233, 366
CLONE option 97
clone tables 97, 211
clustering key, ordering by 139, 321, 408
CMAX option 148, 425
CMRATIO option
  calculating HURBA with formula 125
  installation 425
  with ANALYZE option 125
CNTLCARDS options
  BMCLoad keyword 133, 295, 505
  DB2
    LOAD keyword 366
  DB2 keyword
    description 132, 503
    examples 325, 342, 360
  DB2DDL keyword 135, 508
  DB2LOAD keyword 131, 501
description 130
  EASYTRIEVE keyword 135, 518
easytrievE examples
  DB2 325, 342, 360
  DB2LOAD 366
default 381
  SAS 332
  FOCUS keyword 136, 509
  inserting additional command options using the string parameter 131, 500
NOMAD keyword 136, 528
performance considerations 401
SAS keyword 136, 332, 511
specifying SYSCNTL data set 284
SQL/DS keyword 137, 516
SQL/DS-DDL keyword 137, 514
SQL/DS-LOAD keyword 137, 518
TERADATA keyword 138, 528
TERADATA-BULK keyword 138, 522
TERADATA-FAST keyword 138, 525
TERADATA-MULT keyword 139, 526
CNTLDDN option
  description 139
  examples 342
  specifying SYSCNTL data set 284
COBOL II
  sample user exit 582
  UNLOADEXIT COBOL2 option 169
codes
  function 533
  return 170, 173, 533
  SQL 440
column-level security 29, 205
columns
comparing value to a constant 215
definitions when using FORMAT BMCLOAD 68
identity 70
IMPLICITLY HIDDEN 207
timestamp, with precision 17, 242
timestamp, with time zone 17, 164
unloading selected 321
comma as decimal point 217
comma as separator 217
command options
descriptions 96
diagrams 84
examples 311
performance 401
rules for syntax 79
comparison operators 213, 215
components that UNLOAD PLUS uses 38
compression
BMCDICT table 472
defining for tape data sets 203
dictionary, objects containing 113, 115
ratio, specifying 425
concatenated copy data sets 115
concurrency
considerations 46
SHRLEVEL option 49, 103
concurrent partitions, number of 434
consistent data image, unloading 53
constants
AUTOTAG option 145
binary 217
character string 217
command 264
comparison 264
CONSTRULES option 425
date/time string 217
decimal 217
double-byte character set (DBCS) 205
floating point 228
graphic string 217
hexadecimal 217
IN keyword 264
integer 217
length in string 71
LIKE keyword 264
NULL keyword 219
NULLCHAR option 155
output-record tagging 26
predicate block 217, 264
RECORDID option 211
rules for 217
supported types 228
user-specified 27
with WHERE clause 217
CONSTRULES option 425
CONTINUE UTILITY keyword of ON MESSAGE option 170
contraction, during translation 263
control statements
generating 130, 499
inserting additional command options into 500
output data set specification 284
product type specification 130
controlling access
in UNLOAD PLUS 43
conventions
rules for specifying command options 79
conventions, documentation 12
conversions
data type, allowable 248
errors 145
EXIT data type for user-defined 246
floating-point and graphic constants 246
formats supported 247, 248, 259
order of 265
role of DISCARDS option 145
rounding of values 236, 237
supported types 226
TRIM and TRUNCATE options 232, 240
user-defined exit routines 246
copies, cabinet 110
copies, encrypted
  authorization requirements 42
  key data set for 433
  specifying 113
copies, encrypted, authorization requirements 43
copies, unloading from. See image copies, unloading from
copy data sets
  authorization for 42
  description 35
  specifying 279
  unloading from 110
COPY PLUS
  encrypted copies from 113
  Instant Snapshot copies 118
  key data set 433
copy registration
  BMCXCOPY table 491
cabinet copies 113
  online consistent copies 118
  snapshot copies 118
CSV keyword of FORMAT option
  description 151
  with LRECL option 192
  with USELRECL option 126
CSV output
  considerations 72
  example 76, 357
  specifying 151
CURRENT DATE option 207, 219, 229
current date variables 186, 450
CURRENT RID option 208
CURRENT TIME option 207, 229
current time variables 186, 450
CURRENT TIMESTAMP option 207, 219, 229
CURRENTDEGREE option 102, 360, 426

D
D2U (DB2 Utilities Common Code) 38, 41, 276
UNLOAD PLUS requirements 41
DASD, group 304
data
  availability 25, 103
  compression, specifying 203, 462
  consistent image 53
  conversion 246, 248
  high-speed migration 67, 389, 401
  migration, using UNLOAD PLUS and LOADPLUS 67, 389, 401
  sampling 144
  translation 261, 401
Data Analytics Accelerator for z/OS 24
data classes, SMS 195, 456
data conversions
  allowable 248
  numeric considerations 260
  with respect to data translation 265
data sets
allocating dynamically 122, 442
authorization 43
calculating size with ANALYZE option 302
cleaning up unused 173
compression for tape data sets 203
concatenated 115
copy
description 35
specifying 279
unloading 110
DB2, controlling buffer usage 404
DDLIN 278
deleting 173
description 35
disk 197
dynamic allocation 122, 442
expiration date, setting 205
extended sequential 189, 453
key, for encrypted copies 433
large format 189, 453
LBI 404, 432
logical record length, specifying 192
multiple 289, 409
options for disk and tape 183
output 286
overriding default ddnames 119, 121
overview 35
record format for output, specifying 128, 436
referenced files for LOB and XML data
specifying 287
specifying directory size 190
unloading to 55
retention period, setting 204
setting maximum number in a data group 193
sizing using FILESZPCT option 200
SORTWK 281
space, adjusting when using automatic sizing 200
SYSCNTL 133, 284
SYSREC 286
SYSRED 286
tape 203
unload, multiple 409
VBS, for spanned-record output 58, 129, 286
data sets, UNLOAD PLUS
BMCFORCE 278
data sharing
when DB2 versions are not the same 273
when using table space compression 273
data translation
BIGINT 237
BIGINT EXTERNAL 237
BINARY 240
BINARY VARYING 240
BLOB 244
BLOBF 230, 233
CHAR 230, 259
CLOB 244
CLOBF 231, 233
conversions 248, 265
DATE 241
DATE EXTERNAL 241
DATE-format EXTERNAL 241
date/time considerations 261
date/time formats 247
DBCLOB 244
DBCLOBF 231, 234
DECFLOAT 165, 245, 259
DECIMAL 238
DECIMAL EXTERNAL 238
DECIMAL PACKED 238
DECIMAL ZONED 238
default formats 226
default lengths 248
description of each 229
EXTERNAL 259, 261
FLOAT 239
FLOAT EXTERNAL 239
GRAPHIC 235
INTEGER 236
INTEGER EXTERNAL 236
keywords 229
numeric considerations 260
numeric EXTERNAL considerations 259
option (with SELECT) 226
ROWID 73, 261
SMALLINT 236
SMALLINT EXTERNAL 236
specifying 226, 229
TIME 243
TIME EXTERNAL 243
TIME-format EXTERNAL 243
TIMESTAMP 242
TIMESTAMP EXTERNAL 242
timezone 164, 242
TIMESTAMP WITH TIME ZONE EXTERNAL 242
timestamp-format EXTERNAL 242
user-defined 29
VARBINARY 240
VARCHART 232, 261
VARGRAPHIC 235
XML 245
data-sorting indexes 139
Database Administration (for DB2) solution 23
database name variable 186, 450
DATA CLAS option 195, 456
date
CURRENT DATE option 207
data type conversions 248
data types 241
exit routine 77
expiration date for current disk data set 203
expiration date for current unload data set 205
formats 247
overriding DB2 date exit routine 160
rules for constants 217
specifying current 219
variables 186, 450
DATE data type 241
DATE EXTERNAL data type 241
DATE-format EXTERNAL data type 175, 241
DATEFMT option 160
DB2
authority
UNLOAD PLUS 43
authorization needed to execute UNLOAD PLUS 42
catalog, unloading from 63
control statements 131, 501
CREATE TABLE DDL generation 132, 508
data sets 404
features, unsupported 78
internal format, output in
FORMAT INTERNAL 67, 150
FORMAT STANDARD 66, 150
parallelism 360
platforms, unloading data to non-DB2 401
security exit 42
subsystem identifier (SSID) parameter of EXEC
statement 273
system setup 41
user exits 68, 77, 407
versions supported 41
versions, when different on data sharing group 273
DB2 Data Analytics Accelerator 182
DB2 Solution Common Code
UNLOAD PLUS requirements 41
DB2 Solution Common Code (SCC) 38
DB2 Utilities Common Code (D2U) 38, 41, 276
UNLOAD PLUS requirements 41
DB2STATS keyword of ANALYZE option 124
DBCLOB data type 244
DBCLOBF keyword of character data types 231, 234
DBCS (double-byte character set), support for 205
DD statements
SORTPARM 308
allocating data sets
using DD statements 276
BMCFORCE 278
data sets
allocating 276
description 276
specifying DD statements 276
ddnames
descriptions of data sets 276
DSN1COPY 279
image copy 279
SORTWK 281
specifying 276
specifying DD statements 276
STEPLIB 276
SYSCNTL 284
SYSIN 285
SYSPRIN2 285
SYSPRINT 285
SYSREC 286
SYSSRED 286
UTPRINT 307
VSAMDD 281
DDL, using for DB2 object definitions with INFILE
ddname 278
DDLDDN option 121
DDLIN
data set 35, 278
eample 353
overriding the default ddname 121
ddnames
data sets for DIRECT NO 35
data sets for DIRECT YES 35
specifying for the control statement data set 139
specifying for the DD input data set 121
specifying for the output data set 121, 442
with INFILE option 110, 115
DECFLOAT data type 165, 245, 259
DECFLOAT_ROUNDMODE option 165
decimal constants 217
DECIMAL data type 238
decimal data, zoned 165, 238
decimal floating-point data type 165
DELETEFILES option 173
deleting work files
examples 363, 389
specifying 173, 426
DELFILES option 426
deliminated tokens 79
delimiting characters
for CSV format 151
for FORMAT CSV 151
for FORMAT XML 152
DESC keyword of ORDER BY option 219
dictionaries, compression
BMCDICT table 472
objects containing 113, 115
DIR option 190
DIRECT option
AUTO 101, 139
description 28, 98
NO
data sets 35
description 100
examples 360, 363, 366
number of DB2 threads 45
processing phases 33
relation to ANALYZE processing 123
resolving symbolic variables 188
serialization and concurrency issues 51
processing phases 31
YES 31
data sets 35
description 99
example 366
number of DB2 threads 45
relation to AUTO option 101
serialization and concurrency issues 48
view restrictions 211
DISABLED keyword of ZIIP option 106, 445
disabling zIIP processing 106, 445
discarded records 145
DISCARDS option 145
disk data sets
command options for dynamic allocation 197
expiration date, specifying 203, 461
installation defaults for dynamic allocation 457
retention period, specifying 202, 460
disk space, specifying 457, 459
disk volumes, specifying 460
DISKEXPD option 203, 461
DISKRETN option 202, 460
DISPLAY privileges 43
displaying drain locks 178
displaying status of BMC utilities 471

disposition
and deleting files 173, 389, 426
SYSREC data sets 288
tape files 441
documentation information 11
double-byte character set. See DBCS

drain
canceling threads for 179, 430
displaying locks 178, 429
number of retry attempts 427
processing description 51
read claimers, canceling 180, 430
specifying the number of retry attempts 177
specifying the time between retry attempts 178
time between retry attempts 427
timeout value 175, 428
write claimers, canceling 180, 430

DRAIN_WAIT option 175
DRNDELAY option 427
DRNFAIL keyword of DSPLOCKS option 178
DRNRETRY option 427
DRNWAIT option 428
DSECT, assembler exit parameter block 538
DSN1COPY
buffer specification for reading 404
compression dictionary limitation 115
data sets 26
source of unload 110
DSN1COPY table space status 49
DSNAME option
command 184
example 366
installation 450
substitution variables in 295
DSNDB04 database name 97
DSNHDECP
copy loaded from STEPLIB 433
default values at installation 158
printing values 274
settings for changing decimal points to comma 217
SSID from 273
time zone value 164
DSNTIAUL format, output in 71, 151
DSNTIAUL keyword of FORMAT option
description 71, 151
example of unloaded data 76
relation to FIXEDVARCHAR option 167
relation to NULLCHAR option 155
DSNTYPE option 189, 366, 453
DSNZPARMS, query acceleration 182
DSNZPARMS, utility timeout value 175
DSPLOCKS option 429
command 178
dump, system 429, 437
DYNALLOC option, RETRY parameter 463
DYNALOC installation option 142, 463
dynamic allocation
activating and configuring 122, 442
benefits 52
BMCSORT allocation 281
calculating data set size 302
command options 182
common defaults 449
data set type 189, 453
disk defaults 457
examples 296, 373
extended sequential data sets 189, 453
installation options 447
large format data sets 189, 453
sizing unload data sets 422
SMS 52
syntax options 182
tape defaults 461
UNLOADDN_ACTIVE option 294, 399, 447
using to avoid complex JCL 25
dynamic SQL
DIRECT option 100
example 360
ISOLATION levels 51
maximum number of tasks 148, 425
overview 29
specifying 98

E
Easytrieve 135, 518
EASYTRIEVE keyword of CNTLCARDS option 135, 518
EBCDIC
data translation 261
exit routines 77
option 158
EDITPROCs 68, 77, 444
<table>
<thead>
<tr>
<th>eDSNHDEC</th>
<th>elapsed time</th>
</tr>
</thead>
<tbody>
<tr>
<td>date and time format 248</td>
<td>increased due to data set size 282</td>
</tr>
<tr>
<td>reducing</td>
<td>using FORMAT BMCLOAD 150</td>
</tr>
<tr>
<td></td>
<td>using multiple unload data sets 409</td>
</tr>
<tr>
<td></td>
<td>using multitasking 407</td>
</tr>
<tr>
<td>electronic documentation 11</td>
<td>ENABLED keyword of ZIIP option 106, 445</td>
</tr>
<tr>
<td>enabling dynamic allocation 122, 442</td>
<td>enabling zIIP processing 106, 445</td>
</tr>
<tr>
<td>ENCLOSERBY option of FORMAT option 73, 151</td>
<td>encrypted copies</td>
</tr>
<tr>
<td></td>
<td>authorization requirements 42</td>
</tr>
<tr>
<td></td>
<td>key data set for 433</td>
</tr>
<tr>
<td></td>
<td>specifying 113</td>
</tr>
<tr>
<td>encrypted copies, authorization requirements 43</td>
<td>encrypted data 78</td>
</tr>
<tr>
<td>ENUMROWS option</td>
<td>description 141</td>
</tr>
<tr>
<td></td>
<td>examples 342, 347</td>
</tr>
<tr>
<td></td>
<td>performance considerations 401</td>
</tr>
<tr>
<td>error byte 265</td>
<td>error handling</td>
</tr>
<tr>
<td>IF ERROR option 228</td>
<td>using ON FAILURE to recover from abnormal termination 173</td>
</tr>
<tr>
<td>ERROR keyword of IF option 228</td>
<td>ERRORBYTE fields 265</td>
</tr>
<tr>
<td>ESCAPE keyword of predicate 216</td>
<td>ESTROWS option</td>
</tr>
<tr>
<td>description 221</td>
<td>determining data set size during dynamic allocation 302</td>
</tr>
<tr>
<td></td>
<td>determining number of records to allocate 304</td>
</tr>
<tr>
<td>example UNLOAD PLUS jobs</td>
<td>clustering order, unloading in 321</td>
</tr>
<tr>
<td>DELETEFILES processing 363</td>
<td>dynamic allocation of data sets, unloading 373</td>
</tr>
<tr>
<td>dynamic SQL, unloading 360</td>
<td>entire table, unloading 314</td>
</tr>
<tr>
<td>FORMAT CSV option 357</td>
<td>full image copy, unloading 335</td>
</tr>
<tr>
<td>incremental image copy, unloading 350</td>
<td>LOADPLUS for DB2, unloading data to 381, 389</td>
</tr>
<tr>
<td>LOB data, unloading 366</td>
<td>multiple SELECT statements 342</td>
</tr>
<tr>
<td>multiple tables, unloading 325</td>
<td>nonstandard null indicators, unloading 366</td>
</tr>
<tr>
<td>partitioned table space, unloading 314</td>
<td>prior image copy, unloading 350</td>
</tr>
<tr>
<td>selected columns, unloading 321</td>
<td>table space, unloading 381</td>
</tr>
<tr>
<td>table space, unloading 381</td>
<td>termination, handling an abnormal 363</td>
</tr>
<tr>
<td>termination, handling an abnormal 363</td>
<td>XML data, unloading 366</td>
</tr>
<tr>
<td>example user exits 531</td>
<td>EXCLDUMP option 429</td>
</tr>
<tr>
<td>EXEC statement 272</td>
<td>EXECUTE privileges</td>
</tr>
<tr>
<td>EXECUTE privileges</td>
<td>UNLOAD PLUS 43</td>
</tr>
<tr>
<td>executing UNLOAD PLUS jobs 308</td>
<td>execution phases of UNLOAD PLUS 31, 33</td>
</tr>
<tr>
<td>EXIT data type 246</td>
<td>exit parameters</td>
</tr>
<tr>
<td>exit parameters</td>
<td>assembler DSECT 538</td>
</tr>
<tr>
<td></td>
<td>C structure 550</td>
</tr>
<tr>
<td></td>
<td>LE C structure 566</td>
</tr>
<tr>
<td></td>
<td>record for COBOL II AND LE COBOL 582</td>
</tr>
<tr>
<td>exit routines</td>
<td></td>
</tr>
</tbody>
</table>

Index 605
FETCH, specifying multiple rows for one request
field specification options
FIELDPROCs
how UNLOAD PLUS invokes with FORMAT BMCLOAD option
fields
adding
assigning names for new constant type
binary
century specification
character
constant specification
converting from columns
data type specification
definitions of the output record
delimiting
DSECT
LOB data
mixed double-byte character set (DBCS)
nonnumeric
null indicator
nullable
output record identification
output with NULL values
pseudo-SQLDA
separating character
specification option, with SELECT
SQLDA
SQLTYPE
SQLVAR
TRIM and TRUNCATE options
VARCHAR
variable-length
with FORMAT BMCLOAD
XML data
file referencing. See referenced files
FILEREFDN option
FILESZPCT option
FILL option
command 157
installation 430
of BIGINT data type 237
of CHAR data type 230
of DECIMAL data type 238
of INTEGER data type 236
of SMALLINT data type 236
of VARCHAR data type 233
filtering partitions 109, 401
FILTERPART option
description 109
limits opening of data sets 409
performance considerations 401
fixed-length output records 71, 167
FIXEDVARCHAR option
description 167
example 321
performance considerations 401
fixes applied, generating a report of 274
FlashCopy image copy 110
FLOAT data type 239
floating-point data types 239, 245
FOCUS keyword of CNTLCARDS option 136
FORCE option 179, 430
FORCE_AT option
command, of FORCE option 180
installation 431
FORCE_DELAY option 181
FORCE_RPT option 432
FORCE_RPT option of FORCE option 181
FORMAT options
BMCLoad keyword 67, 150, 401
BMCULOAD keyword (in LOADPLUS) 389
CSV keyword 74, 151
description 149
DSNTIAUL keyword 71, 151, 401
examples 76, 357, 389
EXTERNAL keyword 66, 150, 401
INTERNAL keyword 67, 150, 401
STANDARD keyword 66, 150, 401
XML keyword 74, 152
formats, floating point 165
FROM option 211
nullable fields 314
full image copies, unloading from 49, 113
FULL option 115
FULL keyword
of INFILE ddname option 115
of INFILE IMAGECOPY option 113
FULL option 113

G
GDGEMPTY option 455
GDGLIMIT option 193, 455
GDGs (generation data groups)
for unload data sets 295
limiting number of versions 193, 455
removing from the VTOC 455
uncataloging 455
GDGSCRATCH option 455
GENERATED ALWAYS columns 100, 261
generating control statements
DB2 LOAD 131, 132, 501
Easytrieve 135, 518
FOCUS 136
LOADPLUS 133, 501, 505
NOMAD 136, 528
SAS 136, 511
SQL/DS tables 137, 514
Teradata tables 138, 520
generating DDL statements for DB2 tables 135, 508, 518
generation data groups. See GDGs
global trim option, with FORMAT CSV 151
graphic characters 205
GRAPHIC data type
allowable data type conversions 248
description 235
relation to LIKE operator 215
graphic strings rules for constants 217
GRECP (group RECOVER pending) status 49
group RECOVER pending (GRECP) status 49
guaranteed space 304

H
hardware compression
BMCDICT table 472
hardware copies, unloading from 118
hash-organized tables 29
hexadecimal strings rules for constants 217
HFS files, for LOB and XML data
allocating 287
specifying 189, 453
unloading to 55
HFS keyword of DSNTYPE option 189, 453
hierarchical file system. See HFS
High Speed Utilities (for DB2) solution 23
high-speed data migration 150, 389
high-transaction environment 411
high-used RBA 302
HISTORY installation option
   BMCHIST table 474
description 432
HISTRETN installation option
   BMCHIST table 476
HURBA keyword of ANALYZE option 125, 302

I
I/O tuning 410
IBM z Integrated Information Processor (zIIP). See zIIP processing
IBUFFS option 404, 432
ICTYPE column value 118
ID
   utility (parameter of EXEC statement) 274
identity columns, with FORMAT BMCLOAD 70
IDs
   RACF 43, 436
      subsystem 108, 444
      user 186
   utility 186
   XBM 108, 444
IF NULL VALUE option 321
IF option with SELECT 227
image copies, unloading from 49, 110, 432
   base tables 57
   buffers for 404
   cabinet copies 110
   compression dictionary limitation 113, 115
   copy before most recent 113
   different subsystems 115
   encrypted 113
   examples 335, 350
   FlashCopy 110
   full 113, 115
   incremental 113, 115
   INFIL option 110
   inline 115, 433
      Instant Snapshot copy 110
      online consistent copy 118
   SHRLEVEL CHANGE 104
   SHRLEVEL REFERENCE 103
IMAGECOPY option 113
   implicit time zone 164
   IMPLICITLY HIDDEN columns 207
   IN option, with WHERE clause 215
   incompatible options, when unloading LOB or XML data 57, 60
   incremental image copies
      example 350
      specifying 113, 115
   INCREMENTAL keyword
      of INFILE ddname option 115
      of INFILE IMAGECOPY option 113
   indexes
      ordering by 139
      versioned 68
   INDSN option 389
   INFIL option
      DDL specification 278
      ddname specification 279, 281
description 110
   examples
      IMAGECOPY 335
      IMAGECOPY INCREMENTAL 350
      sequential data set 332, 353
      relation to ANALYZE processing 123
      SNAPCOPY option 118
   inline image copies, unloading from 115, 433
   inline LOB data 29
   INLINE option 115, 433
   input data set 285
   input data set, specifying the number of buffers 432
   installation options
      BMCHIST 474
      definitions and default values 418
      dynamic allocation 447
      HISTORY 474
      HISTRETN 476
      multiple modules 417
      output descriptor 449
      setting for optimal performance 399
      specifying a module 276
   Installation System 417
   Instant Snapshot copy
      authorization requirement 42
      specifying 110
   Instant Snapshot copy, system authority needed 43
   integer constants, rules for 217
   INTEGER data type 236
   internal BMC format, output in 67, 150
   internal DB2 format, output in
      FORMAT INTERNAL 67, 150
      FORMAT STANDARD 66, 150
INTERNAL keyword of FORMAT option
  description 67, 150
  example 76
INTERVAL option 144, 329
INTO option
  description 208
  example 314
  inserting additional command options using
    the string parameter 500
invoking UNLOAD PLUS 308
IRLMWAIT, used with drain timeout 175
IS NOT NULL operator, with WHERE clause 215
IS NULL operator, with WHERE clause 215
ISOLATION levels for DB2 dynamic SQL 51

J
JCL
  building UNLOAD PLUS jobs 271
  DD statements 276
  disposition for SYSREC data sets 288
  examples 311
  EXEC statement 272
  REGION parameter 410
JES3 limitation 307
JOB statement 271
  jobs
    building 271
    canceling 309
    examples 311
    executing 308
    restarting 308
    terminating 309
Julian date variables 186, 450

K
key data set, for encrypted copies 433
KEYDSNAM option 433
  keywords
  $NO_UNITS$ keyword of UNIT option 183
  $NO_UNITS$ option 449
ACTIVE option 122
AGE keyword of ANALYZE DB2STATS option 124
ALL keyword
  of FORCE option 180, 430
ANALYZE option 123, 422
ANY keyword of CURRENTDEGREE option 102
ASC keyword of ORDER BY option 219
ASCII option 158
ASM keyword of UNLOADEXIT option 169
AUTO keyword
  of DIRECT option 101
  of FORMAT XML SELECT_ELEMENT option 152
  of NBRSECD option 200
  of PCTPRIM option 197
  of RECFM option 128, 436
AUTOTAG option 145
BASIC keyword of DSNTYPE option 189, 453
BIGINT data type 237
BINARY data type 240
BINARY VARYING data type 240
BINARYXML keyword
  of CHAR data type 230
  of VARCHAR data type 233
BLOB data type 244
BLOBF keyword of character data types 230, 233
BMCLOAD keyword
  of CNTLCARDS option 133
  of FORMAT option 150
BYTASK keyword of ORDER YES option 139
C keyword of UNLOADEXIT option 169
CCSID option 159
CENTURY option
  command 175
  installation 423
  of DATE data type 241
  of TIMESTAMP data type 242
CHANGE keyword of SHRLEVEL option 104
CHANGE_CONSISTENT option 423
CHANGE_QUIESCE option 424
CHAR data type 230
CLOB data type 244
CLOBF keyword of character data types 231, 233
CLONE option 97
FORCE_DELAY option of FORCE option 181
FORCE_RPT option
  command, of FORCE option 181
  installation 432
FORMAT option 149
FROM option 211
FULL keyword
  of INFILE ddname option 115
  of INFILE IMAGECOPY option 113
GDEMPTY option 455
GDGLIMIT option 193, 455
GDGSCRATCH option 455
GRAPHIC data type 235
HFS keyword of DSNTYPE option 189, 453
HISTORY option 432
HURBA keyword of ANALYZE option 125
IBUFFS option 432
IF option 227
IMAGECOPY option 113
IMPLICIT_TZ option 164
INCREMENTAL keyword
  of INFILE ddname option 115
  of INFILE IMAGECOPY option 113
INFILE option 110
INLINE option 115, 433
INTEGER data type 236
INTERNAL keyword of FORMAT option 150
INTERVAL option 144
INTO option 208
KEYDSNAM option 433
keywords
  SUFSTART keyword of WТОMSG option 108
LARGE keyword of DSNTYPE option 189, 453
LASTRETRY keyword of FORCE_AT option 180, 431
LE_C keyword of UNLOADEXIT option 169
LE_COBOL keyword of UNLOADEXIT option 169
LIBRARY keyword of DSNTYPE option 189, 453
LIKE operator 215
LIMIT option 144
LOADDECP option 433
LOCKROW option 434
LOGICAL option 119, 222
LRECL option 192
MAINT parameter of EXEC statement 274
MAXBLKSIZE option 127
MAXCONNECT option 148
MAXP option 434
MAXPRIM option 198, 458
MAXSECD option 199, 458
MAXSORTS option 147
MGMTCLAS option 195, 456
MIXED keyword
  of CHAR data type 230
  of VARCHAR data type 232
MSGLEVEL option 434
NAME option 209
NBRSECD option 200, 459
NEW parameter of EXEC statement 274
NOCOMP keyword of TRTCH option 203
NOLIMIT keyword of ANALYZE DB2STATS
  AGE option 124
NOMAD keyword of CNTLCARDS option 136
NONE keyword
  of CURRENTDEGREE option 102
  of DRAIN_WAIT option 175
  of DSNTYPE option 189, 453
  of DSPLOCKS option 178
  of FORCE option 180, 430
  of FORMAT XML SELECT_ELEMENT
    option 152
  of TRTCH option 203
NOSUBS option 160
NULL keyword
  of IF option 228
  of predicate 219
  of VALUE option 229
NULLCHAR option 155, 435
NULLSTRING option of FORMAT option 151, 152
NULLTYPE option 156, 435
OBID option 213
ON FAILURE option 171
ON MESSAGE option 170
OPNDB2ID option 436
OPTIONS option 221
ORDER BY option 219
ORDER option 139
OUTPUT command 182
PACKED keyword of DECIMAL data type 238
PARM keyword of EXIT data type 246
PART option 119, 222
PCTPRIM option 197, 457
PDS keyword of DSNTYPE option 189, 453
PLAN option 436
QUIESCE option 104
RECFM option 128, 436
RECORDID option 211
REFERENCE keyword of SHRLEVEL option 103
REPORTONLY keyword of FORCE option 181
RETCODE keyword of ON FAILURE TERMINATE
UTILITY option 173
of ON MESSAGE CONTINUE
UTILITY option 170
RETPD option 204, 462
RETRY keyword of FORCE_AT option 180, 431
RETRY keyword of DSPLOCKS option 178
RETRY option 177
RETRY_DELAY option 178
ROUND keyword of BIGINT data type 237
of DECIMAL data type 238
of FLOAT data type 239
of INTEGER data type 236
of SMALLINT data type 236
ROUND_CEILING keyword of
DECFLOAT_ROUNDMODE option 165
ROUND_DOWN keyword of
DECFLOAT_ROUNDMODE option 165
ROUND_FLOOR keyword of
DECFLOAT_ROUNDMODE option 165
ROUND_HALF_DOWN keyword of
DECFLOAT_ROUNDMODE option 165
ROUND_HALF_EVEN keyword of
DECFLOAT_ROUNDMODE option 165
ROUND_HALF_UP keyword of
DECFLOAT_ROUNDMODE option 165
ROUND_UP keyword of
DECFLOAT_ROUNDMODE option 165
ROWSETSZ option 101, 437
RTRIM option 151
SAS keyword of CNTLCARDS option 136
SDUMP option 437
SELECT statement 205
SELECT_ELEMENT option 152
semicolon 223
SET CURRENT QUERY ACCELERATION
ALL option 182
SET option 182
SHRLEVEL option 103, 438
SMALLINT data type 236
SMAX option 438
SMCORE option 439
SNAPCOPY option 118
SORTDEVT option 142
SORTNUM option 142, 440
SPACE option 197, 457
SPANned option 129
SQL keyword of DRAIN_WAIT option 175
SQL/DS keyword of CNTLCARDS option 137
SQL/DS-DDL keyword of CNTLCARDS option 137
SQL/DS-LOAD keyword of CNTLCARDS option 137
SQLDELAY option 440
SQLRETRY option 441
STANDARD keyword of FORMAT option 150
START keyword of FORCE_AT option 180, 431
STOP UTILITY keyword
of ON FAILURE option 173
of ON MESSAGE option 170
STORCLAS option 194, 456
SUBSETS option 191
SUFSTART keyword of WTOMSG option 108
SYNC option 148
TABLESPACE option 97
TAPEDISP option 441
TAPES option 441
TASKMAX option 441
TERADATA keyword of CNTLCARDS option 138
TERADATA-BULK keyword of CNTLCARDS option 138
TERADATA-FAST keyword of CNTLCARDS option 138
TERADATA-MULT keyword of CNTLCARDS option 139
TERM parameter of EXEC statement 274
TERMINATEDBY option of FORMAT CSV option 151
TIME data type 243
TIMEFMT option 161
TIMESTAMP data type 242
TIMESTAMP WITH TIME ZONE data type 242
TRIM option
of BINARY data type 240
of CHAR data type 232
of VARBINARY data type 240
of VARCHAR data type 234
TRTCH option 203, 462
TRUNCATE option
 of BINARY data type 240
 of CHAR data type 232
 of GRAPHIC data type 235
 of VARBINARY data type 240
 of VARCHAR data type 235
 of VARGRAPHIC data type 235
TSFMT option 162
UBUFFS option 441
UNICODE option 158
UNIT option 183, 449
UNITCNT option 196, 457
UNLDMAX option 442
UNLOAD command 97
UNLOADDN option 121, 442
UNLOADDN_ACTIVE option 442
UNLOADEXIT option 169
USELRECL option 126, 443
UTIL keyword of DRAIN_WAIT option 175
UXSTATE option 444
VALUE option 228
VARBINARY data type 240
VARCHAR data type 232
VARGRAPHIC data type 235
VB keyword of RECFM option 128, 436
VOLCNT option 193, 454
VOLUMES option 201, 460
VSAMDD option 118
VSAMDDPREFIX option 119
WHERE option 213
WORKUNIT option 444
WTOMSG option 108
XBMID option 108, 444
XML data type 245
XML keyword of FORMAT option 152
ZIIP option 106, 445
ZONED keyword of DECIMAL data type 238
ZONEDDECOVP option 165, 446
LASTRETRY keyword of FORCE_AT option 180, 431
LBI data sets 404, 432
LE_C keyword of UNLOADEXIT option 169
LE_COBOL keyword of UNLOADEXIT option 169
length option
 with BINARY data type 240
 with CHAR data type 231
 with DATE EXTERNAL data type 241
 with referenced file subtypes 231, 234
 with TIME EXTERNAL data type 243
 with TIMESTAMP EXTERNAL data type 242
 with VARBINARY data type 240
 with VARCHAR data type 234
 with VARGRAPHIC data type 235
LIBRARY keyword of DSNTYPE option 189, 453
LIKE operator 215
limit key specifications in DDL 115
LIMIT option
calculating data set size 302
description 144
determining number of records to allocate 304
examples 321, 329
limiting retry attempts 427, 441
line feed characters, in FORMAT CSV options 73
load utilities, control statements 130, 209
LOADDECP option 433
LOADPLUS
control cards, from UNLOAD PLUS 295, 401
data types 230, 231, 233, 234
discards due to length, limiting 145
determining number of records to allocate 304
examples 321, 329
field specification 54, 55
inserting additional command options into
LOAD control cards 500
loading CSV data 74
LOB data
BMCSYNC table considerations 481
data types 244
data types for referenced files 230, 231, 233, 234
discards due to length, limiting 145
determining number of records to allocate 304
examples 321, 329
field specification 54, 55
inserting additional command options into
LOAD control cards 500
loading CSV data 74
LOB data
BMCSYNC table considerations 481
data types 244
data types for referenced files 230, 231, 233, 234
discards due to length, limiting 145
determining number of records to allocate 304
examples 321, 329
field specification 54, 55
inserting additional command options into
LOAD control cards 500
loading CSV data 74
lock row option 434
locks, displaying 178
locks, table 51, 440, 441
language for user exits 169
large format sequential data sets 189, 453
LARGE keyword of DSNTYPE option 189, 453
log range table 477
LOGICAL option 119, 222
logical page list (LPL) status 49
logical part REBUILD pending (RBDP*) status 49
logical partitions 119, 222, 450
long object names 79
LPL (logical page list) status 49

M

MAINT parameter 274
maintaining common utility tables 469
management classes, SMS 195, 456
management set, XBM 53
MAXBLKSIZE option 127, 401
MAXCONNECT option 148, 425
maximum disk allocation
  primary 198, 458
  secondary 199, 458
MAXP option 434
MAXPRIM option
  command 198
  installation 458
  relation to specifying guaranteed space
    algorithm 304
  relation to standard SMS algorithm 304
  specifying when calculating space for SMS-
    managed data sets 304
  using to override the PCTPRIM calculations 197
MAXSECD option
  command 199
  installation 458
  relation to FILESZPCT option 200
  relation to SPACE option 199
MAXSORTS option
  concurrent sort reduction 407
  contention reduction 205
  description 147
  relation to SMAX installation option 438
  sort processing 415
MEMBER CLUSTER, table spaces defined with 16
MEMLIMITER system parameter 45, 410
memory
  above the bar 45
  memory allocation for sort processing 405
    performance considerations 282
message level

EXEC statement parameter 275
installation option 434
messages
  BMCSORT, data set for 307
  changing output displayed 275, 434
  MSGLEVEL 275, 434
  performance-related 412
  viewing in worklist environment 285
MGMTCLAS option 195, 456
migration, high-speed DB2 to DB2 150, 389, 401
mixed data, translation 264
MIXED keyword
  of CHAR data type 230
  of VARCHAR data type 232
modes of execution 28
MODESET SVC, impact on performance 407
MSGLEVEL
  EXEC statement parameter 275
  installation option 434
multi-row fetch 101, 437
multi-table table spaces 115
multiple SELECT statements 115
multiple-byte characters, in FORMAT CSV options 73
multitasking
  description 407
  effect on ENUMROWS option 141
  example 321
  parameters information 441
MVS cancel command 309

N

NAME option 209
names of common utility tables, determining 471
names of objects, long 79
NBRSECD option
  command 200
  installation 459
  relation to FILESZPCT option 200
  standard SMS algorithm 304
new line characters, in FORMAT CSV options 73
NEW parameter of EXEC statement 274
NOLIMIT keyword of ANALYZE DB2STATS AGE
  option 124
NOLIMIT value of MEMLIMITER system parameter 45, 272, 410
NOMAD keyword of CNTLCARDS option 136, 528
NONE keyword
of DSNTYPE option 189, 453
of DSPLOCKS option 178
of FORCE option 180, 430
nonpartitioned table spaces 113, 118, 119
NOSUBS option 160
NOT IN operator 215
NULL
   keyword of IF option 228
   value in FORMAT CSV option 151
   value in FORMAT XML option 152
null indicator
   character, position, and length 27
   field 435
NULLCHAR option 155, 435
NULLSTRING option 151, 152
NULLTYPE option 156, 435
   when to use if column is nullable 226
nullable fields 501, 505
NULLCHAR option 155, 435
NULLSTRING option of FORMAT option 151, 152
NULLTYPE option 156, 435
   number of CPUs 415
   number of tasks 25
   number of units, specifying 196, 457
   number of volumes, specifying 193, 454
numeric data types, converting to external 157, 430

0

OBID option 213, 353
OBID, specifying
   for INFILE ddbname 115
   in DDLIN 278
   with OBID option 213
Object Administration (for DB2) solution 23
object availability, SHRLEVEL option 103, 438
object names, long 79
object status 48
ON FAILURE option 171, 363
ON MESSAGE option 170
online consistent copy, specifying 118
operating system requirements 41
operators 213, 215
OPNDB2ID installation option, effects on RACF
   authorizations 43
   OPNDB2ID option 42, 436
optimizing
CMAX option 425
memory amount 405
memory use 414
performance 399
primary and secondary extent sizes 196
SMAX option 438
tables in segmented table spaces 409
tasks during sort processing 412
options, command
   dynamic allocation 182
   field specification 223
   OUTPUT command 182
   setting for optimal performance 401
   unload processing 80
   See also keywords
options, incompatible, when unloading LOB or
   XML data 57, 60
options, installation
   definitions and default values 417
   multiple modules 276, 417
   setting for optimal performance 399
   specifying module 276
ORDER BY option 219
ORDER option
   description 139
   examples 321, 325
output
   BMC internal format 67, 150
   CSV format 72, 151
   data formats 64
   DB2 internal format
      FORMAT INTERNAL 67, 150
      FORMAT STANDARD 66, 150
   device default 449
   DSNTIAUL format 71, 151
   examples of format 76
   external date/time data 66, 150
   external format 66, 150
   for LOADPLUS 67
   format 64, 149
   internal date/time data 67, 150
   records, tagging 145
   specifying format 149
   XML format 74, 152
OUTPUT command 182
output data sets
default ddname, specifying 442
multiple 289
record format, specifying 128, 436
referenced files 287
SYSREC option 286
output descriptors
&PART variable 296
&SELENUM variable 296
adjusting the total calculated space 200, 459
allocation units 197, 457
default volume list 201, 460
descriptor name 182, 449
expiration date, disk data sets 203, 461
expiration date, tape data sets 205, 462
file type 189, 453
installation options 449
length 231, 234
matching to field specification 60
matching to FILEREFDN 429
matching to multiple SELECT statements 296
number of volumes 193, 454
retention period, disk data sets 202, 460
retention period, tape data sets 204, 462
secondary allocation size 200, 459
SMS classes 194, 195, 456
type of file 189, 453
volume serial numbers 304
output records
identifying 209
tagging 145, 211
overpunch values, assigning 27, 165, 446

P
packages 42, 436
UNLOAD PLUS 43
PACKED keyword of DECIMAL data type 238
padding, order of 265
page set REBUILD pending (PSRBD) status 49
parameters, EXEC statement 272
PARAM keyword of EXIT data type 246
PART option
description 119, 222
examples 332, 335, 350
performance considerations 401
partition-by-growth table spaces 109, 139, 219
with ORDER BY 219
partitioned data sets, for referenced files 189, 287, 453
partitioned table spaces 219
clustering order unload example 321
ENUMROWS option 141
INFILE IMAGECOPY option 113
MAXP option 434
MAXSORTS option 147
multiple data sets 409
multitasking performance 408
partition-by-growth table spaces
with FILTERPART 109
with INFILE ddname 115
with ORDER YES 139
performance considerations 401, 408
range-partitioned 115
sorting 141
specifying the partitions to be unloaded 119
unloading to separate data sets 289
with INFILE ddname 115
partitioning key, ordering by 139, 389
partitions
logical
specifying 119, 222
with symbolic variables 186, 450
rebalanced 115
rotated 115
specifying 119, 222
PCTPRIM option
command 197
installation 457
relation to FILESZPCT option 200
relation to the guaranteed space algorithm 304
relation to the standard SMS algorithm 304
PDS keyword of DSNTYPE option 189, 453
PDS or PDSE, for referenced files 189, 287, 453
performance considerations
buffer usage, controlling 404
command options, specifying 401
DB2 user exits 407
I/O processing 410
installation options, specifying 399
LOB data 409
MAXSORTS option 147
multiple table spaces 409
multiple unload data sets 409
multitasking 407
obtaining maximum 28
related messages 412
virtual storage 410
XML data 409
phases of UNLOAD PLUS 31, 33
PLAN option 436
plans
  UNLOAD PLUS 43
porting data to other databases
    FORMAT CSV option 151
    FORMAT XML option 152
position number variable 186, 450
precision option
  of DECIMAL data type 238
  of DECFLOAT data type 245
precision, in timestamp columns 17
predicate
  of IF condition 228
  of WHERE condition 215
PRELOAD LOAD option in LOADPLUS,
specifying for performance 401
PRESERVE WHITESPACE string 131
primary data sets, dynamic allocation 442
primary table space
  allocating using PCTPRIM option 457
  calculating using the guaranteed space
    algorithm 304
  calculating using the standard SMS algorithm
    304
  overriding the calculated value with
    MAXPRIM option 197
  using PCTPRIM option to set a value for the
    percentage 197
privileges
  DISPLAY 42
  READ 42
processing phases 31, 33
processor requirements 41
product changes 14
programming language, of user exit routines 169
pseudo-SQLDA
  assembler user exit 540
  C user exit 553, 569
  COBOL user exit 584
PSRBD (page set REBUILD pending) status 49
publications, related 11

Q
query processing, offloading 182
QUERY_ACCELERATION DSNZPARM 182
QUIESCE option 49, 104

R
RACF
  security exit 42
RACF (IBM Resource Access Control Facility)
  authorization needed to execute UNLOAD
    PLUS 42
    ID 436
  OPNDB2ID option 436
RACF (IBM Resource Access Control Facility)
  authority
    UNLOAD PLUS 43
RBA, high-used (HURBA) 125
RBDP (REBUILD pending) status 49
RBDP* (logical part REBUILD pending) status 49
read claimers, canceling 180, 430
READ privileges 42
read-only (RO) status, SHRLEVEL REFERENCE
    option 103, 438
read-or-replication-only (RREPL) status 14
read/write (RW) status, SHRLEVEL CHANGE
    option 104, 438
REBUILD pending (RBDP) status 49
RECFM option 128, 436
record format, specifying 128, 436
record identifiers (RIDs)
  selecting to include in output 208
  specifying a value for 211, 342
RECP (RECOVER pending) status 49
reference files
  deleting 60
  how to unload to 55
  multiple 287, 409
  performance considerations 409
  specifying 60
  specifying name 185, 450
  specifying number to allocate 191
  specifying OUTPUT statements for 55
  specifying type 189, 453
  using 55, 287
REFP (refresh pending) status 49
refresh pending (REFP) status 49
REGION parameter 272, 410
related publications 11
REORG pending (REORP) status 49
REORP (REORG pending) status 49
report
  of applied fixes 274
  of canceled threads 181, 278, 432
REPORTONLY keyword of FORCE option 181
requirements
  software 41
  system 41
reserved words, delimiting with quotation marks 80
restart parameters 274
restart pending (RESTP) status 49
restarting
  restart parameters of EXEC statement 274
restarting UNLOAD PLUS 308
RESTP (restart pending) status 49
RETCODE keyword
  of ON FAILURE TERMINATE UTILITY option 173
  of ON MESSAGE CONTINUE UTILITY option 170
retention period
  of disk data sets 202, 460
  of tape data sets 204, 462
RETTPD option 204, 462
retry attempts, limiting 427, 441
RETRY keyword
  of FORCE_AT option 180, 431
RETRY keyword of DSPLOCKS option 178
RETRY option 177
RETRY parameter of DYNALOC option 463
RETRY_DELAY option 178
retrying drain 177
retrying dynamic allocation of SORTWK 463
return codes
  -911, specifying wait time before retry 440
  specifying 170, 173
RO (read-only) status, SHRLEVEL REFERENCE option 103, 438
ROUND keyword
  of BIGINT data type 237
  of DECIMAL data type 238
  of FLOAT data type 239
  of INTEGER data type 236
  of SMALLINT data type 236
ROUND_CEILING keyword of DECFLOAT_ROUNDMODE option 165
ROUND_DOWN keyword of DECFLOAT_ROUNDMODE option 165
ROUND_FLOOR keyword of DECFLOAT_ROUNDMODE option 165
ROUND_HALF_DOWN keyword of DECFLOAT_ROUNDMODE option 165
ROUND_HALF_EVEN keyword of DECFLOAT_ROUNDMODE option 165
ROUND_HALF_UP keyword of DECFLOAT_ROUNDMODE option 165
ROUND_UP keyword of DECFLOAT_ROUNDMODE option 165
rounding 165, 260
routines, user exit
  assembler, sample 541
  authorization 77, 531
  C, sample 555
  calling in supervisor or problem state 444
  COBOL II, sample 582
  data conversion specification 246
  data type conversion 246
  discarded records, cause of 145
  invoking DB2 user exits 77, 407
  LE C, sample 571
  LE COBOL, sample 582
  parameter block DSECT 538
  performance considerations 407
  programming language 169
  requirements 531
  specifying 169
  unload exit specification 169
  with FORMAT BMCLoad option 68
row-level security 29, 205
ROWID data type 248
ROWID data type 261
rowset-positioning cursors 101, 437
rowset, specifying 101, 437
ROWSETSZ option 101, 437
RREPL (read-or-replication-only) status 14
RTRIM option 151, 223
rules for command syntax 79
running BMC utilities concurrently 46, 484
running UNLOAD PLUS 308
RW (read/write) status, SHRLEVEL CHANGE option 104, 438
S
sample UNLOAD PLUS jobs 311
sample user exits
assembler 541
C 555
COBOL II 586
LE C 571
LE COBOL 586
where to find 532
sampling data, unloading 144
SAS control statements
example 332
specifying 136, 511
SAS keyword of CNTLCARDS option 136
scale, specifying on numeric fields 260
SCC (DB2 Solution Common Code) 38, 41, 276
SDUMP option 437
secondary allocation, specifying 199, 200
secondary data sets
description 35
dynamic allocation 182, 442
security
exit 42
mechanisms 42
required 42
requirements
authorization 42
row- and column-level 29, 205
using packages other than RACF 43
using RACF 43
verification 42
segmented table spaces 25, 409
SELECT statements
description 205
examples
columns in clustering order 321
data from DSN1COPY 332
full image copy data 335
incremental image copy 350
multiple table spaces 347
OBID and DDLIN data set use 353
partitions to separate data sets 314
prior image copy use 350
sample of catalog rows 329
multiple 57, 115, 133
multiple output data sets 292
multiple SYSCNTL data sets 284
performance considerations 401
semicolon specification when using DIRECT NO 223
sequential data sets, extended 189, 453
serialization 46
SET CURRENT QUERY ACCELERATION ALL
option 182
SET option 182
SHRLEVEL 478
SHRLEVEL option
CHANGE keyword 49, 104, 329
command 103
installation 438
REFERENCE keyword 49, 103
SMALLINT data type 236
SMAX option 407, 415, 438
SMCORE option
description 439
performance considerations 405
recommendation with large SORTNUM values 142
SMS (Storage Management Subsystem)
DATACLAS 195, 304, 456
extended sequential data sets 189, 453
managed data sets 304
MGMTCLAS 195, 456
not passing a unit value 183, 449
standard algorithm for space allocation of data sets 304
STORCLAS 194, 456
SNAPCOPY option 118
SNAPSHOT UPGRADE FEATURE (SUF)
description 53
example 329
licensing 53, 106, 445
SHRLEVEL CHANGE CONSISTENT YES 104
XBM subsystem ID 108, 444
zIIP processing 445
zIIP processing 106
software requirements 41
solution integration 23
sort processing
BMCSORT 141, 142, 281
installation options 417
memory allocation 405, 439
multitasking limitations 408
single task conditions 139
sort work data sets (SORTWK)
allocating dynamically 142, 440
calculating size 142, 440
DD statements 281
deleting 173, 426
description 35
unsupported data set types 282
SORTDEVT option 142, 325
sorting
  installation options for controlling 405, 438
memory usage 415
ORDER BY option 219
ORDER option 139
partitioned table spaces 141
restricting concurrent tasks 147
SMAX option 438
work space, calculating 282
SORTNUM option
  command 142
  installation 440
  performance considerations 401, 415
SORTWK data sets
  allocating dynamically 142, 440
  calculating size 282
  DD statements 281
  deleting 173, 426
  description 35
  unsupported data set types 282
SPACE option
  calculating data set size during dynamic allocation 302
  command 197
  installation 457
  relation to MAXSECD option 199
SPANNED option 58, 129
spanned variable block (VBS) file
  preventing with MAXBLKSIZE 127
  when specifying RECFM option 128
special characters 73
specifying options modules 276
specifying referenced files
  in OUTPUT statement 55
  matching to field specification 60
  specifying name 185, 450
SQL statements
  deleting rows from the BMCDICT table 473
  deleting rows from the BMCHIST table 477
  deleting rows from the BMCSYNC table 483, 491
  deleting rows from the BMCSYNC table for RECOVER UNLOADKEYS 483
  deleting rows from the BMCCOPY table 498
  displaying BMC utilities 471
  querying BMCHIST table 471
  querying BMCCOPY table 471
  specifying the number of times to retry statement 441
  terminating BMC utilities 471
SQL, dynamic, DB2 99
SQL/DS
  control statements, generating 516
  CREATE TABLE DDL, generating 518
  keyword of CNTLCARDS option 137
SQL/DS-DDL keyword of CNTLCARDS option 137, 518
SQL/DS-LOAD keyword of CNTLCARDS option 137, 514
SQLDELAY option 440
SQLRETRY option 441
SSID (DB2 subsystem identifier) parameter of EXEC statement 273
STANDARD keyword of FORMAT option 66, 76, 150
START keyword of FORCE_AT option 180, 431
statistics
  estimating using ANALYZE option 124
  obtaining from the DB2 catalog using DB2STATS 124
  setting a limit on the age used to calculate data set sizes 124
  using HURBA to estimate the number of rows for a table or partition 125
  using the catalog information instead of the age using NOLIMIT option 124
status of table space, initial 49
status, BMC utilities 471
STEPLIB DD statement 276
STOP UTILITY option 170
storage above the bar 45
storage groups 304
storage, virtual 45, 410
STORCLAS option 194, 456
string constants 71
striped data sets 78
structure, table, when using FORMAT BMCLOAD option 68
SUBBYTE fields 265
SUBSETS option 191, 366
subsystem ID parameter of EXEC statement 273
subsystem ID specification
  for XBM subsystem 108, 444
SUF. See SNAPSHOT UPGRADE FEATURE
summary of changes 14
supported DB2 versions 41
symbolic variables, for constructing DSNAMEs 184, 295, 450
SYNC option 148
synonyms, specifying in FROM option 211
syntax
- description of options 96
- diagrams 84
- examples 311
- rules for specifying 79

SYSCTL data sets
- DCB attributes 284
- DD statements 284
- description 35, 284
- multiple, specifying 284
- single, when specifying CNTLCARDS
  - BMCLOAD 133
  - type of control statements written to 130

SYSDISC option of DELETEFILES option (in LOADPLUS) 389

SYSIBM.SYSCOPY table 104, 113, 118
SYSIBM.SYSSTRINGS table, translation processing 261

SYSIN data set
- description 35
- output statements 182

SYSIN data sets
- specification and usage 285

SYSPRINT2 data sets
- changing message levels 275, 434
- description 35
- specification and usage 285
- worklist environment considerations 285

SYSPRINT data set
- changing message levels 275, 434
- description 35

SYSPRINT data sets
- specification and usage 285

SYSREC data sets
- block size, overriding 127
- buffer usage, controlling 405
- calculating size 288
- changing default name 121, 286
- DD statements 286
- deleting 173, 426
- description 35
- DISP parameter 288
- LOB data 54
- multiple 289
- overriding default name 286
- performance considerations 409
- record format, specifying 128, 436
- UBUFFS parameter recommendations 405
- XML data 54

SYSREC keyword of UNLOADDDN option 121

SYSRED data sets 286
See also SYSREC data sets
SYSRED keyword of UNLOADDDN option 121
system authorization
- verification 42
system authorization to execute UNLOAD PLUS 42
system dump
- excluding abend codes from 429
- generating with SDUMP option 437
system requirements 41
system setup 41
system-period temporal tables
  - CNTLCARDS BMCLOAD 133
  - CNTLCARDS DB2 132
  - CNTLCARDS DB2DDL 135
  - CNTLCARDS DB2LOAD 131
  - FORMAT BMCLOAD 67

SYSTEMPAGES NO, copies defined with 113, 115

T

- table locks 51, 440, 441
- table names, determining 471
- table space status 49
- table spaces
  - calculating estimated number of rows 125
  - externalizing pages for using
    - CHANGE_QUIESCE option 424
  - multi-table 115
  - partition-by-growth
    - with FILTERPART 109
    - with INFILE ddname 115
    - with ORDER BY 219
    - with ORDER YES 139
  - partitioned
    - examples 314, 321, 338
    - MAXSORTS option 147
    - multiple data sets 409
    - multitasking performance 408
    - separate data sets unload 289
    - sorting 141
  - range-partitioned 115
  - segmented 141, 409
  - status and concurrency 49
  - unloading entire 97, 381
  - versioned 68
- table structure for FORMAT BMCLOAD option 68
- tables
application-period temporal
  CNTLCARDS BMCLOAD 133
  CNTLCARDS DB2 132
  CNTLCARDS DB2DDL 135
  CNTLCARDS DB2LOAD 131
bitemporal
  CNTLCARDS BMCLOAD 133
  CNTLCARDS DB2 132
  CNTLCARDS DB2DDL 135
  CNTLCARDS DB2LOAD 131
FORMAT BMCLOAD 67

clone 97, 211
organized by hash 29
specifying 209, 211
system-period temporal
  CNTLCARDS BMCLOAD 133
  CNTLCARDS DB2 132
  CNTLCARDS DB2DDL 135
  CNTLCARDS DB2LOAD 131
FORMAT BMCLOAD 67
temporal
  CNTLCARDS BMCLOAD 133
  CNTLCARDS DB2 132
  CNTLCARDS DB2DDL 135
  CNTLCARDS DB2LOAD 131
description of support 16
FORMAT BMCLOAD 67

tables, BMC
  backing up 469
  BMCHIST 474
  BMCLGRNX 477
  BMCSYNC 478
  BMCTRANS 486
  BMCUTIL 488
  BMXCOPY 491
call considerations 469
determining names 471
querying 471
warnings 469
tables, BMCDICT 472
TABLESPACE option 97, 381
tagging records
  AUTOTAG option 145
  RECORDID option 211
tape data sets

device identification 441
dynamic allocation options 183, 461
expiration date 462
options, dynamic allocation 203
retention period 204, 462
specifying 447
using data compression 203, 462
tape files, final disposition 441
TAPEDISP option 441
tapes. See tape data sets or tape files
TAPES option 441
TASKMAX option 441
tasks that UNLOAD PLUS performs 24
temporal tables
  CNTLCARDS BMCLOAD 133
  CNTLCARDS DB2 132
  CNTLCARDS DB2DDL 135
  CNTLCARDS DB2LOAD 131
description of support 16
  FORMAT BMCLOAD 67
TERADATA keyword of CNTLCARDS option 138, 528
TERADATA-BULK keyword of CNTLCARDS option 138, 522
TERADATA-FAST keyword of CNTLCARDS option 138, 525
TERADATA-MULT keyword of CNTLCARDS option 139, 526
Teradata, generating control statements for 138, 520
TERM parameter of EXEC statement 274
TERMINATEDBY option of FORMAT CSV option 73, 151
terminating a job 309
terminating BMC utilities 471
termination, recovering from abnormal 171
threads, canceling
  authorizations required 42
  FORCE option 179, 430
  FORCE_AT option 431
  FORCE_RPT option 432
reporting 181, 278, 432
specifying when to cancel 180, 431
threads, number used by UNLOAD PLUS 45
time
CURRENT TIME option 229
exit routines 77
field 243
formats 247
overriding the default DB2 external format 161
rules for constants 217
variables 186, 450
TIME data type 243
time zone
data type 242
implicit 164
support 17
TIMEFMT option 161
timeout
displaying locks 178, 429
number of retry attempts 427
specifying 175, 428
specifying the number of retry attempts 177
specifying the time between retry attempts 178
time between retry attempts 427
timestamp
CURRENT TIMESTAMP option 219, 229
default DB2 external format, overriding 162
field 242
formats 247
with precision 17
with time zone 17, 164, 242
TIMESTAMP data type 242
TIMESTAMP WITH TIME ZONE data type 242
tokens in syntax, rules for 79
Top Secret. See CA Top Secret
trailing blanks, removing
RTRIM option 151
TRIM option 232, 234, 240
truncating
binary values 240
character values 232
DECFLOAT values 165
graphic values 235
TSFMT option 162
TSO cancel command 309
tuning UNLOAD PLUS
I/O 410
memory use 414
performance 399
TRANSTAB field 261
TRIM option
of BINARY data type 240
of CHAR data type 232
of VARBINARY data type 240
of VARCHAR data type 234
TRUNCATE option
of BINARY data type 240
of CHAR data type 232
of GRAPHIC data type 235
of VARBINARY data type 240
of VARCHAR data type 235
of VARGRAPHIC data type 235
Unicode
elements 381
option to support 158
restrictions 78
supported translation 262
translation process 261
UNLOAD TABLESPACE option 97
unloading catalog data 63
UNICODE option 158
union in view 211
unit count default for dynamic allocation 457
UNIT option 183, 449
UNITCNT option 196, 457
universal table spaces 109, 139, 219
UNLDMAX option 442
UNLOAD command 97
UNLOAD PLUS
  authorizations needed 43
UNLOADADDN option
  command 121
  example 342
  installation 442
UNLOADADDN_ACTIVE option 399, 442
UNLOADEXIT option 169
unloading
  catalog data 63
  data for LOADPLUS 67, 389
  LOB data 54
  XML data 54
UNLOADKEYS entries, cleaning up 483
unsupported features of DB2 78
USELRECL option
  command 126
  installation 443
  performance considerations 401
user exit routines
  assembler, sample 541
  authorization 77, 531
  C, sample 555
  calling in supervisor or problem state 407, 444
  COBOL II, sample 582
  data conversion specification 246
  data type conversion 246
  discarded records, cause of 145
  invoking DB2 user exits 77, 407
  LE C, sample 571
  LE COBOL, sample 582
  parameter block DSECT 538
  performance considerations 407
  programming language 169
  requirements 531
  specifying 169
  unload exit specification 169
    with FORMAT BMCLOAD option 68
user-defined data types 29
USERID (user identifier) parameter of EXEC statement 275
UTILID (utility identifier) parameter of EXEC statement 274
UTILINIT keyword of ON FAILURE option

abends
  using TERMINATE UTILITY for 171
ALL keyword
  of ON FAILURE option 171
error handling
  using ON FAILURE to recover from abnormal termination 171
keywords
  ALL keyword
    of ON FAILURE option 171
  TERMINATE UTILITY keyword of ON FAILURE option 171
  UNLOAD keyword of ON FAILURE option 171
  UTILINIT keyword of ON FAILURE option 171
  UTILTERM keyword of ON FAILURE option 171
  multiple ON FAILURE statements 171
utility identifier (UTILID) parameter of EXEC statement 274
utility parameters of EXEC statement 272
utility restrictive state, utility exclusive control (UTUT) 49
UTILTERM keyword of ON FAILURE option 171
UTPRINT data set 35
  specification and usage 307
UTUT (utility restrictive state, utility exclusive control) 49
UXSTATE option 444

V

VALIDPROCs with FORMAT BMCLOAD option 68
VALUE option 228, 259, 321
VARBINARY data type 240
VARCHAR data type 232, 261
VARGRAPHIC data type 235, 261
variable block (VB) record format, specifying 128, 436
variable-block spanned (VBS) file
  for spanned-record output 58, 129, 286
  when specifying RECFM option 128
  when UNLOAD PLUS calculates record size 288
variables
construction of data set names 186, 295, 450
write claimers, canceling 180, 430
differences when using with DIRECT YES or
write error page range (WEPR) status 49
DIRECT NO 188
resolving when using DIRECT NO 188
WTOMSG option 108
user-defined 169
VB keyword of RECFM option 128, 436
XBM (EXTENDED BUFFER MANAGER)
VBS data sets, for spanned-record output 58, 129,
example 329
286
verifying
licensing 53, 106, 445
authorization 42
versioning, index and table space 68
management set 53
versions of DB2 41
purpose and use 53
when different on data sharing group 273
SHRLEVEL CHANGE CONSISTENT YES 104
views, specifying 211
XBM subsystem ID 108, 444
virtual storage 410
zIIP processing 106, 445
above the bar 45
XBMGROUP name 108, 444
and MEMLIMIT 45
XBMID option
cardinal command 108
installation 108
example 329
installation 444
XML data
VOLUMES option 201
binary 230, 233
command 193
VSAM data sets, unloading from 35, 118
installation 454
Versam DD statements 281
relation to VOLUMES option 201
VSAMDD keyword of INFILE option 118
VOLUMES option 201, 460
VSAMDDPREFIX keyword of INFILE option 119
VSAM data sets, unloading from 35, 118
XML data type
VSAMDD keyword of INFILE option 118
data type 245
unloading to referenced files 55
VOLUMES option 201, 460
unloading to unload data sets 54
VSAMDD keyword of INFILE option 118
XML data type 245
XML keyword of FORMAT option
XML output
description 74, 152
considerations and restrictions 74
example of unloaded data when using 76
specifying a string 152
XML data type 245
XML type modifier 78
work data set, specifying with WORKUNIT option 444
XML output
work files
deleting 173, 426
output format 74
dynamically allocating 122
performance considerations 409
worklist environment
PRESERVE WHITESPACE string 131
DELETFILES option 173
unloading to referenced files 55
DELFILES option 426
unloading to unload data sets 54
NULLCHAR option 435
XML data type
NULLTYPE option 435
XML output
viewing utility output 285
XML type modifier 78
WORKUNIT option 444
Z/OS Unicode Conversion Services 261
zeros
with FILL option 157, 230
with FIXEDVARCHAR option 167
with TRIM option 240
with TRUNCATE option 240
ZIIP option 106, 445
zIIP processing
  enabling and disabling 106, 445
  requirements 41
  specifying an XBM subsystem 108, 444
  UNLOAD PLUS requirements 41
zoned decimal data

assigning overpunch values using
  ZONEDDECOVP option 165, 446
data type 238
example 377
ZONEDDECOVP option
command 165
example 377
installation 446