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 2101 CITYWEST BLVD HOUSTON TX 77042-2827 USA</td>
<td>1 713 918 8800 or 1 800 841 2031</td>
<td>1 713 918 8000</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.

DB2, BatchPipes, FlashCopy, IBM, Language Environment, MVS, RACF, System z, System z10, and z/OS are the 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 UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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

You can obtain technical support by using the BMC Support Central website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see “Before contacting BMC.”

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 issues similar to yours and possible solutions
- order or download product documentation
- download products and maintenance
- report an issue or ask a question
- subscribe to receive proactive e-mail alerts when new product notices are released
- 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 issue
- 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, such as file system full
  - messages from related software
License key and password information

If you have questions about your license key or password, use one of the following methods to get assistance:

- Send an e-mail message to customer_support@bmc.com.
Contents

About this book 19

Related publications ................................................. 19
Conventions .......................................................... 20
Syntax diagrams ....................................................... 21
Summary of changes .................................................. 22

Chapter 1  Introducing UNLOAD PLUS 31

Overview .............................................................. 31
  The need to unload data ........................................... 32
  The UNLOAD PLUS solution ..................................... 32
  UNLOAD PLUS benefits ......................................... 32
BMC Software solution integration ............................... 33
  Product components .............................................. 34
  Technology components ......................................... 34
  Features ........................................................... 35
  Other BMC solutions .............................................. 35
Tasks that UNLOAD PLUS performs ............................... 36
Understanding execution modes .................................... 39
  DIRECT YES ....................................................... 39
  DIRECT NO ....................................................... 40
How UNLOAD PLUS works .......................................... 42
  DIRECT YES processing phases ................................. 42
  DIRECT NO processing phases ................................ 44
UNLOAD PLUS data sets ............................................. 46
  Associated products and common components that UNLOAD PLUS
  uses ............................................................... 48

Chapter 2  Operational considerations 49

System setup .......................................................... 50
  DB2 support ....................................................... 50
System requirements ................................................. 50
Software requirements .............................................. 50
  Required authorization .......................................... 51
  MEMLIMIT system parameter ................................. 53
Number of DB2 threads that UNLOAD PLUS uses ............ 54
  DIRECT YES ....................................................... 54
  DIRECT NO ....................................................... 54
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serialization and concurrency</td>
<td>55</td>
</tr>
<tr>
<td>Executing BMC utilities concurrently</td>
<td>55</td>
</tr>
<tr>
<td>Object status with DIRECT YES</td>
<td>57</td>
</tr>
<tr>
<td>Object status with DIRECT NO</td>
<td>59</td>
</tr>
<tr>
<td>Drain and lock table processing</td>
<td>59</td>
</tr>
<tr>
<td>Dynamic allocation</td>
<td>60</td>
</tr>
<tr>
<td>SHRLEVEL CHANGE CONSISTENT YES considerations</td>
<td>61</td>
</tr>
<tr>
<td>Unloading LOB and XML data</td>
<td>62</td>
</tr>
<tr>
<td>Unloading LOB and XML data to unload data sets</td>
<td>62</td>
</tr>
<tr>
<td>Unloading LOB and XML data to referenced files</td>
<td>63</td>
</tr>
<tr>
<td>General considerations when unloading LOB and XML data</td>
<td>65</td>
</tr>
<tr>
<td>Considerations when unloading LOB and XML data to unload data sets</td>
<td>66</td>
</tr>
<tr>
<td>Considerations when unloading LOB or XML data to referenced files</td>
<td>67</td>
</tr>
<tr>
<td>Unloading catalog data</td>
<td>70</td>
</tr>
<tr>
<td>BatchPipes output</td>
<td>71</td>
</tr>
<tr>
<td>Output format</td>
<td>71</td>
</tr>
<tr>
<td>DB2 output with external date and time (FORMAT STANDARD)</td>
<td>72</td>
</tr>
<tr>
<td>External-formatted output (FORMAT EXTERNAL)</td>
<td>73</td>
</tr>
<tr>
<td>DB2 output with internal date and time (FORMAT INTERNAL)</td>
<td>73</td>
</tr>
<tr>
<td>Data migration using LOADPLUS (FORMAT BMLOAD)</td>
<td>73</td>
</tr>
<tr>
<td>DSNŢIAUL-formatted output (FORMAT DNTIAUL)</td>
<td>78</td>
</tr>
<tr>
<td>CSV output (FORMAT CSV)</td>
<td>79</td>
</tr>
<tr>
<td>XML output (FORMAT XML)</td>
<td>81</td>
</tr>
<tr>
<td>Examples of FORMAT option output</td>
<td>83</td>
</tr>
<tr>
<td>DB2 user exits</td>
<td>84</td>
</tr>
<tr>
<td>DB2 features that UNLOAD PLUS does not support</td>
<td>85</td>
</tr>
</tbody>
</table>

**Chapter 3 Syntax of the UNLOAD command**

- Command syntax rules for UNLOAD PLUS                                       87
- Specifying object names in your UNLOAD PLUS syntax                         88
- Alphabetical listing of UNLOAD PLUS options                                 89
- UNLOAD PLUS command syntax diagrams                                         92
- UNLOAD PLUS options                                                         103
  - Basic processing options                                                  103
  - OUTPUT syntax options                                                     190
  - SELECT options                                                           216
  - Data type keywords                                                        240
  - Supported DATE, TIME, and TIMESTAMP formats                               254
  - Supported data type conversions                                           255
  - Additional numeric EXTERNAL or CHARACTER field considerations             263
  - Additional numeric field considerations                                   264
  - Additional date, time, or timestamp field considerations                  265
  - Additional variable field considerations                                  265
  - Additional ROWID field considerations                                     266
  - Data translation                                                          266
Chapter 4  Building and executing UNLOAD PLUS jobs 271

Building the UNLOAD PLUS job ................................................................. 271
  JOB statement .......................................................... 272
  EXEC statement .......................................................... 272
  REGION parameter ...................................................... 272
  Utility parameters on the EXEC statement ......................... 273
  STEPLIB DD statement ................................................... 276
  UNLOAD PLUS DD statements ........................................... 276
Running UNLOAD PLUS jobs ..................................................... 307
  Invoking UNLOAD PLUS .................................................. 307
  Restarting UNLOAD PLUS ................................................ 308
  Terminating or canceling an UNLOAD PLUS job ................. 308

Chapter 5  Examples of UNLOAD PLUS jobs 309

Overview ................................................................. 309
Example 1: Unloading partitions to separate data sets ........... 313
Example 2: Unloading columns in clustering order .......... 320
Example 3: Unloading multiple tables to a single data set .... 325
Example 4: Unloading while allowing read/write access to the table space .. 330
Example 5: Unloading data from a DSN1COPY .............. 334
Example 6: Unloading a full image copy ......................... 338
Example 7: Using FILTERPART ........................................... 342
Example 8: Using multiple SELECT statements ............ 346
Example 9: Unloading from multiple table spaces ............ 353
Example 10: Unloading from incremental image copies .... 357
Example 11: Using a DDLIN data set ............................. 361
Example 12: Unloading to CSV format ............................. 366
Example 13: Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism .. 369
Example 14: Handling an abnormal termination ............ 374
Example 15: Unloading LOB and XML data; using nonstandard null indicators ...................................................... 378
Example 16: Using dynamic allocation ............................. 386
Example 17: Changing overpunch values in zoned decimal data .. 392
Example 18: Unloading a table space and unloading data as Unicode .... 396
Example 19: Unloading data to LOADPLUS ....................... 407

Chapter 6  Tuning UNLOAD PLUS jobs 419

Tuning for performance .................................................. 419
  Setting installation options for optimal performance ........ 419
  Specifying command options for optimal performance .... 421
  Additional performance information for installation and command options ...................................................... 423
  Enabling multitasking for performance ......................... 427
  Additional performance tuning recommendations .......... 429
  Performance tuning for specific scenarios ................. 430
  Interpreting performance-related messages ............... 431
Tuning to improve memory use. ........................................... 434
Tuning sort processing ............................................. 434
Tuning when defining a large number of partitions .......... 434

Appendix A  UNLOAD PLUS installation options  437
Overview ................................................................. 437
Basic UNLOAD PLUS installation options .................... 438
Dynamic allocation installation options ......................... 464
  Default output descriptor options for dynamic allocation . 465
  Options that are common to disk and tape data sets ....... 466
  Options for disk data sets ...................................... 474
  Options for tape data sets .................................... 478
DYNALOC installation option ....................................... 480

Appendix B  Common utility tables  483
Overview ................................................................. 483
  Considerations and warnings for common utility tables. . 484
  Managing common utility tables ................................. 485
BMCDICT table ......................................................... 487
  Considerations for the BMCDICT table ....................... 487
  Maintaining the BMCDICT table ............................... 488
BMCHIST table ......................................................... 488
  Maintaining the BMCHIST table ............................... 490
BMCLGRNX table ....................................................... 490
BMCSYNC table ........................................................ 491
  Considerations for the BMCSYNC table ....................... 493
  Maintaining the BMCSYNC table ............................... 494
BMCTRANS table ....................................................... 495
BMCUTIL table ......................................................... 496
  Maintaining the BMCUTIL table ............................... 498
BMXCOPY table ........................................................ 499
  Maintaining the BMXCOPY table ............................... 503

Appendix C  Generating control statements for DB2 or other software products  505
Overview ................................................................. 505
Inserting additional command options into utility control statements .................................................. 506
Control statements to create and load DB2 tables .......... 507
  DB2LOAD option ................................................... 508
  DB2 option .......................................................... 509
  BMCLOAD option for LOADPLUS for DB2 ................... 512
  DB2DDL option ..................................................... 514
Control statements for FOCUS processing ..................... 515
Control statements for SAS processing ......................... 518
Control statements to create and load SQL/DS tables ....... 520
  SQL/DS-LOAD option ............................................. 520
  SQL/DS option ..................................................... 522
  SQL/DS-DDL option .............................................. 524
Figures

UNLOAD PLUS processing phases when using DIRECT YES .................. 43
UNLOAD PLUS processing phases when using DIRECT NO .................. 45
Example OUTPUT statements and field specification for unloading to
    referenced files ....................................................... 67
UNLOAD PLUS command syntax diagrams .................................... 92
Detail syntax diagrams ..................................................... 97
Sample OUTPUT statement with SUBSETS ................................. 199
Data set names resulting from SUBSETS option ......................... 199
Unloading a partitioned table space (part 1 of 2) ...................... 290
Directing multiple SELECT statements to one or more data sets ...... 292
Example 1: No JCL DD statements: multiple SELECT statements with
different output descriptors ........................................... 296
Example 2a: No JCL DD statements: multiple SELECT statements with
    identical output descriptor .......................................... 296
Example 2b: No JCL DD statements: multiple SELECT statements using the
    &SELNUM variable ...................................................... 297
Example 3: No JCL DD statements: single SELECT statement and using the
    &PART variable ......................................................... 297
Example 4a: Mixture of JCL and dynamic allocation: multiple SELECT
    statements and using the &PART variable ............................ 297
Example 4b: Mixture of JCL and dynamic allocation: multiple SELECT
    statements and using the &PART variable ............................ 298
Example 5: Mixture of JCL and dynamic allocation: multiple SELECT
    statements with the &SELNUM variable ................................ 298
Dynamic allocation examples .................................................. 299
Dynamic allocation example 1 .............................................. 302
Dynamic allocation example 2 .............................................. 303
Dynamic allocation example 3 .............................................. 303
JCL for example 1 .......................................................... 313
SYSPRINT for example 1 .................................................... 317
JCL for example 2 .......................................................... 321
SYSPRINT for example 2 .................................................... 322
JCL for example 3 .......................................................... 325
SYSPRINT for example 3 .................................................... 326
JCL for example 4 .......................................................... 331
SYSPRINT for example 4 .................................................... 331
JCL for example 5 .......................................................... 335
SYSPRINT for example 5 .................................................... 335
JCL for example 6 .......................................................... 339
SYSPRINT for example 6 .................................................... 339
Tables

Execution modes ................................................................. 39
Processing phases of UNLOAD PLUS (DIRECT YES) .................. 42
Processing phases of UNLOAD PLUS (DIRECT NO) ................. 44
UNLOAD PLUS data sets for DIRECT YES ............................... 46
UNLOAD PLUS data sets for DIRECT NO ................................. 47
Common components that UNLOAD PLUS uses ....................... 48
Executing BMC utilities concurrently ................................. 55
Status changes based on the value of QUIESCE ..................... 58
Options that are incompatible when unloading LOB or XML data  65
Incompatible OUTPUT options for PDS and PDSE referenced files 68
FORMAT option keywords .................................................... 71
Options that are incompatible with FORMAT BMCLOAD ............ 76
Table definition for FORMAT example ................................. 83
Examples of unloaded data when using the FORMAT option ...... 83
UNLOAD PLUS command options ....................................... 89
Values for the ROWSETSZ option ....................................... 108
Values for the QUIESCE option ......................................... 111
Alternate sources of UNLOAD PLUS input ............................ 118
Values for the INLINE option ............................................ 124
Values for the ACTIVE option ............................................ 130
ANALYZE HURBA formula parameters .................................. 132
SORTNUM values ............................................................. 149
Values for the NULLCHAR option ....................................... 162
Values for the NULLTYPE option ....................................... 163
ROUND_HALF_DOWN rounding .......................................... 172
ROUND_HALF_EVEN rounding ........................................... 173
ROUND_HALF_UP rounding ............................................... 173
Supported program languages for user exits .......................... 176
Symbolic variables for output data set names .................... 194
Resolution of symbolic variables when using DIRECT NO ....... 196
Values for the TRTCH option ............................................. 214
NAME option values ....................................................... 221
Strings to add to INTO statement for temporal objects .......... 222
Truth table for AND, OR, NOT .......................................... 225
Comparison operators ....................................................... 226
Rules for constants ......................................................... 228
Additional information about data types ............................. 240
User exit return codes ...................................................... 253
Date formats ............................................................... 254
Timestamp formats .......................................................... 254
UNLOAD PLUS command options that correspond to installation options .................................................. 421
Command option values for performance ................................................................. 422
Performance tuning scenarios .............................................................. 431
Customized jobs and installation macros ........................................... 438
Basic UNLOAD PLUS installation options ......................................................... 438
SORTNUM values .................................................................................. 458
UNLOAD PLUS installation options for dynamic allocation .............. 465
Shipped values for DSNNAME ................................................................. 467
Symbolic variables for data set names within SYSIN .................. 468
Valid DSNTYPE values ................................................................. 471
DYNALLOC parameters ................................................................. 480
Common utility tables ................................................................. 484
BMCDICT table ............................................................................. 487
BMCHIST table ............................................................................. 489
BMCLGRNX table ............................................................................. 490
BMCSYNC table ............................................................................. 491
BMCTRANS table ............................................................................. 495
BMCUTIL table ............................................................................. 496
BMCXCOPY table ............................................................................. 499
Sample user exits ................................................................. 539
Function and return codes ................................................................. 540
Fields of the SQLDA portion of the pseudo-SQLDA ................ 541
Fields of the SQLVAR portion of the pseudo-SQLDA ................ 541
Valid SQLTYPE field values for pseudo-SQLDA DSECT ............ 542
Assembler user exit parameter block DSECT fields ................... 545
C user exit parameter fields ................................................................. 563
LE C user exit parameter fields ................................................................. 583
COBOL user exit parameter fields ................................................................. 602
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 (http://www.bmc.com/support), you can

- download a zipped set of documentation PDFs from each product’s EPD page
- link to the BMC Documentation Center (https://webapps.bmc.com/infocenter/index.jsp) to browse documentation sets, or to view video demos (short overviews of selected product concepts, tasks, or features)
- view individual product documents (books and notices) within the “A – Z Supported Product List”
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.

Conventions

This book 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

- Change bars signify changes that clarify or correct existing information, or that provide new information corresponding to product changes. This book does not use change bars to denote editorial or formatting changes, unless these updates significantly affect your use of the information.
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 MVS™ commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum portion of the MVS command or keyword might be displayed in uppercase letters with the remainder of the word in lowercase letters (for example, CANcel).

- The following conventions apply to variables in syntax diagrams:
  - Variables typically are 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).

Summary of changes

This section summarizes changes to the functionality of the product, listing the changes by product version and release date. The summary includes enhancements to the product and any major changes to the documentation.

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. For full requirement information, see “Software requirements” on page 50.

— 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. For more information, see “CLONE” on page 105.

— 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
  - BMLOAD
  - DB2
  - DB2DDL

For specific information about which control cards UNLOAD PLUS generates for temporal tables, see the individual CNTLCARDS values in “CNTLCARDS” on page 137.

— 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. For more information, see “TIMESTAMP WITH TIME ZONE” on page 250.

  — 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.

For more information, see “IMPLICIT_TZ” on page 170.
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. For more information, see page 241 or page 243.

- 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. For more information, see “PART or LOGICAL PART” on page 127.

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. For more information, see “INFILE” on page 118.

- 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. For more information, see “FORCE” on page 187 or the installation options that begin on page 449.

This option requires additional DB2 authorization. For more information, see “Additional authorizations for canceling threads” on page 52.

You can also specify a new DD statement, BMCFORCE DD, to contain the thread cancelation report. For more information, see “BMCFORCE data sets” on page 277.

- 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. For more information, see “SYSPRIN2 data set” on page 285.

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. For more information, see the DSNTYPE installation option or command option on page 470 and page 197.
Summary of changes

- 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. For more information, see “ON FAILURE” on page 178.

- 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. For more information, see “VSAMDDPREFIX prefix” on page 126.

- 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. For full requirement information, see “System setup” on page 50.

  — 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
**Version 10.1.00  April 2011**

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

**DB2 Version 10 support**

UNLOAD PLUS supports the following features of DB2 Version 10:

- For both DIRECT YES and DIRECT NO, UNLOAD PLUS supports the following features:
  
  — catalog and directory changes and restructuring
  — new system and database authorities
  — migration to DB2 Version 10 from either DB2 Version 8 or DB2 Version 9
  — table spaces that contain pending DDL changes
    
    UNLOAD PLUS does not materialize the DDL changes.
  — XML columns that support XML versions
  — LOB and XML table spaces and index spaces that are defined with DEFINE NO
  — objects that contain compression dictionaries that were created during DB2 SQL INSERT processing
    
    UNLOAD PLUS does not unload copies that contain these compression dictionaries if the copies were created with the DSN1COPY utility or with the SYSTEMPAGES NO option.

- When DIRECT YES is in effect, UNLOAD PLUS supports unloading from an IBM FlashCopy® image copy.
  
  Specify INFILE VSAMDD to unload from a FlashCopy.

- When DIRECT NO is in effect, UNLOAD PLUS supports the following features:
  
  — row- and column-level security when performing SELECT processing
  — tables that are defined as ORGANIZE BY HASH
  — universal table spaces that are defined with MEMBER CLUSTER
Summary of changes

— inline LOB data

When unloading inline LOB data, generated control cards do not include DDL that is specific to inline LOB columns.

— temporal tables

Temporal table support has the following limitations:

■ UNLOAD PLUS does not unload history tables.

■ Generated control cards do not include DDL that is specific to temporal tables.

— timestamp columns that are defined with a precision other than 6

The only supported CNTLCARDS options are DB2, DB2LOAD, and DB2DDL.

— timestamp columns that are defined as TIMESTAMP WITH TIME ZONE

Timestamp with time zone support has the following limitations:

■ You cannot include TIMESTAMP WITH TIME ZONE on your field specification.

■ The only supported CNTLCARDS options are DB2, DB2LOAD, and DB2DDL.

Additional enhancements and changes

■ UNLOAD PLUS now provides the option to offload eligible processing to an IBM System z® Integrated Information Processor (zIIP). To enable and use zIIP processing, you must have an installed and authorized version of the EXTENDED BUFFER MANAGER (XBM) product or the SNAPSHOT UPGRADE FEATURE (SUF) technology. For this version, the minimum version of XBM or SUF is 5.6.00 with PTF BPE0313.

To enable DB2 Version 10 support, XBM and SUF also require PTF BPE0311.
The following installation and command options apply to this feature:

— The new ZIIP installation and command options enable this functionality. For more information, see “ZIIP” on page 113 or page 463.

— You can also use the existing XBMID installation or command option to specify an XBM subsystem to use to access this functionality. For more information, see “XBMID” on page 114 or page 462.

For more information about the XBM component that enables the use of zIIPs, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

■ Enhancements to I/O handling routines have changed the way that UNLOAD PLUS uses the values that you specify for the buffer installation options. For information about these changes, see the following options:

— IBUFFS on page 451
— UBUFFS on page 459

■ UNLOAD PLUS now supports the DECFLOAT data type. UNLOAD PLUS has added the following new keywords to support this feature:

— DECFLOAT data type (see page 252)
— DECFLOAT_ROUNDMODE option (see page 172), which enables you to override your subsystem’s default rounding mode for DECFLOAT values

■ When DIRECT NO is in effect, UNLOAD PLUS unloads clone tables and base tables that participate (or have participated) in a clone relationship.

■ You can now specify SELECT statements up to 2 MB long.

■ UNLOAD PLUS now supports extended address volume (EAV) data sets.

■ When both of the following conditions exist, you no longer need to specify an OBID when providing a DDLIN data set for unloading an image copy:

— You specify a single SELECT statement.
— You are unloading a single-table table space.

■ When unloading a partition-by-growth table space and you specify FILTERPART, UNLOAD PLUS now ignores the FILTERPART option instead of terminating.
This version of UNLOAD PLUS has the following changes to minimum requirements. For full requirement information, see “System setup” on page 50.

— z/OS Version 1.10
— DB2 Solution Common Code (SCC) version 10.1.00
— DB2 Utilities Common Code (D2U) version 10.1.00
— for SHRLEVEL CHANGE, XBM or SUF version 5.6.00

**NOTE**
To enable DB2 Version 10 support, XBM and SUF also require PTF BPE0311.

The dynamic allocation information for specific data set types that was in Chapter 2 has been moved to the relevant data set sections in Chapter 4.
Overview

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.

Overview

This chapter presents the following topics:

Overview ................................................................. 31
The need to unload data .............................................. 32
The UNLOAD PLUS solution ....................................... 32
UNLOAD PLUS benefits ............................................ 32
BMC Software solution integration ............................... 33
Product components .................................................. 34
Technology components ............................................ 34
Features .................................................................... 35
Other BMC solutions .................................................. 35
Tasks that UNLOAD PLUS performs ............................. 36
Understanding execution modes .................................. 39
DIRECT YES ............................................................. 39
DIRECT NO ............................................................. 40
How UNLOAD PLUS works ........................................ 42
DIRECT YES processing phases ................................. 42
DIRECT NO processing phases ................................. 44
UNLOAD PLUS data sets .......................................... 46
Associated products and common components that UNLOAD PLUS uses . . . 48
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 BMC Database Administration for DB2 solution. You can use the Database Administration solution to manage your DB2 databases quickly, efficiently, and effectively. Customers who acquire this solution benefit from the features of the following individual products and technologies.
Product 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

Technology components

- BMC Common Statistics
  (required by LOADPLUS to update DB2 catalog and DASD MANAGER PLUS statistics tables)

- BMCSORT
  (required for full sort support for LOADPLUS, UNLOAD PLUS, and RECOVER PLUS)

- Common SQL (ACS)
  (required by CATALOG MANAGER and CHANGE MANAGER)

- Cross-System Image Manager (XIM)
  (required by CHANGE MANAGER to execute portions of a worklist concurrently)

- DB2 Solution Common Code (SCC)
  (a set of common components)

- DB2 Utilities Common Code (D2U)
  (required by LOADPLUS and UNLOAD PLUS)

- High-speed Apply Engine (formerly named Apply Plus)
  (a Log Master for DB2 component that LOADPLUS requires for the SQLAPPLY feature)

- Install Execution Code (AIN)
  (used to create DB2 objects for BMC products and solutions)

- JCL Generation and Execution
  (required by CATALOG MANAGER and CHANGE MANAGER to construct a job control language (JCL) file for running utilities or DB2 commands and execute the commands)
Features

These components provide the following features:

- analyze the effects of changes to database structures
- automate creating, altering, and dropping DB2 objects
- provide easy navigation and management of the DB2 catalog
- aid in developing schema changes for application development and production maintenance
- increase application availability and optimizes resources by reducing the time that it takes to perform changes and to administer multiple DB2 environments
- maintain referential integrity and data integrity
- complete complex structure changes to databases quickly and accurately
- use the fastest, most efficient utilities available to copy, unload, and reload data
- unload and load large object (LOB) and XML data

If you have a password for the solution, you can take advantage of the following additional features of the solution that are available when one component can rely on the presence of other components. The following additional features are available with the solution password:

- execute portions of the worklist concurrently, thus reducing the elapsed time required for executing a worklist that CHANGE MANAGER generates
- unload and load LOB data by using the LOB DATA MOVER program

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
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 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 solution

  - VSAM linear data sets

  - IBM® FlashCopy image copies
Tasks that UNLOAD PLUS performs

- 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
Understands execution modes

UNLOAD PLUS provides two modes of execution. You can use the DIRECT command option to select one of the 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.

Table 1 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 more information about authorization, see “DB2 authorization” on page 51.

For the syntax options that are supported with DIRECT YES, see Table 15 on page 89.
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

**NOTE**

For restrictions and considerations when unloading these types of files, see “INFILE” on page 118.

- 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

**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 Table 15 on page 89.

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
  - inline LOB data

**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

Table 2 describes the processing phases of UNLOAD PLUS and their primary functions when you use DIRECT YES. Figure 1 on page 43 illustrates processing during each phase. All three phases update the BMCUTIL and BMCSYNC tables.

Table 2   Processing phases of UNLOAD PLUS (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 |
Figure 1  UNLOAD PLUS processing phases when using DIRECT YES

All three 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

Table 3 describes the processing phases of UNLOAD PLUS and their primary functions when you use DIRECT NO. Figure 2 on page 45 illustrates processing that occurs during each phase.

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,</td>
</tr>
<tr>
<td></td>
<td>SYSREDnn, 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 (SYSCNTLnn)</td>
</tr>
<tr>
<td></td>
<td>- optionally updates the BMCHIST table in the UNLOAD PLUS database</td>
</tr>
</tbody>
</table>
Figure 2  UNLOAD PLUS processing phases when using DIRECT NO

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

UNLOAD PLUS updates the BMCUTIL table and the BMCHIST table.

Chapter 1  Introducing UNLOAD PLUS  45
Table 4 and Table 5 on page 47 list 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

Table 4 lists the data sets that UNLOAD PLUS uses when DIRECT YES is in effect.

<table>
<thead>
<tr>
<th>Data set or ddname</th>
<th>Description</th>
<th>See page</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>277</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>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>279</td>
</tr>
<tr>
<td>Referenced files</td>
<td>These 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>287</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>281</td>
</tr>
<tr>
<td>SYSCNTLnn</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>284</td>
</tr>
<tr>
<td>SYSIN</td>
<td>SYSIN is the input data set that contains the UNLOAD command.</td>
<td>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>285</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>SYSPRINT is the output data set that contains UNLOAD PLUS messages.</td>
<td>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 UNLOADADDN option described on page 129.</td>
<td>286</td>
</tr>
</tbody>
</table>
Table 4  UNLOAD PLUS data sets for DIRECT YES (part 2 of 2)

<table>
<thead>
<tr>
<th>Data set or ddname</th>
<th>Description</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<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 UNLOADDDN option described on page 129.</td>
<td>286</td>
</tr>
<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 SYSnnnn data sets, where nnnn 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 “UTPRINT data set” on page 307 for more information about this limitation.</td>
<td>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 described on page 126.</td>
<td>280</td>
</tr>
</tbody>
</table>

**Data sets for DIRECT NO**

Table 5 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 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>287</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>284</td>
</tr>
<tr>
<td>SYSIN</td>
<td>SYSIN is the input data set that contains the UNLOAD command.</td>
<td>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, this data set enables you to view UNLOAD PLUS output in real time.</td>
<td>285</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>SYSPRINT is the output data set that contains UNLOAD PLUS messages.</td>
<td>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.</td>
<td>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.</td>
<td>286</td>
</tr>
</tbody>
</table>
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 described in Table 6.

**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>
| EXTENDED BUFFER MANAGER (XBM) or SNAPSHOT UPGRADE FEATURE (SUF) | XBM or SUF provides the following capabilities:  
  - a snapshot image of data in a table space  
  - zIIP processing  
  XBM and SUF are licensed, installed, and maintained separately from UNLOAD PLUS. |
Operational considerations

This chapter presents the following topics:

System setup ................................................................. 50
  DB2 support ............................................................. 50
  System requirements .................................................... 50
  Software requirements .................................................. 50
  Required authorization .................................................. 51
  MEMLIMIT system parameter ............................................ 53
Number of DB2 threads that UNLOAD PLUS uses ......................... 54
  DIRECT YES ............................................................. 54
  DIRECT NO ............................................................. 54
Serialization and concurrency ................................................ 55
  Executing BMC utilities concurrently ................................ 55
  Object status with DIRECT YES ...................................... 57
  Object status with DIRECT NO ....................................... 59
  Drain and lock table processing ..................................... 59
Dynamic allocation .......................................................... 60
SHRLEVEL CHANGE CONSISTENT YES considerations .................. 61
Unloading LOB and XML data ............................................... 62
  Unloading LOB and XML data to unload data sets .................. 62
  Unloading LOB and XML data to referenced files .................. 63
  General considerations when unloading LOB and XML data .......... 65
  Considerations when unloading LOB and XML data to unload data sets 66
  Considerations when unloading LOB or XML data to referenced files 67
Unloading catalog data ...................................................... 70
BatchPipes output .......................................................... 71
Output format ............................................................... 71
  DB2 output with external date and time (FORMAT STANDARD) .......... 72
  External-formatted output (FORMAT EXTERNAL) .................... 73
  DB2 output with internal date and time (FORMAT INTERNAL) ............ 73
  Data migration using LOADPLUS (FORMAT BMCLoad) .................. 73
  DSNTIAUL-formatted output (FORMAT DSNTIAUL) .................... 78
  CSV output (FORMAT CSV) ........................................... 79
  XML output (FORMAT XML) ............................................ 81
Examples of FORMAT option output ....................................... 83
System setup

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

DB2 support

This version of UNLOAD PLUS supports IBM DB2 Versions 9, 10, and 11.

System requirements

This version of UNLOAD PLUS has the following system requirements:

- IBM System z10® processor or a later 64-bit z/Architecture® processor
- IBM z/OS Version 1.10 or later

Software requirements

This version of UNLOAD PLUS has the following requirements for additional BMC software:

- For all types of unload jobs, UNLOAD PLUS requires a minimum of the following versions of BMC common components:
  - version 2.4.01 of BMCSORT
  - version 11.1.00 of the DB2 Utilities Common Code (D2U)
  - version 11.1.00 with PTF BPJ0689 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 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 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

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

If the DB2 DSNX@XAC authorization exit is available for your system, UNLOAD PLUS 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 RACF component.
- CA Technologies provides the DSNX@XAC exit with CA-ACF2 Security for DB2 and CA-Top Secret Security for DB2.

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, UNLOAD PLUS uses the standard DB2 method to check security.

DB2 authorization

To run UNLOAD PLUS, you must have the following authorizations:

- sufficient DB2 authority to execute the UNLOAD PLUS plan and all packages that the UNLOAD PLUS plan uses
Required authorization

- 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.

Additional authorizations for canceling threads

To use the FORCE option to cancel DB2 threads that might prevent a successful drain during an unload job, you must have the following authorizations:

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

These additional authorizations might be implicit in the authority that you have.

Additional authorizations for using XBM or SUF

For zIIP processing and SHRLEVEL CHANGE CONSISTENT YES, UNLOAD PLUS uses features of the EXTENDED BUFFER MANAGER (XBM) product or its SNAPSHOT UPGRADE FEATURE (SUF). For information about security levels and authorizations for XBM, see the BMC Products and Solutions for DB2 Configuration Guide.

Data set authorization

When using DIRECT NO, UNLOAD PLUS uses DB2 to access data sets. However, when using DIRECT YES, UNLOAD PLUS does not use DB2 to access data sets. Therefore, you must have system authorization that is equivalent to the authorization that DB2 requires. The following sections describe how to obtain this authorization depending on the value of the OPNDB2ID installation option.

Establishing authorization when OPNDB2ID=NO

Some sites use 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. In that case, you must have 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 created by COPY PLUS
- key data sets for encrypted copies

**Using a security package other than RACF**

The following procedure illustrates 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 that are described in the preceding section to ADUUMAIN.

**Using RACF and OPNDB2ID=YES**

If you use RACF and OPNDB2ID=YES is set in UNLOAD PLUS, the user who is running UNLOAD PLUS is not required to have the authorizations that “Establishing authorization when OPNDB2ID=NO” on page 52 describes. Because OPNDB2ID=YES tells UNLOAD PLUS to use the DB2 RACF ID instead of the user’s RACF ID, the DB2 RACF ID must have RACF (READ) authorization for these data sets.

**MEMLIMIT system parameter**

UNLOAD PLUS 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. Use any of the following methods if you need to override the default value:

- Specify the MEMLIMIT parameter in the JCL.
- Specify REGION=0M in the JCL.
- Use the SMF IEFUSI exit.
If you are unable to specify REGION=0M, 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.

## 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.
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 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 496.

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.

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 Table 7.

The “Access level” column in Table 7 refers to the value of the SHRLEVEL column in the BMCSYNC table. 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>
<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 (BMCSTATS)</td>
<td>S</td>
<td>none</td>
</tr>
</tbody>
</table>
The setting of the LOCKROW installation option determines whether UNLOAD PLUS uses MVS enqueues or SQL LOCK TABLE statements to serialize the BMCSYNC and BMCUTIL tables. For information about the LOCKROW option, see Appendix A, “UNLOAD PLUS installation options.”
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

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)
- read-or-replication-only (RREPL)
- 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

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 110.
**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 CONSISTENT NO, access to the object is not restricted.

Table 8 describes status changes based on the value of the QUIESCE option when you specify CONSISTENT NO.

<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.</td>
</tr>
</tbody>
</table>

**Warning:** 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.

**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

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 allocation

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.

**SHRLEVEL CHANGE CONSISTENT YES considerations**

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

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.

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 (as described in “Software requirements” on page 50) 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 (see page 112).

5 (optional) To use a specific XBM subsystem, specify that subsystem ID with either the XBMID installation option (page 462) or the XBMID command option (page 114).

6 (optional) Specify the WTOMSG option (page 115) 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.

Unloading LOB and XML data to unload data sets

You can unload LOB or XML data to unload data sets (SYSREC and SYSRED) only when DIRECT NO is in effect.
To unload LOB or XML data to unload data sets

1  Specify DIRECT NO.

For restrictions when using DIRECT NO, see page 107.

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 65
   - “Incompatible options” on page 65
   - “Considerations when unloading LOB and XML data to unload data sets” on page 66

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.

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.
Unloading LOB and XML data to referenced files

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

For more information about using DSNAME patterns, see “OUTPUT statement considerations” on page 67.

- 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 syntax options” on page 190.

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 “Linking referenced files to the output fields” on page 67. For more information about the BLOBF, CLOBF, and DBCLOBF subtypes, see page 241.

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 65
- “Incompatible options” on page 65
- “Considerations when unloading LOB or XML data to referenced files” on page 67

For an example that illustrates how to specify your OUTPUT statements and your field specification, see Figure 3 on page 67.
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.
- You are unloading inline LOB data.
- 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**

Table 9 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>terminates</td>
</tr>
<tr>
<td>FORMAT DSNTIAUL</td>
<td>terminates</td>
</tr>
<tr>
<td>FORMAT EXTERNAL</td>
<td>terminates</td>
</tr>
<tr>
<td>FORMAT XML</td>
<td>terminates</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>

Table 9   Options that are incompatible when unloading LOB or XML data (part 1 of 2)
Considerations when unloading LOB and XML data to unload data sets

Table 9  Options that are incompatible when unloading LOB or XML data
(part 2 of 2)

<table>
<thead>
<tr>
<th>Command or installation option</th>
<th>UNLOAD PLUS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECFM</td>
<td>ignores this option for referenced files</td>
</tr>
<tr>
<td>ROWSETSZ</td>
<td>ignores this option</td>
</tr>
<tr>
<td>UNLOAD TABLESPACE</td>
<td>terminates</td>
</tr>
<tr>
<td>USELRECL</td>
<td>ignores this option for referenced files</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 and XML data to 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.
Considerations when unloading LOB or XML data to referenced files

In addition to the general considerations, the following considerations apply when unloading your 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. For more information about these data types, see the information beginning on page 241.

Figure 3 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')
    DSNTYPE(PDS) UNIT(SYSDA) SPACE(10,10) TRK DIR 50
OUTPUT SYSREF3 DSNAME('RDAJDM.UNLOAD.XML')
    DSNTYPE(LIBRARY) UNIT(SYSDA) SPACE(50,50) CYL DIR 50

SELECT *
INTO
    NAME          CHAR(20),
    EXT           CHAR(3),
    AVI1          CHAR(30) CLOBF SYSREF1,
    AVI2          CHAR(30) CLOBF SYSREF2,
    AVI3          CHAR(30) CLOBF SYSREF3
FROM RDAJDM.AVI

* 
```

**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>
<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>
<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
- DSNAMES
- DSNSTYPE
- 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.

```
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.

Unloading catalog data

DB2 stores catalog data asUnicode 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:

- To avoid contention in your DB2 subsystem, consider specifying the WITH UR clause in your SELECT statements.

- If you are unloading catalog data to ASCII or EBCDIC, see the information about data translation, particularly supported character conversions, in “Data translation” on page 266.
BatchPipes output

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.

Output format

You can use the FORMAT option to specify the format of your output data. Table 11 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>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD</td>
<td>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.</td>
<td>page 72</td>
</tr>
<tr>
<td>EXTERNAL</td>
<td>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.</td>
<td>page 73</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>provides all output data in DB2 internal format</td>
<td>page 73</td>
</tr>
<tr>
<td>BMCLOAD</td>
<td>provides data in an internal format that only LOADPLUS can use. This internal format is a high-speed migration solution for DB2 data.</td>
<td>page 73</td>
</tr>
<tr>
<td>DSNTIAUL</td>
<td>provides output data that matches the output of the IBM DSNTIAUL program.</td>
<td>page 78</td>
</tr>
</tbody>
</table>
The sections 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 83.
- For the FORMAT option, see page 156.
- For performance information, see Table 88 on page 422.

### 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
Data migration using LOADPLUS (FORMAT BMCLOAD)

- 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 the data**

To unload data that you plan to reload by using LOADPLUS, complete the following steps. For an example of this feature, see “Example 19: Unloading data to LOADPLUS” on page 407.

1. Ensure that the table that you are unloading meets the requirements specified in “Table structure” on page 75.
2. Specify FORMAT BMCLOAD (page 157) with your UNLOAD command.
3. Specify CNTLCARDS BMCLOAD (page 140) 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.

For more information, see “Allocating SYSREC and SYSRED data sets” on page 286.

7. Ensure that your UNLOAD command does not contain any of the invalid or restricted commands described in “Incompatible command options” on page 76.

8. Ensure that your job meets additional requirements described in “Additional restrictions and considerations” on page 77.
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

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 section 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.

**Incompatible command options**

Table 12 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>DATERNGMT</td>
<td>ignores this option</td>
</tr>
<tr>
<td>DIRECT NO</td>
<td>terminates</td>
</tr>
<tr>
<td>DIRECT AUTO</td>
<td></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>
Limited options

The following options are limited when you also 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.

Additional restrictions and considerations

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

- You cannot use this option 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.

- 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.
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, however, UNLOAD PLUS (using DIRECT YES) also 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 and, optionally, additional keywords as described on page 158. For an example of this feature, see “Example 12: Unloading to CSV format” on page 366.

**NOTE**

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 (see page 156). For example, if you use TERMINATEDBY, that option must appear before ENCLOSEDBY or NULLSTRING.

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
Considerations

The following considerations apply when you specify FORMAT CSV.

LOB and XML data

If you specify FORMAT CSV when you are unloading LOB or XML data, UNLOAD PLUS terminates.

USELRECL option

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

Data type conversion

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

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

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:

- 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 and, optionally, additional keywords (described on page 159).

**NOTE**

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 (see page 156). 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:

- AUTOTAG
- RECORDID
- NULLTYPE
- CONSTRULES
- USELRECL YES

The following options are not valid when you specify FORMAT XML:

- CNTLCARDS BMCLOAD
  
  UNLOAD PLUS issues a warning message and does not produce control cards.

- INTO
  
  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.

How to import 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.

Table 13 and Table 14 illustrate the results of using each FORMAT option. Table 13 lists different data types and sample data. Table 14 shows the data that would be unloaded, based on the table definitions in Table 13. 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) NOT NULL WITH DEFAULT</td>
<td>&quot;CHARACTERS&quot;</td>
</tr>
<tr>
<td>DEC_COL</td>
<td>DECIMAL(7,2) NOT NULL WITH DEFAULT</td>
<td>142.23</td>
</tr>
<tr>
<td>INTEGER_COL</td>
<td>INTEGER NOT NULL WITH DEFAULT</td>
<td>12996</td>
</tr>
<tr>
<td>DATE_COL</td>
<td>DATE NOT NULL WITH DEFAULT</td>
<td>&quot;08/22/1994&quot;</td>
</tr>
<tr>
<td>TIME_COL</td>
<td>TIME NOT NULL WITH DEFAULT</td>
<td>&quot;08:41 AM&quot;</td>
</tr>
</tbody>
</table>

**Table 14** uses the data from the preceding table for the example formats.

### Table 14  Examples of unloaded data when using the FORMAT option (part 1 of 2)

<table>
<thead>
<tr>
<th>FORMAT option</th>
<th>Form</th>
<th>Example of unloaded data (part 1 of 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD</td>
<td>EBCDIC</td>
<td>CHARACTERS D08/22/199408:41 AM</td>
</tr>
<tr>
<td></td>
<td>HEX</td>
<td>CCCDCCCECDE0123003CFF6FF6FF67FF4CD 3819133592042C0024081221199408A41014</td>
</tr>
</tbody>
</table>
UNLOAD PLUS invokes DB2 user exits only when DIRECT YES is in effect.

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)
DB2 features that UNLOAD PLUS does not support

- authorization exit

By default, UNLOAD PLUS invokes these exits in supervisor state (and PSW key=7). For information about changing the UNLOAD PLUS installation option to achieve better performance, see “UXSTATE installation option” on page 427.

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
DB2 features that UNLOAD PLUS does not support
Chapter 3 Syntax of the UNLOAD command

This chapter presents the following topics:

Command syntax rules for UNLOAD PLUS ........................................ 87
Specifying object names in your UNLOAD PLUS syntax ..................... 88
Alphabetical listing of UNLOAD PLUS options .................................. 89
UNLOAD PLUS command syntax diagrams .................................... 92
UNLOAD PLUS options ................................................................. 103
  Basic processing options .......................................................... 103
  OUTPUT syntax options ............................................................. 190
  SELECT options ................................................................. 216
  Data type keywords ............................................................... 240
  Supported DATE, TIME, and TIMESTAMP formats ......................... 254
  Supported data type conversions ............................................... 255
  Additional numeric EXTERNAL or CHARACTER field considerations ... 263
  Additional numeric field considerations ...................................... 264
  Additional date, time, or timestamp field considerations ................ 265
  Additional variable field considerations ................................... 265
  Additional ROWID field considerations ................................... 266
  Data translation .................................................................. 266

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.
Specifying object names in your UNLOAD PLUS syntax

- 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. For more information about reading syntax diagrams, see “Syntax diagrams” on page 21.

**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

Table 15 lists the UNLOAD PLUS command options alphabetically and indicates where to find each option description. The final two columns indicate whether the option is supported with DIRECT NO or DIRECT YES.

Table 15  UNLOAD PLUS command options (part 1 of 3)

<table>
<thead>
<tr>
<th>Command option</th>
<th>See page</th>
<th>DIRECT NO</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>130</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ANALYZE</td>
<td>131</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>ASCII</td>
<td>165</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>AUTOTAG</td>
<td>152</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CCSID</td>
<td>166</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CENTURY</td>
<td>183</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CLONE</td>
<td>105</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>CNTLCARDS</td>
<td>137</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CNTLDDN</td>
<td>145</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CURRENTDEGREE</td>
<td>109</td>
<td>yes</td>
<td>ignored</td>
</tr>
<tr>
<td>DATACLAS</td>
<td>203</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DATEFMT</td>
<td>167</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DDLDDN</td>
<td>128</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>DECFLOAT_ROUNDMODE</td>
<td>172</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DELETEFILES</td>
<td>181</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DIR</td>
<td>199</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DIRECT</td>
<td>106</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DISCARDS</td>
<td>152</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DISKEXPD</td>
<td>213</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DISKRETN</td>
<td>213</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DRAIN_WAIT</td>
<td>184</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>DSNNAME</td>
<td>193</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>197</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DSPLOCKS</td>
<td>186</td>
<td>ignored</td>
<td>yes</td>
</tr>
</tbody>
</table>
## Table 15 UNLOAD PLUS command options (part 2 of 3)

<table>
<thead>
<tr>
<th>Command option</th>
<th>See page</th>
<th>DIRECT NO</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC</td>
<td>165</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ENUMROWS</td>
<td>148</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>ESTROWS</td>
<td>232</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>EXPDT</td>
<td>215</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FILESZPCT</td>
<td>211</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FILL</td>
<td>164</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FILTERPART</td>
<td>116</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>FIXEDVARCHAR</td>
<td>174</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>FORCE</td>
<td>187</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>FORCE_AT</td>
<td>187</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>FORCE_DELAY</td>
<td>188</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>FORCE_RPT</td>
<td>188</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>FORMAT</td>
<td>156</td>
<td>yes, except FORMAT BMCLOAD</td>
<td>yes</td>
</tr>
<tr>
<td>GDGLIMIT</td>
<td>202</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>IMPLICIT_TZ</td>
<td>170</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>INFILE</td>
<td>118</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>151</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>INTO</td>
<td>220</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>LIMIT</td>
<td>151</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>LOGICAL PART</td>
<td>127</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>LRECL</td>
<td>201</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MAXBLKSIZE</td>
<td>135</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MAXCONNECT</td>
<td>155</td>
<td>yes</td>
<td>ignored</td>
</tr>
<tr>
<td>MAXPRIM</td>
<td>208</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MAXSECD</td>
<td>209</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MAXSORTS</td>
<td>154</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>204</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NAME</td>
<td>220</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NBRSECD</td>
<td>210</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NOSUBS</td>
<td>166</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>NULLCHAR</td>
<td>162</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NULLTYPE</td>
<td>163</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>OBID</td>
<td>224</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>ON FAILURE</td>
<td>178</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ON MESSAGE</td>
<td>177</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>233</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>ORDER</td>
<td>146</td>
<td>ignored</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Table 15  UNLOAD PLUS command options (part 3 of 3)

<table>
<thead>
<tr>
<th>Command option</th>
<th>See page</th>
<th>DIRECT NO</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER BY</td>
<td>231</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>190</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>PART</td>
<td>127</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>PCTPRIM</td>
<td>207</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RECFM</td>
<td>136</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RECORDID</td>
<td>222</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RETPD</td>
<td>215</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RETRY</td>
<td>185</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>RETRY_DELAY</td>
<td>185</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>SELECT</td>
<td>217</td>
<td>yes, with documented exceptions</td>
<td>yes</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>110</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>SORTDEVT</td>
<td>149</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>SORTNUM</td>
<td>149</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>SPACE</td>
<td>207</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>STORCLAS</td>
<td>203</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SUBSETS</td>
<td>199</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>SYNC</td>
<td>155</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>104</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>TIMEFMT</td>
<td>167</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TRTCH</td>
<td>214</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TSFMT</td>
<td>168</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNICODE</td>
<td>165</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNIT</td>
<td>192</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNITCNT</td>
<td>205</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>104</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNLOADADDN</td>
<td>129</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UNLOADEXIT</td>
<td>176</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>USELRECL</td>
<td>134</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VOLUMES</td>
<td>201</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>VOLUMES</td>
<td>201</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>WTRMSG</td>
<td>115</td>
<td>ignored</td>
<td>yes</td>
</tr>
<tr>
<td>XBMID</td>
<td>14</td>
<td>yes for zIIP processing ignored for snapshot processing</td>
<td>yes</td>
</tr>
<tr>
<td>ZIIP</td>
<td>113</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ZONEDDECOPV</td>
<td>171</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
UNLOAD PLUS command syntax diagrams

Figure 4 illustrates the syntax of the UNLOAD command. For more information about how to read syntax diagrams, see “Syntax diagrams” on page 21.

Figure 4  UNLOAD PLUS command syntax diagrams (part 1 of 5)

---

\[\text{UNLOAD PLUS command syntax diagrams (part 1 of 5)}\]

---

\[^a\text{Option has a corresponding installation option}\]
Figure 4  UNLOAD PLUS command syntax diagram (part 2 of 5)

aOption has a corresponding installation option
Figure 4 UNLOAD PLUS command syntax diagram (part 3 of 5)

- **SORTDEV** \( deviceType \)
- **SORTNUM** \( integer \)
- **LIMIT** \( integer \)
- **INTERVAL** \( integer \)
- **DISCARDS** \( integer \)
- **AUTOTAG** \( NO \) or \( YES \)
- **MAXSORTS** \( integer \)
- **MAXCONNECT** \( integer \)
- **SYNC** \( integer \)
- **FORMAT**
  - **STANDARD**
  - **EXTERNAL**
  - **INTERNAL**
  - **BMLOAD**
  - **DSNTIAIL**
  - **CSV**
  - **XML**
- **NULLCHAR** \( ? \)
- **HIVAL**
- **'c'**
- \( 'x'xx' \)
- **NULLTYPE**
  - **T1**
  - **T2**
  - **L1**
  - **L2**
- **FILL** \( NO \) or \( YES \)
- **CCSID** \( sbcs,mixed,dbs \)
- **NOSUBS**

\(^a\)Option has a corresponding installation option

See CSV and XML format block details on page 99.
Figure 4  UNLOAD PLUS command syntax diagram (part 4 of 5)

UNLOAD PLUS command syntax diagrams

Option has a corresponding installation option
Figure 4  UNLOAD PLUS command syntax diagram (part 5 of 5)

Figure 5 on page 97 shows syntax diagram details.

Option has a corresponding installation option
Figure 5  Detail syntax diagrams (part 1 of 7)

**OUTPUT block - dynamic allocation options**

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

**Common options**

- UNIT\(^a\)
- SYSALDDA
- SNO_UNITS
- name
- DSNAME\(^a\)
  - \&USERID.\&JOBNAME.\&TS.\&TYPE.\&SELNUM'
  - pattern
  - dataSetName

- DSNTYPE
  - PDS
  - LIBRARY
  - HFS
  - NONE
  - LARGE
  - BASIC
  - EXTREQ
  - EXTREFF

- DIR integer
- SUBSETS 1

- LRECL integer
- VOLCNT\(^a\)
  - 25
- GDGLIMIT\(^a\)
  - 5
- STORCLAS\(^a\)
  - class
  - NONE

- DATACLAS\(^a\)
  - class
  - NONE

- MGMTCLAS\(^a\)
  - class
  - NONE

- UNITCNT\(^a\)
  - 0

\(^a\)Option has a corresponding installation option
Figure 5  Detail syntax diagrams (part 2 of 7)

**OUTPUT block - dynamic allocation options (continued)**

**Disk options**

- SPACE *(primary,secondary)*
- CYL
- TRK
- PCTPRIM
- AUTO integer

- MAXPRIMA 0 integer
- MAXSECD 0 integer
- NBRSECD AUTO integer

- FILESZPCT 100 integer
- VOLUMESA volume
- DISKRETA integer

- DISKEXPDA date

**Tape options**

- TRTHA
- COMP
- NOCOMP

- RETPD integer
- EXPDA 99000 date

*Option has a corresponding installation option
Figure 5  Detail syntax diagrams (part 3 of 7)

2 control card formats block detail  page 137

CNTLCARDS

DB2LOAD

DB2

BMCLOAD

DB2DDL

EASYTRIEVE

FOCUS

NOMAD

SAS

SQL/DS

SQL/DS-DDL

SQL/DS-LOAD

TERADATA

TERADATA-BULK

TERADATA-FAST

TERADATA-MULT

3 CSV and XML format block detail  page 156

TERMINATEDBYa

SELECT_ELEMENTb

ENCLOSEDBY

AND

NULLSTRING

RTRIMb

a Option is valid for CSV format only

b Option is valid for XML format only
Figure 5  Detail syntax diagrams (part 4 of 7)

4. SELECT block detail

```
SELECT *
  columnName  constant
  CURRENT DATE
  CURRENT TIME
  CURRENT TIMESTAMP
  CURRENT RID

INTO record options block field specification block

FROM 
  OBID (creatorName)
    tableName
    viewName
  X'byteString'

WHERE condition

ORDER BY columnName ASC DESC

OPTIONS (PART partitionNumber : partitionNumber)

NAME name1 RECORDID 'string'

. name2

```
Figure 5  Detail syntax diagrams (part 5 of 7)

6 field specification block detail

- **fieldName**
  - **CHAR**
    - **VARCHAR**
      - **MIXED**
        - **FILL**
          - **NO**
            - **YES**
        - **(length)**
          - **TRIM**
            - **TRUNCATE**
      - **BLOBF**
        - **outputDescriptorName**
          - **BINARYXML**
      - **CLOBF**
        - **outputDescriptorName**
      - **DBCLOBF**
  - **GRAPHIC**
    - **EXTERNAL**
      - **(length)**
      - **TRUNCATE**
  - **VARGRAPHIC**
    - **(length)**
  - **SMALLINT**
    - **(length, scale)**
  - **INTEGER**
    - **EXTERNAL**
      - **(length, scale)**
      - **ROUND**
      - **FILL**
        - **NO**
          - **YES**
  - **BIGINTEGER**
    - **(length, scale)**
  - **decimal specification block**
    - See decimal specification block detail on page 102.
  - **FLOAT**
    - **EXTERNAL**
      - **(length, scale)**
      - **ROUND**
  - **BINARY**
    - **EXTERNAL**
      - **(length, scale)**
  - **VARBINARY**
    - **(length)**
      - **TRUNCATE**
      - **TRIM**
  - **BINARY VARYING**
  - **DATE**
    - **EXTERNAL**
      - **(length)**
      - **CENTURY(ccyy,ccyy)**
  - **TIME**
    - **EXTERNAL**
      - **(length)**
      - **CENTURY(ccyy,ccyy)**
  - **TIMESTAMP**
    - **EXTERNAL**
      - **(length)**
      - **CENTURY(ccyy,ccyy)**
      - **WITH TIME ZONE**
  - **BLOB**
  - **CLOB**
  - **DBCLOB**
  - **DECFLOAT**
    - **(precision)**
  - **XML**
    - **EXTERNAL**
      - **(length)**
  - **EXIT**
    - **programName**
      - **PARM**
        - **constant**
          - **(length)**

a Option has a corresponding installation option
b Not valid for BLOBF, CLOBF, and DBCLOBF subtype
Figure 5  Detail syntax diagrams (part 6 of 7)

6 field specification block detail (continued)

See predicate block detail on page 103

7 decimal specification block detail

8 condition block detail  page 225

See predicate block detail on page 103
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

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

You can use the UNLOAD TABLESPACE command to unload all data from the designated table space. 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 (page 272) or in your installation options (page 453), 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 396.

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.

**NOTE**
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 152.

**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 unloading the following types of data:
  - catalog data
    
    For more information, see “Unloading catalog data” on page 70.
  - LOB or XML data when unloading to data sets (SYSREC and SYSRED)

NOTE
You can specify DIRECT YES or DIRECT NO to unload LOB or XML data to files that are referenced in your unload data set. For more information, see “Unloading LOB and XML data” on page 62.

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, as described in “DIRECT NO” on page 40.
For more information about DIRECT YES mode, see “DIRECT YES” on page 39.

**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 40.

**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 on page 89.

**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 `userID.$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 Table 16.

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>

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 (page 455). 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.

NOTE
To ensure consistent behavior of your UNLOAD PLUS jobs, select 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 on page 106.

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

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

You can use the CURRENTDEGREE keyword to specify that you want DB2 to attempt to use parallelism. 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 (page 444). 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**

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

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 (page 456). 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 (page 442). 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. Table 17 describes the values that you can specify with QUIESCE.

### Table 17  Values for the QUIESCE option

<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>
</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. |

Chapter 3  Syntax of the UNLOAD command  111
Basic processing options

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 (page 443). 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 “SHRLEVEL CHANGE CONSISTENT YES considerations” on page 61. 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 (page 462) or command option (page 114).
- 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. For details, see page 115.

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 (page 442). 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.
The ZIIP option tells UNLOAD PLUS whether to attempt to use IBM® System 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 (page 462). 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. For more information, see “XBMID” on page 114 or page 462.

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.

**XBomid**

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 “SHRLEVEL CHANGE CONSISTENT YES considerations” on page 61.

**Specifying the default**

You can specify the default for the XBMID command option in your installation options module by using the XBMID installation option (page 462). The XBMID command option overrides the default that is in the installation options module.
WTOMSG

This option is applicable only with SHRLEVEL CHANGE CONSISTENT YES.

Specify WTOMSG SUFSTART to write message BMC50008I to the MVS system log to indicate that snapshot-processing initialization for UNLOAD PLUS has successfully completed. You can use the text of this message to trigger the submission of jobs that you want to run concurrently with the UNLOAD PLUS 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. UNLOAD PLUS truncates strings that are greater than 50 characters. Quotes cannot appear within the text string.
FILTERPART

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

If you use the first column of the partitioning key in your WHERE clause, and that column creates the breaks of the partitions in your WHERE clause, this option allows UNLOAD PLUS to filter out partitions that do not meet the WHERE clause criteria.

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</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>199</td>
<td>xxx</td>
<td>005</td>
</tr>
<tr>
<td>2</td>
<td>299</td>
<td>xxx</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>399</td>
<td>xxx</td>
<td>020</td>
</tr>
<tr>
<td>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

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

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.

Table 18 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 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 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.
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. For more information about the BMCXCOPY table, see “BMCXCOPY table” on page 499.

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.

---

**NOTE**

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 the description of the input copy data set under “UNLOAD PLUS DD statements” on page 276.

**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.
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 (see page 177)

— the OBIDs of the tables that you specify, unless you specify the OBID for the table by using the OBID option (see page 224).

— 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 set” 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
Basic processing options

- 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. Table 19 describes the values that you can specify for this option.

Table 19 Values for the INLINE 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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

Warning: UNLOAD PLUS supports unloading data from an inline image copy when you specify INFIL 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.
**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). For more information about the BMCXCOPY table, see “BMCXCOPY table” on page 499.

---

**– 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 the description of the VSAM input data set under “UNLOAD PLUS DD statements” on page 276.

---

**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 the description of the VSAM input data set under “UNLOAD PLUS DD statements” on page 276.

**Restriction**
You cannot specify multiple VSAM data sets for nonpartitioned objects.
PART or LOGICAL PART

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

The PART option specifies the partition number of a partitioned table space to unload. 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 defining a large number of partitions” on page 434.

**DDLDDN**

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

The DDLDDN option allows you to override the default ddname of the DDL input data set. 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 (see page 118). 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. For more information about the DDLIN data set, see Chapter 4, “Building and executing UNLOAD PLUS jobs.”
UNLOADDN

The UNLOADDN option provides UNLOAD PLUS with one of the following pieces of information:

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

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 Chapter 4, “Building and executing UNLOAD PLUS jobs.”

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 190.

NOTE

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 (page 460). 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. Table 20 describes the values that you can specify for this option.

Table 20 Values for the ACTIVE 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</td>
</tr>
<tr>
<td></td>
<td>To activate dynamic allocation for both data sets, specify (YES,YES).</td>
</tr>
<tr>
<td>NO</td>
<td>does not activate dynamic allocation for the corresponding unload data set</td>
</tr>
<tr>
<td></td>
<td>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 (page 460). 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.
Basic processing options

Chapter 3 Syntax of the UNLOAD command

ANALYZE

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

The ANALYZE option tells UNLOAD PLUS to estimate the number of rows for a specific table or partition. UNLOAD PLUS uses the results of the ANALYZE option to calculate the sizes of unload data sets during dynamic allocation. See page 301, which includes several examples.

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.

Specifying the default
You can specify the default for the ANALYZE command option in your installation options module by using the ANALYZE installation option (page 441). 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 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 catalog, UNLOAD PLUS defaults to HURBA. To determine the age of the statistics, use the following formula:

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

**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.

**NOTE**

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, HURBA might report fewer rows than actual for a table space. BMC therefore recommends that you use DB2STATS for the most accurate row estimations.

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**

\[
\text{estimated rows per table} = \frac{\text{HURBA}}{\text{PGSZ}} - 2 \times \text{RPP}
\]

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

**Table 21  ANALYZE HURBA formula parameters (part 1 of 2)**

<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: ( \frac{\text{PGSZ}}{\text{AVGROWSZ}} ) limited by MAXROWS</td>
</tr>
<tr>
<td></td>
<td>- compressed objects: ( \frac{\text{PGSZ}}{\text{AVGROWSZ} \times (\text{CMRATIO}/100)} ) limited by MAXROWS</td>
</tr>
<tr>
<td>AVGROWSZ</td>
<td>average row length</td>
</tr>
</tbody>
</table>
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 (page 444). 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.

Specifying the default

You can specify the default for the USELRECL command option in your installation options module by using the USELRECL installation option (page 461). 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.

NO

Specify USELRECL NO to have UNLOAD PLUS use the record length that it calculates instead of one of the specified LRECL values (as described in “YES”).

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 (page 455). 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.
CNTLCARDS

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 Appendix C, “Generating control statements for DB2 or other software products.”

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.

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 page 137.

- 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 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 506.

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 information about this string, see “string” on page 221.

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 FIELDPROCs.
Additional considerations
The following considerations are specific to CNTLCARDS DB2. For general considerations for the CNTLCARDS option, see page 137.

- 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 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.
Basic processing options

‘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 506.

**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 information about this string, see “string” on page 221.

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 page 137.

**Restrictions**
The following restrictions apply to CNTLCARDS BMCLOAD:

- If you specify CNTLCARDS BMCLOAD with FORMAT XML, UNLOAD PLUS issues a warning message and does not produce control statements.

- UNLOAD PLUS terminates if you specify CNTLCARDS BMCLOAD when unloading LOB or XML columns.

**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.

- If you specify CNTLCARDS BMCLOAD and the following conditions exist, also specify AUTOTAG YES (page 152) 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. (See the restrictions described on page 153.)

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 506.

NOTE
You can also insert additional load syntax to follow the INTO option for the BMC LOADPLUS generated utility control statements. For information about this string, see “string” on page 221.

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 page 137.
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 page 137.

*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 page 137.

*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 page 137.

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 page 137.

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 page 137.

- 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. For more information, see “NULLCHAR” on page 162 and “NULLTYPE” on page 163.

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.
Basic processing options

Restrictions
The following restrictions are specific to CNTLCARDS SQL/DS-DDL. For general considerations for the CNTLCARDS option, see page 137.

- 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. For more information, see “NULLCHAR” on page 162 and “NULLTYPE” on page 163.

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 page 137.

- 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. For more information, see “NULLCHAR” on page 162 and “NULLTYPE” on page 163.

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 page 137.

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 page 137.

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 page 137.

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 page 137.

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” 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 (page 231) 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:

- 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

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

Specify this option to estimate the total number of rows that BMCSORT processes during the UNLOAD phase. 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.
SORTDEV

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

The SORTDEV option specifies the device type for the sort work files that BMCSORT will allocate dynamically. This option overrides the value specified in the BMCSORT DYNALOC installation option.

**NOTE**

If the value of the third parameter in the BMCSORT DYNALOC installation option is OFF, specifying the SORTDEV option turns BMCSORT dynamic allocation on.

**Restriction**

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

SORTNUM

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

The SORTNUM option affects the allocation of sort work files when BMCSORT is allocating your sort work files dynamically. You can specify any integer value from 0 through 255.

Table 22 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 (part 1 of 2)**

<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 480.</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” on page 281.

**NOTE**
If you depend on BMCSORT to dynamically allocate your sort work data sets, BMC recommends that you also specify the ENUMROWS option. For more information, see “ENUMROWS” on page 148.

**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 (page 458). UNLOAD PLUS was shipped with a value of 32. The command option overrides the default that is in the installation options module.

### Table 22  SORTNUM values (part 2 of 2)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Additional considerations</th>
</tr>
</thead>
<tbody>
<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>
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. 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.

For additional information about the effect that the LIMIT keyword has on dynamic allocation, see page 301.

INTERVAL

The INTERVAL option allows you to unload a sample of data, rather than all rows that meet the selection criteria. 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.

**NOTE**

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 269.

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. (See the restrictions described on page 153.)
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:
  - FORMAT XML
  - RECORDID
MAXSORTS

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

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 “Controlling the number of sort processes (SMAX)” on page 426.

**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 (page 456). 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

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

MAXCONNECT specifies the maximum number of tasks that UNLOAD PLUS can execute concurrently when unloading data using DB2 dynamic SQL.

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 (page 443). 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

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. 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.

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. See page 236 for more information about field specification options.

For more information about using the different output formats and the restrictions associated with each option, see “Output format” on page 71.
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 73.

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 73.
Basic processing options

**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 78.

**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 79.

---

**NOTE**

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 (see page 156). 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 81.

**NOTE**
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 (see page 156). 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 DIRECT YES, DIRECT NO, and INTO NAME.

<table>
<thead>
<tr>
<th>Using DIRECT YES</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>SELECT_ELEMENT AUTO</td>
<td></td>
</tr>
<tr>
<td>SELECT NAME,ADDRESS,CITY,STATE FROM MY.CONTACTS;</td>
<td></td>
</tr>
<tr>
<td>Unloaded data</td>
<td></td>
</tr>
<tr>
<td>&lt;CONTACTS&gt;&quot;BMC Software, Inc.&quot;&lt;NAME&gt;&quot;10431 MORADO CIRCLE&quot;&lt;ADDRESS&gt;&quot;AUSTIN&quot;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;&lt;/CONTACTS&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using DIRECT NO</th>
<th>DIRECT NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>SELECT_ELEMENT AUTO</td>
<td></td>
</tr>
<tr>
<td>SELECT NAME,ADDRESS,CITY,STATE FROM MY.CONTACTS;</td>
<td></td>
</tr>
<tr>
<td>Unloaded data</td>
<td></td>
</tr>
<tr>
<td>&lt;$TABLE1&gt;&quot;BMC Software, Inc.&quot;&lt;NAME&gt;&quot;10431 MORADO CIRCLE&quot;&lt;ADDRESS&gt;&quot;AUSTIN&quot;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;&lt;/$TABLE1&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using INTO NAME</th>
<th>DIRECT NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>SELECT_ELEMENT AUTO</td>
<td></td>
</tr>
<tr>
<td>SELECT NAME,ADDRESS,CITY,STATE INTO NAME &quot;CUSTOMER&quot; FROM MY.CONTACTS;</td>
<td></td>
</tr>
<tr>
<td>Unloaded data</td>
<td></td>
</tr>
<tr>
<td>&lt;CUSTOMER&gt;&quot;BMC Software, Inc.&quot;&lt;NAME&gt;&quot;10431 MORADO CIRCLE&quot;&lt;ADDRESS&gt;&quot;AUSTIN&quot;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;&lt;/CUSTOMER&gt;</td>
<td></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.

<table>
<thead>
<tr>
<th>Using DIRECT YES</th>
<th>DIRECT YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT XML</td>
<td></td>
</tr>
<tr>
<td>SELECT_ELEMENT NONE</td>
<td></td>
</tr>
<tr>
<td>SELECT NAME,ADDRESS,CITY,STATE FROM MY.CONTACTS;</td>
<td></td>
</tr>
<tr>
<td>Unloaded data</td>
<td></td>
</tr>
<tr>
<td>&lt;NAME&gt;&quot;BMC Software, Inc.&quot;&lt;NAME&gt;&quot;10431 MORADO CIRCLE&quot;&lt;ADDRESS&gt;&quot;AUSTIN&quot;&lt;STATE&gt;&quot;TX&quot;&lt;/STATE&gt;</td>
<td></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.

Using DIRECT YES
DIRECT YES
FORMAT XML
SELECT_ELEMENT 'TRANSACTION'
SELECT NAME,ADDRESS,CITY,STATE FROM MY.CONTACTS;

Unloaded data
<TRANSACTION><NAME>'BMC Software, Inc.'</NAME><ADDRESS>'10431 MORADO CIRCLE'</ADDRESS><CITY>'AUSTIN'</CITY><STATE>'TX'</STATE></TRANSACTION>

---

**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.
Basic processing options

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. Table 23 describes the values that you can specify for this option.

Table 23  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 (page 453). 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. Table 24 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>This value specifies that the null indicator is 1 byte long, following the column.</td>
</tr>
<tr>
<td>T2</td>
<td>This value specifies that the null indicator is 2 bytes long, following the column.</td>
</tr>
<tr>
<td>L1</td>
<td>This value specifies that the null indicator is 1 byte long, preceding the column.</td>
</tr>
<tr>
<td>L2</td>
<td>This value 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.

- 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 (page 454). 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 supports a format modifier that allows UNLOAD PLUS to fill numeric external fields with zeros when converting numeric columns to their external representations. This capability supports applications such as COBOL that require leading zeros to edit and check unloaded data properly. You can use this global option or the FILL option on your field specification to override the FILL installation option default. See the various discussions under “Data type keywords” on page 240.

Restriction
This option does not apply to conversions that involve the DECFLOAT data type.

Specifying the default
You can specify the default for the FILL command option in your installation options module by using the FILL installation option (page 448). 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.

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, which is the default. For information about data translation, see “Data translation” on page 266.

**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” on page 266.

**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” on page 266.

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 396.

**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” on page 266.

**NOTE**

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**

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

Specify NOSUBS if UNLOAD PLUS should refuse to accept substitution characters during translation between CCSIDs. 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. It 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. It 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.

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 the 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.

- 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.
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.

**NOTE**
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 392.

**Specifying the default**
You can specify the default for the ZONEDECOVP command option in your installation options module by using the ZONEDECOVP installation option (page 464). 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, UNLOAD PLUS uses the DECFLOAT round mode value from your DSNHDECP settings.

ROUND_CEILING

ROUND_CEILING tells UNLOAD PLUS to round toward positive infinity. If the extra digits are all zero or the number is negative, UNLOAD PLUS truncates the digits. Otherwise, UNLOAD PLUS rounds the remaining value up (increments the rightmost digit by 1).

ROUND_DOWN

ROUND_DOWN tells UNLOAD PLUS to truncate the extra digits (round toward 0).

ROUND_FLOOR

ROUND_FLOOR tells UNLOAD PLUS to round toward negative infinity. If the extra digits are all zero or the number is positive, UNLOAD PLUS truncates the digits. Otherwise, the remaining value is negative and UNLOAD PLUS rounds up (increments the rightmost digit by 1).

ROUND_HALF_DOWN

ROUND_HALF_DOWN tells UNLOAD PLUS to round to the nearest number as explained in Table 25.

Table 25  ROUND_HALF_DOWN rounding

<table>
<thead>
<tr>
<th>Extra digits</th>
<th>UNLOAD PLUS 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 tells UNLOAD PLUS to round to the nearest number as explained in Table 26.

Table 26  ROUND_HALF_EVEN rounding

<table>
<thead>
<tr>
<th>Extra digits</th>
<th>Rightmost digit of remaining value</th>
<th>UNLOAD PLUS 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 tells UNLOAD PLUS to round to the nearest number as explained in Table 27.

Table 27  ROUND_HALF_UP rounding

<table>
<thead>
<tr>
<th>Extra digits</th>
<th>UNLOAD PLUS 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 tells UNLOAD PLUS to round away from 0. If the extra digits are all zero, UNLOAD PLUS truncates the digits. Otherwise, UNLOAD PLUS rounds the remaining value up (increments the rightmost digit by 1).
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 DSNTIACUL, 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 any referenced files that it allocates.
UNLOADEXIT

The UNLOADEXIT option allows you to specify the name and programming language of a user-written exit routine.

**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 Appendix D, “UNLOAD PLUS user exits.”

**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 28.

**Table 28  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(^a)</td>
<td>specifies a COBOL II language program</td>
</tr>
<tr>
<td>LE_COBOL(^a)</td>
<td>specifies an LE COBOL language program</td>
</tr>
<tr>
<td>C(^a)</td>
<td>specifies a C language program</td>
</tr>
<tr>
<td>LE_C(^a)</td>
<td>specifies an LE C language program</td>
</tr>
</tbody>
</table>

\(^a\) UNLOAD PLUS sets MAXSORTS to 1 when you specify LE COBOL, COBOL II, C, or LE C user exit routines.
ON MESSAGE

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

ON MESSAGE allows you to specify whether UNLOAD PLUS should stop or continue processing when it encounters message BMC50253 or BMC50254. If you do not specify ON MESSAGE, the utility stops when it encounters either of these messages.

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.

NOTE

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. For more information about this now-obsolete syntax, see “STOP UTILITY” on page 180.

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, SYSCNTL, 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.
Basic processing options

- If you specify an integer that is greater than 31, UNLOAD PLUS issues a user abend that is equal to the integer specified.

**NOTE**
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, SYSCNTL, 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 374.

Specifying the default
You can specify the default for the DELETEFILES command option in your installation options module by using the DELFILES installation option (page 445). 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, SYSCNTL, 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.

NOTE
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.
Basic processing options

NOTE
To specify the final disposition of tape files when you specify DELETEFILES YES, you can specify the TAPEDISP installation option. See Appendix A, “UNLOAD PLUS installation options,” for details.

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. 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. See “DATE-format EXTERNAL(length)” on page 249 or “TIMESTAMP-format EXTERNAL(length)” on page 250 for more information.

Specifying the default
You can specify the default for the CENTURY command option in your installation options module by using the CENTURY installation option (page 442). UNLOAD PLUS was shipped without an installation default value for this option and a value must be specified. 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.
Basic processing options

DRAIN_WAIT

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

The DRAIN_WAIT option specifies the drain timeout value to use. 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)

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 (page 446). 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**

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

The RETRY option specifies the maximum number of times that UNLOAD PLUS attempts to obtain a drain before the job terminates. The number of attempts can range from 0 through 255.

**Restriction**

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 (page 446). UNLOAD PLUS was shipped with a default value of 255 for this option. The command option overrides the default that is in the installation options module.

**RETRY_DELAY**

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

After a drain times out, the RETRY_DELAY 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.

**Restriction**

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 (page 446). UNLOAD PLUS was shipped with a default value of 1 for this option. The command option overrides the default that is in the installation options module.

DSPLOCKS

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

The DSPLOCKS option tells UNLOAD PLUS what action to take regarding displaying claims and locks if a drain attempt times out.

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 (page 447). UNLOAD PLUS was shipped with a default value of DRNFAIL for this option. The command option overrides the default that is in the installation options module.

DRNFAIL

DRNFAIL tells UNLOAD PLUS to display the claims and locks once, after the final attempt to obtain the drain times out.

NONE

NONE tells UNLOAD PLUS not to display any claims or locks.

RETRY

RETRY tells UNLOAD PLUS to display claims and locks after each drain timeout.
### Basic processing options

**Chapter 3 Syntax of the UNLOAD command**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FORCE</strong></td>
<td>This option is meaningful only when DIRECT YES is in effect.</td>
</tr>
<tr>
<td></td>
<td>The FORCE option specifies whether to cancel DB2 threads that might prevent a drain process from completing.</td>
</tr>
</tbody>
</table>

**NOTE**

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” on page 59.

**Specifying the default**

You can specify a default for the FORCE command option in your installation options module by using the FORCE installation option (page 449). 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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NONE</strong></td>
<td>FORCE NONE tells UNLOAD PLUS not to cancel DB2 threads that might prevent the drain process from completing.</td>
</tr>
<tr>
<td><strong>ALL</strong></td>
<td>FORCE ALL tells UNLOAD PLUS to cancel both read and write claimers at the point specified by the FORCE_AT option.</td>
</tr>
<tr>
<td><strong>FORCE_AT</strong></td>
<td>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.</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 (page 449). 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.)

START. FORCE_AT START tells UNLOAD PLUS to start canceling threads when the drain request begins.

RETRY. 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.

LASTRETRY. 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 (page 185) or DRNRETRY installation option (page 446).

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 (page 449). 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.

NO. NO tells UNLOAD PLUS not to display the report.
YES. 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.

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 (page 450). 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 “FORCE_AT” on page 187.

FORCE_DELAY
For information about this option, see “FORCE_DELAY” on page 188.

FORCE_RPT
UNLOAD PLUS ignores FORCE_RPT when you specify FORCE REPORTONLY.
OUTPUT syntax options

This section describes the options that you can specify in an OUTPUT descriptor statement. These options are in the order of the syntax diagrams (see “Common options for the OUTPUT command” on page 191, “Disk options for the OUTPUT command” on page 206, and “Tape options for the OUTPUT command” on page 214). Options that are specific to disk data sets and those that are specific to tape data sets are mutually exclusive.

Output

The OUTPUT option 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.

You can specify the defaults for these options at installation in the $ADUOUTP macro for dynamic allocation of output data sets (see Appendix A, “UNLOAD PLUS installation options” for details about $ADUOUTP and these installation 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.
Common options for the OUTPUT command

<table>
<thead>
<tr>
<th>Common options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UNIT</strong></td>
</tr>
<tr>
<td>$NO_UNITS</td>
</tr>
<tr>
<td><strong>DSNAME</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

| **UNITCNT**   | **integer** |
|              |             |

| **DSNTYPE**   | **PDS** |
|               | **LIBRARY** |
|               | **HFS** |
|               | **NONE** |
|               | **LARGE** |
|               | **BASIC** |
|               | **EXTREQ** |
|               | **EXTPREF** |

| **DIR**       | **integer** |
|              |             |

| **SUBSETS**   | **integer** |
|              |             |

| **LRECL**     | **integer** |
|              |             |

| **VOLCNT**    | **25** |
|              | **integer** |

| **GDGLIMIT**  | **integer** |
|              |             |

| **STORCLAS**  | **class** |
|              | **NONE** |

| **DATACLAS**  | **class** |
|              | **NONE** |

| **MGMTCLAS**  | **class** |
|              | **NONE** |

| **UNITCNT**   | **integer** |
|              |             |

This section describes options that you can use for output data sets that UNLOAD PLUS writes to disk or to tape.
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 (page 462). 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.

**$NO_\text{UNIT}$**
Specify $NO_\text{UNIT}$ to tell UNLOAD PLUS not to pass a unit value to dynamic allocation. This value is particularly useful in an SMS environment.
The DSNAME option specifies a data set name for the output data set.

**Unload data sets**

You can use any of the symbolic variables in Table 29 on page 194 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 (page 467). UNLOAD PLUS was shipped with one of the following values for this option:

- &USERID.&JOBNAME.&TS.&TYPE.&SELNUM (for unload data sets)
- &USREID.&JOBNAME.&TYPE.&SELNUM.P&POSNUM (for referenced files)

The command option overrides the default that is in the installation options module.

**Using symbolic variables**
Table 29 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 29 Symbolic variables for output data set names (part 1 of 2)**

<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>&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>
</tbody>
</table>
### Table 29  Symbolic variables for output data set names (part 2 of 2)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;PART&lt;sup&gt;a,b&lt;/sup&gt;</td>
<td>data set or partition that UNLOAD PLUS is unloading</td>
<td>3 bytes in either of the following circumstances:</td>
</tr>
<tr>
<td></td>
<td>Note the following information about &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 using &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&lt;sup&gt;a&lt;/sup&gt;</td>
<td>position number of the XML or LOB column within the select list (valid only for referenced file data sets)</td>
<td>3</td>
</tr>
<tr>
<td>&amp;SEC&lt;sup&gt;a,b&lt;/sup&gt;</td>
<td>current second (in the format SS)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;SECOND&lt;sup&gt;a,b&lt;/sup&gt;</td>
<td>current second (in the format SS)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;SELENUM&lt;sup&gt;a&lt;/sup&gt;</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>UNLOAD PLUS ignores PROC names.</td>
<td></td>
</tr>
<tr>
<td>&amp;TIME&lt;sup&gt;a,b&lt;/sup&gt;</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>
<tr>
<td>&amp;TYPE</td>
<td>data set type that UNLOAD PLUS is allocating</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>This symbol resolves to a value of SYSREC, SYSRED, or SYSREF (where SYSREF indicates a referenced file).</td>
<td></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</td>
<td>8 maximum</td>
</tr>
<tr>
<td></td>
<td>Note the following information about &amp;UTIL:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- UNLOAD PLUS truncates values longer than 8 bytes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Utility IDs that include special characters might cause UNLOAD PLUS to generate invalid names.</td>
<td></td>
</tr>
<tr>
<td>&amp;YEAR&lt;sup&gt;a,b&lt;/sup&gt;</td>
<td>current year (in the format YY)</td>
<td>2</td>
</tr>
</tbody>
</table>

<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 30 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 Table 30 illustrates a sample output descriptor.

**Table 30  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>$DBn</td>
</tr>
<tr>
<td>&amp;TS</td>
<td>$TSn</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.&OBID.&PART'
```

To the following data set name

```
ADU.$DB1.$TS1.000.000
```
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 62.

- For unload data sets (SYSREC and SYSRED), the remaining values for DSNTYPE tell UNLOAD PLUS what type of data set to allocate dynamically.

**Specifying the default**
You can specify the default for the DSNTYPE command option in your installation options module by using the DSNTYPE installation option (page 470). 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.

**PDS**

This option is valid only for referenced files.

PDS tells UNLOAD PLUS to dynamically allocate a partitioned data set (PDS) as your referenced file.

**LIBRARY**

This option is valid only for referenced files.

LIBRARY tells UNLOAD PLUS to dynamically allocate an extended partitioned data set (PDSE) as your referenced file.
HFS

This option is valid only for referenced files.

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.

NONE

This option is valid only for unload (SYSREC and SYSRED) data sets.

NONE tells UNLOAD PLUS to not use any extended attributes for this data set allocation.

LARGE

This option is valid only for unload (SYSREC and SYSRED) data sets.

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.

BASIC

This option is valid only for unload (SYSREC and SYSRED) data sets.

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.

EXTREQ

This option is valid only for unload (SYSREC and SYSRED) data sets.

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 (“UNIT” on page 192) supports extended attributes.

EXTPREF

This option is valid only for unload (SYSREC and SYSRED) data sets.

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 (“UNIT” on page 192) supports extended attributes.

**DIR**

DIR tells UNLOAD PLUS the number of 256-byte records to allocate for the directory of a new PDS or PDSE. 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**

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

The SUBSETS option tells UNLOAD PLUS how many data sets to allocate per base table space partition or per simple or segmented table space. 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 6), UNLOAD PLUS allocates the data sets listed in Figure 7.

**Figure 6  Sample OUTPUT statement with SUBSETS**

| OUTPUT LOBF2 DSNAME 'RDAJDM.UNLOAD.LOBFR2.P&PART' DSNTYPE PDS UNIT SYSDA SPACE 10,10 TRK DIR 10 SUBSETS 3 |

**Figure 7  Data set names resulting from SUBSETS option**

|----------------------------------|----------------------------------|----------------------------------|
**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 DSNNAME 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. If you specify USELRECL NO, the LRECL specification is ignored.

When you specify LRECL, the dynamically allocated data set will also be defined with a fixed record format.

**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.

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.

*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.

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 (see page 212). 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 (page 471). 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 (page 472). 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 \textit{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.

\textit{Specifying the default}
You can specify the default for the STORCLAS command option in your installation options module by using the STORCLAS installation option (page 473). 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. The value of \textit{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.

\textit{Specifying the default}
You can specify the default for the DATACLAS command option in your installation options module by using the DATACLAS installation option (page 473). 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. 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 (page 473). 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.

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.

NOTE

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. For more information, see page 303.

Specifying the default

You can specify the default for the UNITCNT command option in your installation options module by using the UNITCNT installation option (page 474). 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.
This section describes options that apply only to output data sets that UNLOAD PLUS writes to disk devices.
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 (page 474). 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 “MAXPRIM” on page 208.

**Specifying the default**
You can specify the default for the PCTPRIM command option in your installation options module by using the PCTPRIM installation option (page 474). 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.
OUTPUT syntax options

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.

Specify the default

You can specify the default for the MAXPRIM command option in your installation options module by using the MAXPRIM installation option (page 475). 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.
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 (page 475). 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 (page 476). 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.

NOTE
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.
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:

\[
\text{resultant\_file\_size} = \text{ROUND} (\text{calculated\_file\_size} \times \text{FILESZPCT}/100)
\]

**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 (page 476). 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 (page 477). 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. The value of integer must be from 1 through 9999.

**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.

**Specifying the default**

You can specify the default for the DISKRETN command option in your installation options module by using the DISKRETN installation option (page 477). 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. 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).

**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.

**Specifying the default**

You can specify the default for the DISKEXPD command option in your installation options module by using the DISKEXPD installation option (page 478). If you do not specify DISKEXPD, UNLOAD PLUS does not provide an expiration date. When you specify the DISKEXPD command option, it supersedes DISKRETN.
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. Table 31 describes the values that you can specify for this option.

**Table 31 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 (page 479). 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 (page 479). 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 (page 479). 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

- SELECT block
  - SELECT block detail
    - SELECT
      - * ' columnName constant CURRENT DATE CURRENT TIME CURRENT TIMESTAMP CURRENT RID
    - INTO
      - record options block field specification block
    - FROM
      - tableName creatorName ( OBID integer X'vebyteString'
    - WHERE condition
    - ORDER BY
      - columnName ASC DESC
    - ESTROWS integer OPTIONS ( PART partitionNumber )
  - ;
SELECT

The SELECT statement specifies the

- 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. See “Using JCL to specify multiple unload data sets” on page 289 for an example.

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 BMCLOAD, see page 157 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.
SELECT options

- 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.

* Specify an asterisk (*) to tell UNLOAD PLUS to select all columns in the table.

**NOTE**
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 time stamp 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.

  **NOTE**
  
  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
  , EMPLNAME CHAR (20),
  SALARY DECIMAL EXTERNAL (9,2),
FROM EMPL.TB1
```

Assuming 4 byte-RIDs, a sample output record would be in the following format:

```
RID  EMPLNAME CHAR (20)  SALARY DECIMAL EXTERNAL (9,2)
000  0  JOHN SMITH       50000.00
003  3
```

3-byte page number  ID map entry
SELECT options

INTO

The INTO option defines the output record. You can omit it 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 Appendix C, “Generating control statements for DB2 or other software products.”

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 32 on page 221. When using DIRECT YES, the default name is the name of the table being unloaded. When using DIRECT NO, the default name is the $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.
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. Table 33 on page 222 describes which strings to add for which conditions.
### 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 page 236 for a complete description, including restrictions.

---

#### Table 33  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>
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.
SELECT options

- 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.

**NOTE**

To specify the OBID, you must 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. Table 34 shows the result of the WHERE condition when you use the AND, OR, or NOT operator.

<table>
<thead>
<tr>
<th>Result of predicate</th>
<th>Result of WHERE condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p AND q</td>
<td>p OR q</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</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.
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.

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 Table 35.

### Table 35  Comparison operators (part 1 of 2)

<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>
</tbody>
</table>

\(^a\) Not supported for the IF predicate
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 “IF” on page 237.

- 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).

---

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>!=</code></td>
<td>not equal</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
</tr>
<tr>
<td><code>!&gt;</code></td>
<td>not greater than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
</tr>
<tr>
<td><code>IN</code></td>
<td>equal to any</td>
</tr>
<tr>
<td><code>NOT IN</code></td>
<td>not equal to all</td>
</tr>
<tr>
<td><code>IS NULL</code></td>
<td>column is null</td>
</tr>
<tr>
<td><code>IS NOT NULL</code></td>
<td>column is not null</td>
</tr>
<tr>
<td>LIKE <code>string</code></td>
<td>valid only in conjunction with the WHERE clause</td>
</tr>
</tbody>
</table>

This operator is a subset of the DB2 LIKE operator. Only CHARACTER, VARCHAR, BINARY, VARBINARY, GRAPHIC, and VARGRAPHIC columns are allowed.

**Note:** UNLOAD PLUS treats double quotes within a string as a single quote, and interprets this single quote as a literal character.
Additional considerations
Table 36 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 page 237.

Table 36  Rules for constants (part 1 of 2)

<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>decimal</td>
<td>identical to DB2 SQL decimal constants</td>
</tr>
<tr>
<td></td>
<td>The constant must be within the range of the column’s data type.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: 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.</td>
</tr>
<tr>
<td></td>
<td>Example of using the comma as a decimal point:</td>
</tr>
</tbody>
</table>
|                  | ```
SELECT * FROM ownerName.tableName
WHERE SALARY > 100,00
``` |
|                  | Example using the comma as both a decimal point and a separator: |
|                  | ```
SELECT * FROM ownerName.tableName
WHERE SALARY IN (100,00, 200,00, 300,00)
``` |
| character string | identical to DB2 SQL strings except that the length is limited to 255 bytes |
|                  | **Note**: UNLOAD PLUS does not validate character data against CCSIDs. |
|                  | Use ‘’ to denote an empty string. |
**Table 36  Rules for constants (part 2 of 2)**

<table>
<thead>
<tr>
<th>Type of constant</th>
<th>Additional details</th>
</tr>
</thead>
</table>
| hexadecimal strings | identical to DB2 SQL hexadecimal strings with the following exceptions:  
  - The length is limited to 255 bytes.  
  - You can specify hexadecimal strings for a binary column.  
  **Notes:**  
  - UNLOAD PLUS does not validate hexadecimal string data against CCSIDs.  
  - UNLOAD PLUS does not support the hexadecimal graphic string constants UX'xxxx' and GX'xxxx'. |
| binary strings | identical to DB2 SQL binary string constants except that the length is limited to 255 |
| graphic string | identical to DB2 SQL graphic string constants except that the length is limited to 255 |
| date/time string | with the following exceptions, identical to DB2 SQL date/time strings  
  **Exceptions:**  
  - Use only ISO, USA, EUR, and JIS formats.  
  - Precision on a timestamp constant must be less than or equal to the precision defined in the column.  
  **Note:** If you specify a time zone on a timestamp constant, this value overrides any value that you specify for the IMPLICIT_TZ option. |
| other | not supported |

**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 313 for sample output.

**NOTE**

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 (page 146). ORDER BY overrides ORDER YES.

Restriction
UNLOAD PLUS terminates if you specify a LOB or XML column on the ORDER BY option.

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 “Allocating SORTWK data sets in your JCL” on page 282.

- 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.
SELECT options

**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 131 for more details.

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

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

OPTIONS provides support for options that apply to a particular SELECT statement. 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.
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.TABLE
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.
<table>
<thead>
<tr>
<th>Field Specification Block Details</th>
<th>Page 236</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fieldName</strong></td>
<td>Page 240</td>
</tr>
<tr>
<td><strong>CHAR</strong></td>
<td></td>
</tr>
<tr>
<td><strong>VARCHAR</strong></td>
<td>Fill NO TRUNCATEb</td>
</tr>
<tr>
<td><strong>BLOB</strong></td>
<td>outputDescriptorName BINARYXML</td>
</tr>
<tr>
<td><strong>CLOB</strong></td>
<td>outputDescriptorName DBCLOBF</td>
</tr>
<tr>
<td><strong>GRAPHIC</strong></td>
<td>External (length) TRUNCATE</td>
</tr>
<tr>
<td><strong>VARGRAPHIC</strong></td>
<td>(length)</td>
</tr>
<tr>
<td><strong>SMALLINT</strong></td>
<td>(length,scale) ROUND FILLa NO YES</td>
</tr>
<tr>
<td><strong>INTEGER</strong></td>
<td>External (length,scale) ROUND FILLa NO YES</td>
</tr>
<tr>
<td><strong>BIGINT</strong></td>
<td></td>
</tr>
<tr>
<td><strong>decimal specification block</strong></td>
<td></td>
</tr>
<tr>
<td><strong>FLOAT</strong></td>
<td>External (length,scale) ROUND</td>
</tr>
<tr>
<td><strong>BINARY</strong></td>
<td></td>
</tr>
<tr>
<td><strong>VARBINARY</strong></td>
<td>(length) TRUNCATEb</td>
</tr>
<tr>
<td><strong>BINARY VARYING</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DATE</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TIME</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TIMESTAMP</strong></td>
<td></td>
</tr>
<tr>
<td><strong>BLOB</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CLOB</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DBCLOB</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DECFLOAT</strong></td>
<td></td>
</tr>
<tr>
<td><strong>XML</strong></td>
<td>External (length)</td>
</tr>
<tr>
<td><strong>EXIT</strong></td>
<td></td>
</tr>
</tbody>
</table>

*Not valid for BLOBF, CLOBF, and DBCLOBF subtype*
The field specification option defines a field in the output record. It 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 place holder. 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 the discussion starting on page 240. For allowable conversions, see Table 42 on page 255.

**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 consideration**
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 on page 226 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 DSNHDECPI IMPLICIT_TIMEZONE value.
Data type keywords

You can use the following keywords to specify data types. For more information about data types, see the sections that Table 37 describes.

**Table 37  Additional information about data types**

<table>
<thead>
<tr>
<th>Information</th>
<th>See reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>supported formats for date, time, and timestamp</td>
<td>tables that start on page 254</td>
</tr>
<tr>
<td>data types</td>
<td></td>
</tr>
<tr>
<td>valid data type conversions</td>
<td>page 255</td>
</tr>
<tr>
<td>default output data lengths</td>
<td>page 262</td>
</tr>
<tr>
<td>considerations for numeric EXTERNAL or CHARACTER fields</td>
<td>“Additional numeric EXTERNAL or CHARACTER field considerations” on page 263</td>
</tr>
<tr>
<td>considerations for numeric fields</td>
<td>“Additional numeric field considerations” on page 264</td>
</tr>
<tr>
<td>considerations for date, time, and timestamp fields</td>
<td>“Additional date, time, or timestamp field considerations” on page 265</td>
</tr>
<tr>
<td>considerations for VARCHAR, VARGRAPHIC, and VARBINARY</td>
<td>“Additional variable field considerations” on page 265</td>
</tr>
<tr>
<td>fields</td>
<td></td>
</tr>
<tr>
<td>considerations for ROWID fields</td>
<td>“Additional ROWID field considerations” on page 266</td>
</tr>
</tbody>
</table>

**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 “OUTPUT” on page 190.

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 “OUTPUT” on page 190.

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 “OUTPUT” on page 190.

You cannot unload XML data to a DBCLOB file.

(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 DSNTAUL.
- 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” on page 265.

**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 “OUTPUT” on page 190.

**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 “OUTPUT” on page 190.

**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 “OUTPUT” on page 190.

You cannot unload XML data to a DBCLOB file.

**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” on page 255.

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” on page 265.

**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” on page 264.

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” on page 263.

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” on page 264.

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” on page 263.
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” on page 264.

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” on page 263.

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” on page 264.

DECIMAL PACKED(*precision, 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(*precision, scale*)

This option is identical to DECIMAL PACKED.

DECIMAL ZONED(*precision, 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 ZONEDDECOVP=(F,F) (page 464).

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” on page 263.

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” on page 264.

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” on page 263.

**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” on page 265.
(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 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” on page 265.

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 Table 39 on page 254.

NOTE
When using DIRECT NO, specifying DATE-format EXTERNAL(length) requires extra consideration due to the various ways in which the data is returned from DB2. Often, the same DATE-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 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 yy specification and 99 has the first cc value prefixed to create a four-digit year. Any two-digit year that is 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**

This keyword specifies a timestamp field. For more information, see “Additional date, time, or timestamp field considerations” on page 265.

**TIMESTAMP**

TIMESTAMP specifies a DB2 internal timestamp format.

**TIMESTAMP EXTERNAL(length)**

This option specifies a DB2 timestamp string value.

**TIMESTAMP-format EXTERNAL(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. See Table 40 on page 254 for information about the available timestamp formats.

---

**NOTE**

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” on page 265.

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 Table 41 on page 255.

NOTE
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” on page 264.

**(`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.

**(`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.

Table 38 shows the return codes that UNLOAD PLUS expects in register 15.

<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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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 Appendix D, “UNLOAD PLUS user exits.”

*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 FPBFMODE 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. For allowable conversions, see Table 42 on page 255.
Supported DATE, TIME, and TIMESTAMP formats

The tables in this section 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.

**NOTE**
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 39 Date formats</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nondelimited</strong></td>
</tr>
<tr>
<td>Format number</td>
</tr>
<tr>
<td>Char/Int/Dec</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>

**Table 40 Timestamp formats**

<table>
<thead>
<tr>
<th><strong>Nondelimited</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Format number</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1E</td>
</tr>
<tr>
<td>2E</td>
</tr>
</tbody>
</table>

**a** ‘/’ can be any character.

**Table 40 Timestamp formats**

<table>
<thead>
<tr>
<th><strong>Non-delimited</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Format number</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1E</td>
</tr>
<tr>
<td>2E</td>
</tr>
</tbody>
</table>

**a** Formats are CHAR only except 3.

**b** The number of microseconds (N) can be from 0 through 12.
Supported data type conversions

The following sections describe the allowable data conversions and default output lengths. A blank cell indicates that the data type conversion is not allowed.

### Numeric output

Table 42 lists the conversions that UNLOAD PLUS supports to numeric output. Table 43 on page 257 lists the default output lengths for these conversions.

#### Table 42 Allowable data type conversions for numeric output fields (part 1 of 2)

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>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</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>INTEGER</td>
<td>X</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BIGINT</td>
<td>X</td>
<td>X</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>D</td>
<td>X</td>
<td>X</td>
<td>X</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>D</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X, d</td>
</tr>
<tr>
<td>CHAR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>VARCHARM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</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>
</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
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 42  Allowable data type conversions for numeric output fields (part 2 of 2)

<table>
<thead>
<tr>
<th>COLUMN</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>DECIMAL, FLOAT, or DECFLOAT EXTERNAL</th>
</tr>
</thead>
<tbody>
<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

---

\( ^d \) For integer and decimal columns, you can assign positive or negative overpunch values to decimal-zoned numeric output.

\( ^b \) 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.

\( ^c \) 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.

\( ^d \) 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.

---

256  **UNLOAD PLUS for DB2 Reference Manual**
### Table 43 Default lengths

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</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>DECIMAL or FLOAT EXTERNAL</th>
<th>DECFLOAT</th>
<th>DECFLOAT EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td></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>16</td>
<td>42</td>
</tr>
<tr>
<td>INTEGER</td>
<td></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>16</td>
<td>42</td>
</tr>
<tr>
<td>BIGINT</td>
<td></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>16</td>
<td>42</td>
</tr>
<tr>
<td>DECIMAL</td>
<td></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>16</td>
<td>42</td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>16</td>
<td>42</td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td></td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</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>DECIMAL or FLOAT EXTERNAL</th>
<th>DECFLOAT</th>
<th>DECFLOAT EXTERNAL</th>
</tr>
</thead>
<tbody>
<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></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></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>
<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>
<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></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>

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 DSNHDECQ format is LOCAL, the length is the local DATE and TIME length. If your DSNDECQ format is ISO, USA, EUR, or JIS, the default length for DATE is 10 and the default length for TIME is 8.
- **T** = See Table 39 on page 254, Table 41 on page 255, and Table 40 on page 254 for length information.

### Character, graphic, and binary output

Table 44 on page 258 lists the conversions that UNLOAD PLUS supports to character, graphic, and binary output. Table 45 on page 259 lists the default output lengths for these conversions.

---

**NOTE**

When you specify FORMAT CSV or FORMAT XML, UNLOAD PLUS does not support conversion to VARCHAR, VARGRAPHIC, BINARY, VARBINARY, or ROWID data types.
### Table 44  Allowable data type conversions for character, graphic, and binary output fields

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>CHAR or VARCHAR BLOBF</th>
<th>CHAR or VARCHAR CLOBF</th>
<th>CHAR or VARCHAR DBCLOBF</th>
<th>GRAPHIC</th>
<th>GRAPHIC EXTERNAL</th>
<th>VARGRAPHIC</th>
<th>BINARY</th>
<th>VARBINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>Xa</td>
<td>Xa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>Xa</td>
<td>Xa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>Xa</td>
<td>Xa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Xa</td>
<td>Xa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>Xa</td>
<td>Xa</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>
<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></td>
<td></td>
<td>D</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>D</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>X</td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>D</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>X</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>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td>Xb</td>
<td>X</td>
<td>X</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

---

- **a** 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.

- **b** When unloading XML data to a CHAR BLOBF or VARCHAR BLOBF field, you must specify BINARYXML.
**Table 45  Default lengths**

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CHAR</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>X</td>
</tr>
<tr>
<td>INTEGER</td>
<td>X</td>
</tr>
<tr>
<td>BIGINT</td>
<td>X</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>X</td>
</tr>
<tr>
<td>FLOT</td>
<td>X</td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>C</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>C</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>C</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>C</td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>10</td>
</tr>
<tr>
<td>TIME</td>
<td>8</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>26b</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>26b</td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
</tr>
<tr>
<td>XML</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 Table 39 on page 254, Table 41 on page 255, and Table 40 on page 254 for length information.

- **a** When unloading from CHAR to CHAR or VARCHAR to VARCHAR, and translating from EBCDIC or ASCII to Unicode using DIRECT YES, the default length is three times the length of the column.
- **b** For timestamp columns that are defined with precision, the default length of the output field is defined by the input value.

**Date, time, and timestamp output**

Table 46 on page 260 lists the conversions that UNLOAD PLUS supports to date, time, and timestamp output. Table 47 on page 261 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 46  Allowable data type conversions for date, time, and timestamp output fields

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DATE</th>
<th>DATE-EXTERNAL</th>
<th>DATE-format EXTERNAL</th>
<th>TIME</th>
<th>TIME-EXTERNAL</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>SMALLINT</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></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>
</tr>
<tr>
<td>DECIMAL</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></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>X</td>
<td></td>
<td></td>
<td>X</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>
</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>
</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>
</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></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>X</td>
<td></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></td>
<td>X</td>
<td>N</td>
<td></td>
<td>X</td>
<td>D</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>X</td>
<td>N</td>
<td></td>
<td>X</td>
<td>N</td>
<td></td>
<td>X</td>
<td>N</td>
<td>D</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
Table 47  Default lengths

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DATE</td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>4</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>4</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>4</td>
</tr>
<tr>
<td>TIME</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>4</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>4</td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td></td>
</tr>
<tr>
<td>XML</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 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 Table 39 on page 254, Table 41 on page 255, and Table 40 on page 254 for length information.

\(^{a}\) For EXTERNAL date/time fields, if you also specified DATEFMT, TIMEFMT, or TSFMT, the length defaults to the format element string length.

\(^{b}\) 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**

Table 48 on page 262 lists the conversions that UNLOAD PLUS supports to LOB and XML output. Table 49 on page 262 lists the default output lengths for these conversions.
### Table 48  Allowable data type conversions for LOB and XML output fields

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLOB</td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>BLOB</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The column values are:</td>
<td></td>
</tr>
<tr>
<td>X = allowable conversion</td>
<td></td>
</tr>
<tr>
<td>D = allowable conversion and indicates the default output data type, which is the data type of the column</td>
<td></td>
</tr>
<tr>
<td>N = allowable conversion but not supported with DIRECT NO</td>
<td></td>
</tr>
</tbody>
</table>

### Table 49  Default lengths (part 1 of 2)

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLOB</td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td></td>
</tr>
<tr>
<td>DECFLOAT</td>
<td></td>
</tr>
<tr>
<td>CHAR</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 Table 39 on page 254, Table 41 on page 255, and Table 40 on page 254 for length information.
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.

Table 50 on page 264 illustrates what you can expect when you use numeric EXTERNAL.
Additional numeric field considerations

This section 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.

### Table 50  Results from 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>leading ’-’ sign if negative</td>
</tr>
<tr>
<td>BIGINT EXTERNAL</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). 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. 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 +-.dddddddE±ee</td>
</tr>
<tr>
<td></td>
<td>The maximum field width is 15 for single precision and 24 for double precision.</td>
</tr>
<tr>
<td>DECFLOAT EXTERNAL</td>
<td>character representation of the decimal floating-point value</td>
</tr>
<tr>
<td></td>
<td>The maximum field width is 23 for a precision of 16 and 42 for a precision of 34.</td>
</tr>
</tbody>
</table>
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 (page 172) determines the rounding method.

Additional date, time, or timestamp field considerations

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

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

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

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 DSNXVTC.

   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 51 on page 267.
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**

Table 51 identifies the character conversions that UNLOAD PLUS supports for translation processing.

<table>
<thead>
<tr>
<th>Source Column</th>
<th>Target Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII SBCS</td>
<td>ASCII SBCS</td>
</tr>
<tr>
<td>ASCII MIXED</td>
<td>X</td>
</tr>
<tr>
<td>ASCII DBCS</td>
<td>X</td>
</tr>
<tr>
<td>EBCDIC SBCS</td>
<td>X</td>
</tr>
<tr>
<td>EBCDIC MIXED</td>
<td>X</td>
</tr>
<tr>
<td>EBCDIC DBCS</td>
<td>X</td>
</tr>
<tr>
<td>UNICODE SBCS</td>
<td>X</td>
</tr>
<tr>
<td>UNICODE MIXED</td>
<td>X</td>
</tr>
<tr>
<td>UNICODE DBCS</td>
<td>X</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 269.

**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 reformating 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 226.

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 (see page 152) 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. Table 52 on page 270 details when translation occurs with respect to the data type conversion that UNLOAD PLUS is performing.
### Table 52  Conversion and translation processing

<table>
<thead>
<tr>
<th>Column</th>
<th>SMALLINT</th>
<th>SMALLINT EXTERNAL</th>
<th>INTEGER</th>
<th>INTEGER EXTERNAL</th>
<th>BIGINT</th>
<th>BIGINT EXTERNAL</th>
<th>DECIMAL</th>
<th>DECIMAL EXTERNAL</th>
<th>DECIMAL ZONED</th>
<th>FLOAT</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>GRAPHIC, GRAPHIC EXTERNAL</th>
<th>VARCHARCHAR</th>
<th>VARCHARCHAR EXTERNAL</th>
<th>VARBINARY</th>
<th>DATE</th>
<th>DATE EXTERNAL</th>
<th>TIME</th>
<th>TIME-EXTERNAL</th>
<th>TIME-EXTERNAL WITH TIME</th>
<th>TIMESTAMP</th>
<th>TIMESTAMP-EXTERNAL</th>
<th>TIMESTAMP-EXTERNAL WITH TIME Zone</th>
<th>BLOB</th>
<th>CLOB</th>
<th>DBCLOB</th>
<th>DECFLOAT</th>
<th>DECFLOAT EXTERNAL</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>s</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>INTEGER</td>
<td>s</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>BIGINT</td>
<td>s</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>s</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>FLOAT</td>
<td>s</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>CHAR</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>VARCHAR</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>GRAPHIC</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>VARGRAPHIC</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>BINARY</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>VARBINARY</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>DATE</td>
<td>c</td>
<td>t</td>
<td></td>
<td>t</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>TIME</td>
<td>c</td>
<td>t</td>
<td></td>
<td>t</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>TIMESTAMP</td>
<td>c</td>
<td>t</td>
<td></td>
<td>t</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>TIMESTAMP WITH TIME ZONE</td>
<td>c</td>
<td>t</td>
<td></td>
<td>t</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>BLOB</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>CLOB</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>DBCLOB</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>DECFLOAT</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<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>
</tbody>
</table>

**Note:** 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 CCSID

*When using DIRECT NO, specifying DATE-EXTERNAL, TIMESTAMP-EXTERNAL, or TIME-EXTERNAL 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.*
Building and executing UNLOAD PLUS jobs

This chapter presents the following topics:

Building the UNLOAD PLUS job ................................. 271
  JOB statement .............................................. 272
  EXEC statement .......................................... 272
  REGION parameter ....................................... 272
  Utility parameters on the EXEC statement .......... 273
  STEPLIB DD statement .................................. 276
  UNLOAD PLUS DD statements ......................... 276
Running UNLOAD PLUS jobs ................................. 307
  Invoking UNLOAD PLUS .................................. 307
  Restarting UNLOAD PLUS ................................ 308
  Terminating or canceling an UNLOAD PLUS job .... 308

Building the UNLOAD PLUS job

Building a job for the UNLOAD PLUS for DB2 product involves creating a set of JCL that includes the following elements:

- a JOB statement (page 272)
- an EXEC statement with the appropriate utility parameters (page 272)
- STEPLIB or JOBLIB DD statements (page 276)
- DD statements for the appropriate number and size of data sets (page 276)
- UNLOAD PLUS control statements using the appropriate command syntax (page 87)

For examples of UNLOAD PLUS jobs, see Chapter 5, “Examples of UNLOAD PLUS jobs.”
JOB statement

Include an UNLOAD PLUS JOB statement that conforms to your site’s standards. You can include the REGION parameter to specify the region size on either your JOB statement or your EXEC statement. See “REGION parameter” for recommendations.

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 EXEC statement” on page 273.

You can use the REGION parameter to specify the region size on either your EXEC statement or your JOB statement. See “REGION parameter” 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 of available virtual storage to the UNLOAD PLUS job. If your data center does not permit you to specify REGION=0M, 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

**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.

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.
Utility parameters on the 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 (SSID) that indicates where the table space resides.

If you do not specify the SSID, UNLOAD PLUS uses the DB2 installation default from the DSNHDECP module. UNLOAD PLUS depends on the application defaults module being named DSNHDECP. If you do not specify an SSID and UNLOAD PLUS cannot find a module named DSNHDECP in your LINKLIST or STEPLIB, UNLOAD PLUS terminates.

UNLOAD PLUS supports the group attachment name capability. When you supply a group attachment name as the SSID, UNLOAD PLUS uses the name to connect all plans. UNLOAD PLUS then determines the actual DB2 SSID from within that group to use for the current unload job.

**NOTE**

When restarting in a data sharing environment, UNLOAD PLUS can use either the same member that it chose in the original unload job or any other member in the specified group.
Utility parameters on the EXEC statement

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, UNLOAD PLUS uses the default, userId.jobName. Each BMC utility job should have a unique utility ID.

NOTE
Utility IDs that include special characters might cause UNLOAD PLUS to generate invalid data set names when using dynamic allocation. For more information, see the discussion about using the utility ID variable with the DSNNAME option on page 193.

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 BMCUTIL 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 UNLOAD PLUS notifies after the completion of each phase and at the end of utility command execution.

Message level (MSGLEVEL)

This parameter controls which messages UNLOAD PLUS returns in the SYSPRINT and SYSPRINT2 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. For details, see Appendix A, “UNLOAD PLUS installation options.”
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, ADUSOPTS.

For more information about installation options, see Appendix A, “UNLOAD PLUS installation options.” For information about how to create multiple installation options modules, see the Installation System User Guide.

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 Table 53 on page 277 to quickly find the data set for which you want more information.
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 allocation” on page 60
- “OUTPUT” on page 190
- “Using dynamically allocated unload data sets” on page 293

### BMCFORCE data sets

UNLOAD PLUS generates a thread cancelation report when you specify one of the following options:

- FORCE REPORTONLY
- FORCE_RPT YES with FORCE ALL

You can specify a BMCFORCE DD statement in your JCL to allocate a data set to contain the thread cancelation report output. If you do not specify a BMCFORCE DD statement in your JCL, UNLOAD PLUS sends the report to your UNLOAD PLUS SYSPRINT.

<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>page 285</td>
<td>SYSIN</td>
<td>Not applicable</td>
</tr>
<tr>
<td>DDL definition (for input object)</td>
<td>page 278</td>
<td>DDLIN</td>
<td>DDLDDN (page 128)</td>
</tr>
<tr>
<td>input copy</td>
<td>page 279</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>message output</td>
<td>page 285</td>
<td>SYSPRINT</td>
<td>Not applicable</td>
</tr>
<tr>
<td>message output</td>
<td>page 285</td>
<td>SYSPRIN2</td>
<td>Not applicable</td>
</tr>
<tr>
<td>other</td>
<td>page 307</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>output control</td>
<td>page 284</td>
<td>SYSCNTL</td>
<td>CNTLDDN (page 145)</td>
</tr>
<tr>
<td>output data</td>
<td>page 286</td>
<td>SYSREC</td>
<td>UNLOADDDN (page 129)</td>
</tr>
<tr>
<td>secondary output data</td>
<td>page 286</td>
<td>SYSRED</td>
<td>UNLOADDDN (page 129)</td>
</tr>
<tr>
<td>sort message output indicator</td>
<td>page 307</td>
<td>UTPRINT</td>
<td>Not applicable</td>
</tr>
<tr>
<td>sort work</td>
<td>page 281</td>
<td>SORTWK</td>
<td>Not applicable</td>
</tr>
<tr>
<td>thread cancelation report output</td>
<td>page 277</td>
<td>BMCFORCE</td>
<td>Not applicable</td>
</tr>
<tr>
<td>VSAM linear input</td>
<td>page 280</td>
<td>VSAMDD</td>
<td>INFILE VSAMDDPREFIX</td>
</tr>
</tbody>
</table>
**DDLIN data set**

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.

**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 option (page 118).

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” on page 280.

**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.
UNLOAD PLUS DD statements

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

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 option (page 126). Use the VSAMDDPREFIX keyword 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 option (page 118).

**SORTWK data sets**

SORTWK data sets are the work files that BMCSORT uses. When you specify DIRECT YES, BMCSORT uses the 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 54 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. Table 54 on page 282 describes the results of combining these factors.
To ensure that BMCSORT has enough information to allocate SORTWK space accurately and efficiently, specify the ENUMROWS option (page 148) and a value greater than 0 for the SORTNUM option (page 149).

### 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.

<table>
<thead>
<tr>
<th>SORTNUM or SORTDEVN</th>
<th>Third parameter of BMCSORT DYNALOC</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTDEVT specified</td>
<td>ON or OFF</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>If the value is OFF, specifying a value greater than 0 for SORTNUM or specifying SORTDEVT changes this value to ON.</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>or</td>
<td>If the value is OFF, specifying a value greater than 0 for SORTNUM or specifying SORTDEVT changes this value to ON.</td>
<td></td>
</tr>
<tr>
<td>SORTDEVN not specified and SORTNUM 0</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 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

UNLOAD PLUS always requires a SYSCNTL data set if you specify the CNTLCARDS (page 137) or CNTLDDN option (page 145). 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 SYSCNTL{n} 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 SYSCNTL{n}, 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 set

UNLOAD PLUS always requires a SYSIN data set. This input data set contains the UNLOAD command and control statements. The UTILINIT phase reads, parses, and verifies the UNLOAD command that is provided in this 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.

SYSPRINT data set

UNLOAD PLUS always requires a SYSPRINT data set. This output data set contains UNLOAD PLUS messages.

**NOTE**

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.

UNLOAD PLUS 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, such as those from a QUIESCE utility, 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 UNLOAD PLUS displays and how to change the message level, see “Message level (MSGLEVEL)” on page 275.

SYSPRIN2 data set

The SYSPRIN2 data set is an optional output data set that contains the UNLOAD PLUS messages. The content of this data set 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.

**NOTE**

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.
UNLOAD PLUS 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, such as those from a QUIESCE utility, 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 UNLOAD PLUS displays and how to change the message level, see “Message level (MSGLEVEL)” on page 275.

**SYSREC and SYSRED data sets**

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 SYSRECNn 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.

---

**NOTE**

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 page 287.

---

**Overriding the default ddname**

To change the installation default ddname or ddname prefix SYSREC or SYSRED, use the UNLOADDN installation option (page 460) or command option (see page 129).

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 293.
Explicitly specify SYSREC and SYSRED DD statements in your JCL. Use this option when you want to control the allocation of these data sets.

**NOTE**

BMC does not recommend combining 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 “Unloading LOB and XML data” on page 62.

**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

<table>
<thead>
<tr>
<th>A = sum for each table or view:</th>
<th>B = sum for each table or view:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(# rows selected * (avg. output record length + 4 bytes if variable length))</td>
<td>(# rows selected * (max. output record length + 4 bytes if variable length))</td>
</tr>
</tbody>
</table>

Considerations
The following considerations apply to allocating SYSREC and SYSRED data sets:

- 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 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 (see page 134)
  - RECFM VB, which overrides the record format (see page 136)
  - MAXBLKSIZE, which overrides the block size (see page 135)

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. For more information, see “MAXBLKSIZE” on page 135.
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. For information about using multiple SELECT statements, see page 291.

Table 55 describes examples (shown in Figure 8 on page 290) of configurations for unloading partitions to separate data sets.

**Table 55 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 SYSREC(nn), 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 8  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
```

Directing selected partitions to separate data sets

```
//MYDD
//MYDD3
//MYDD4
UNLOAD
UNLOADDN MYDD
SELECT *
FROM
TABLE1
```

Directing all partitions to a single data set

```
//SYSREC
UNLOAD
SELECT *
FROM
TABLE1
```
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 SYSREC\(nn\), where \(nn\) is the numerical position of the SELECT statement.

Table 56 describes examples (shown in Figure 9 on page 292) of configurations for specifying multiple SELECT statements.

### Table 56  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 9   Directing multiple SELECT statements to one or more data sets

<table>
<thead>
<tr>
<th>Directing all SELECT statements to separate data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM TABLE1</td>
</tr>
<tr>
<td>SELECT * FROM TABLE2</td>
</tr>
<tr>
<td>SELECT * FROM TABLE2</td>
</tr>
<tr>
<td>SELECT * FROM TABLE3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Directing some SELECT statements to separate data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM TABLE1</td>
</tr>
<tr>
<td>SELECT * FROM TABLE2</td>
</tr>
<tr>
<td>SELECT * FROM TABLE2</td>
</tr>
<tr>
<td>SELECT * FROM TABLE3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Directing all SELECT statements to a single data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM TABLE1</td>
</tr>
<tr>
<td>SELECT * FROM TABLE2</td>
</tr>
<tr>
<td>SELECT * FROM TABLE2</td>
</tr>
<tr>
<td>SELECT * FROM TABLE3</td>
</tr>
</tbody>
</table>
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 (page 460).
   - To activate dynamic allocation for primary and secondary unload data sets by using command options, specify UNLOADDN\((primary, secondary)\) ACTIVE(YES,YES). For more information, see page 129.

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. For more information, see page 301.
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 (page 194 and page 294). For more information about dynamic allocation options, see “Dynamic allocation installation options” on page 464 and “OUTPUT syntax options” on page 190.

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 Table 29 on page 194.

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 correct and the second statement is correct.

Incorrect:

```
UNLOAD UNLOADDN (DATAOUT)  
OUTPUT DATAOUT UNIT SYSDA  
   DSNAME 'BMC.&DB.&TS.&JDAY'  
SELECT *  
   FROM BMC.EMPLS
```

Correct:

```
UNLOAD UNLOADDN (DATAOUT)  
OUTPUT DATAOUT UNIT SYSDA  
   DSNAME 'BMC.&DB.&TS.J&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 DSNAME keyword can cause UNLOAD PLUS to generate multiple files for an OUTPUT descriptor name, especially when you specify the &PART substitution variable in the DSNAME 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.

For details about the syntax of the OUTPUT command, see page 190. For information about the default output descriptor options, see page 464.
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 10  Example 1: No JCL DD statements: multiple SELECT statements with different output descriptors

```
//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 17 on page 299 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 11  Example 2a: No JCL DD statements: multiple SELECT statements with identical output descriptor

```
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM TABLE1
SELECT * FROM TABLE2
SELECT * FROM TABLE3
OUTPUT SYSREC   DSNAME '&USERID.SYSREC'
```

```
UNLOAD PLUS for DB2 Reference Manual
296
```
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 17 on page 299 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 17 on page 299 illustrates this scenario.

**Figure 12** 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 13** 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 17 on page 300 illustrates this scenario.

**Figure 14** 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 17 on page 300 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 17 on page 300 illustrates this scenario.

**Figure 15**  Example 4b: Mixture of JCL and dynamic allocation: multiple SELECT statements and using the &PART variable

```
//SYSREC01 DD DISP=(,CATLG)
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM NONPARTTB
SELECT * FROM PART3TB
OUTPUT SYSREC DSNAME '&USERID.P&PART'
```

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 16**  Example 5: Mixture of JCL and dynamic allocation: multiple SELECT statements with the &SELNUM variable

```
//SYSREC03 DD DISP=(,CATLG)
//SYSIN DD *
UNLOAD UNLOADDN(SYSREC) ACTIVE(YES)
SELECT * FROM TABLE1
SELECT * FROM TABLE2
SELECT * FROM TABLE3
OUTPUT SYSREC DSNAME '&USERID.SYSREC.S&SELNUM'
```

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 17 on page 301 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.
Example 1: Multiple SELECT statements with different output descriptors

- SELECT * FROM TABLE1
- SELECT * FROM TABLE2
- SELECT * FROM TABLE3

Example 2a: Multiple SELECT statements with identical output descriptor

- SELECT * FROM TABLE1
- SELECT * FROM TABLE2
- SELECT * FROM TABLE3

Example 2b: All SELECT statements to separate primary unload data sets (using the &SELNUM variable)

- SELECT * FROM TABLE1
- SELECT * FROM TABLE2
- SELECT * FROM TABLE3
Figure 17  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 produced from a DD statement specified in the JCL are shaded in this figure.
How UNLOAD PLUS calculates data set size during dynamic allocation

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. See “HURBA” on page 132 for more details.
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 211 for more details.

The following examples show how this hierarchy works. The examples use the following table information:

Table 57  Dynamic allocation examples data

<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>

Figure 18  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
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

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 211 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.
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.

**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.
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 set

UNLOAD PLUS always requires a UTPRINT data set if sorting is necessary. The presence of this data set tells UNLOAD PLUS to report sort messages. However, the actual messages for each sort process appear in separate SYS\textit{n} data sets, where \textit{n} is a system-assigned sequential number. You cannot specify a sequential or partitioned data set for UTPRINT. UTPRINT supports only SYSOUT data sets.

\textbf{WARNING}

JES3 users should be aware of a limitation within JES3 that does not allow concurrent tasks to share SYSOUT data sets. (See IBM® APAR OY23946 for a full description of this limitation.) 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 UNLOAD PLUS is multitasking its sort activity. If you attempt to use a nonsupported DD, you risk S1FB abends when concurrent sort tasks are running. JES3 version 4.2.1 users should also refer to IBM APARs OW00111 and OY63725.

Other data sets

Because it uses BMCSORT, UNLOAD PLUS 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 “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 about restart parameters, see 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 the following topics:

Overview .................................................. 309
Example 1: Unloading partitions to separate data sets ......................... 313
Example 2: Unloading columns in clustering order .......................... 320
Example 3: Unloading multiple tables to a single data set ................. 325
Example 4: Unloading while allowing read/write access to the table space .......................... 330
Example 5: Unloading data from a DSN1COPY .................................. 334
Example 6: Unloading a full image copy ...................................... 338
Example 7: Using FILTERPART ............................................ 342
Example 8: Using multiple SELECT statements .............................. 346
Example 9: Unloading from multiple table spaces ............................ 353
Example 10: Unloading from incremental image copies .................... 357
Example 11: Using a DDLIN data set ....................................... 361
Example 12: Unloading to CSV format ....................................... 366
Example 13: Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism .... 369
Example 14: Handling an abnormal termination ............................... 374
Example 15: Unloading LOB and XML data; using nonstandard null indicators .......................... 378
Example 16: Using dynamic allocation ........................................ 386
Example 17: Changing overpunch values in zoned decimal data ............ 392
Example 18: Unloading a table space and unloading data as Unicode ....... 396
Example 19: Unloading data to LOADPLUS .................................. 407

Overview

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 Chapter 3, “Syntax of the UNLOAD command.” For information about JCL statement requirements, see “Building the UNLOAD PLUS job” on page 271.

Use Table 58 to locate an example of a specific unload type or option. Use the chapter table of contents to find the example.

Table 58  Cross-reference of examples by function  (part 1 of 3)

<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>
</tbody>
</table>
### Table 58  Cross-reference of examples by function  (part 2 of 3)

<table>
<thead>
<tr>
<th>Functions</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<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><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>DELETEFILES</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>DSNAME</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>
</tbody>
</table>
## Table 58  Cross-reference of examples by function  (part 3 of 3)

<table>
<thead>
<tr>
<th>Functions</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<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>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>UNLOADADDN</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>ZONEDDECOVP</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. Table 59 describes the key command options for this job.

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.

Table 59   Command options for example 1

<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>

Figure 21 shows the JCL for example 1.

Figure 21   JCL for example 1 (part 1 of 2)

```sql
// JOB //UNLOAD01 EXEC PGM=ADUUMAIN,REGION=0M,COND=EVEN, // PARM=(DEHJ,'ADUXM01','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 //SYSRECO1 DD DSN=ADU.EXAMPLE1.SYSRECO1, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
```
Table 60, Table 61 on page 315, Table 62 on page 315, and Table 63 on page 316 show the records that the job unloads in example 1. A NULL indicator field (NUL in the table row headings) follows each nullable field. The numbers given for POS indicate the start and end positions for each field.

Table 60  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</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>POS(1,5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSN</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>POS(6,14)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS(15,24)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC_NULL1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS(25)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT_NO</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>POS(26,29)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOB_CODE</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>POS(30,31)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC_NULL2</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS(32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARY_CODE</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>POS(33)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARY</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>POS(34,42)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC_NULL3</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS(43)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PART401K</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>POS(44)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC_NULL4</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS(45)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_LNAME</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>POS(*,46)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_FNAME</td>
<td>Raymond</td>
<td>Samuel</td>
<td>Kathy</td>
<td>William</td>
<td>David</td>
<td>Randy</td>
<td>Ellen</td>
<td></td>
</tr>
<tr>
<td>POS(<em>,</em>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_MNAME</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>
<tr>
<td>POS(<em>,</em>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 61  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</td>
<td>15.</td>
<td>13.</td>
<td>9.</td>
<td>2.</td>
<td>7.</td>
</tr>
<tr>
<td>SSN</td>
<td>837625104</td>
<td>463762574</td>
<td>737626464</td>
<td>288738465</td>
<td>958473673</td>
</tr>
<tr>
<td>HIRE_DATE</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>BMC_NULL1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT_NO</td>
<td>3200</td>
<td>3300</td>
<td>3600</td>
<td>4200</td>
<td>4300</td>
</tr>
<tr>
<td>JOB_CODE</td>
<td>44</td>
<td>84</td>
<td>93</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>BMC_NULL2</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARY_CODE</td>
<td>H</td>
<td>S</td>
<td>S</td>
<td>H</td>
<td>S</td>
</tr>
<tr>
<td>SALARY</td>
<td>6.25</td>
<td>32000.00</td>
<td>44500.00</td>
<td></td>
<td>32000.00</td>
</tr>
<tr>
<td>BMC_NULL3</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PART401K</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BMC_NULL4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_LNAME</td>
<td>Bronson</td>
<td>Doe</td>
<td>Welch</td>
<td>Britain</td>
<td>Franklin</td>
</tr>
<tr>
<td>EMPL_FNAME</td>
<td>Parker</td>
<td>John</td>
<td>John</td>
<td>Michelle</td>
<td>Erica</td>
</tr>
<tr>
<td>EMPL_MNAME</td>
<td>James</td>
<td>Frank</td>
<td>Elaine</td>
<td>Jean</td>
<td></td>
</tr>
</tbody>
</table>

### Table 62  Sample output for SYSREC03 (part 1 of 2)

<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</td>
<td>3.</td>
<td>18.</td>
<td>14.</td>
<td>6.</td>
<td>7</td>
</tr>
<tr>
<td>SSN</td>
<td>828382664</td>
<td>182864652</td>
<td>376275478</td>
<td>868474632</td>
<td>463984856</td>
</tr>
<tr>
<td>HIRE_DATE</td>
<td>04/03/1992</td>
<td>05/19/1989</td>
<td>07/31/1991</td>
<td></td>
<td>12/19/1989</td>
</tr>
<tr>
<td>BMC_NULL1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT_NO</td>
<td>5100</td>
<td>5100</td>
<td>5200</td>
<td>6300</td>
<td>6700</td>
</tr>
<tr>
<td>JOB_CODE</td>
<td>13</td>
<td>24</td>
<td>43</td>
<td>13</td>
<td>73</td>
</tr>
<tr>
<td>BMC_NULL2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARY_CODE</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>SALARY</td>
<td>5.75</td>
<td>7.50</td>
<td>6.25</td>
<td></td>
<td>6.90</td>
</tr>
</tbody>
</table>
### Table 62  
Sample output for SYSREC03 (part 2 of 2)

<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>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 63  
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>
<tr>
<td>DEPT_NO POS(26,29)</td>
<td>8200</td>
<td>8200</td>
</tr>
<tr>
<td>JOB_CODE POS(30,31)</td>
<td>82</td>
<td>84</td>
</tr>
<tr>
<td>BMC_NULL2 POS(32)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARY_CODE POS(33)</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>SALARY POS(34,42)</td>
<td>36000.00</td>
<td>45000.00</td>
</tr>
<tr>
<td>BMC_NULL3 POS(43)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PART401K POS(44)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BMC_NULL4 POS(45)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPL_LNAME POS(46,*)</td>
<td>Kennedy</td>
<td>Wilson</td>
</tr>
<tr>
<td>EMPL_FNAME POS(<em>,</em>)</td>
<td>Jennifer</td>
<td>Lindsay</td>
</tr>
<tr>
<td>EMPL_MNAME POS(<em>,</em>)</td>
<td>Ann</td>
<td>Ann</td>
</tr>
</tbody>
</table>
Figure 22 shows the SYSPRINT for example 1.

**Figure 22  SYSPRINT for example 1 (part 1 of 3)**

```sql
<table>
<thead>
<tr>
<th>BMC50047I</th>
<th>UNLOAD PLUS FOR DB2 V11R1.00</th>
<th>**** B M C   U N L O A D   P L U S   F O R   D B 2   V11R1.00 ****</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50047I</td>
<td>UTILITY EXECUTION STARTING 10/28/2013 12:59:34</td>
<td>(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>z/OS 2.1.0, PID=H8B7790, DFSMS FOR Z/OS 2.1.0 DB=11.1.0</td>
<td></td>
</tr>
<tr>
<td>BMC50471I</td>
<td>REGION=QM, BELOW 16M=8844K, ABOVE 16M=1413300K, IEFSU=NO, CPUS=3</td>
<td></td>
</tr>
<tr>
<td>BMC50471I</td>
<td>MEMLIMIT=1759218604320M, AVAILABLE=175921860431M, MEMLIMIT SET BY: REGION=0</td>
<td></td>
</tr>
</tbody>
</table>

BMC50471I | DB2 UTILITIES COMMON CODE -- V11.01.00 | NO MAINTENANCE TO REPORT |
BMC50471I | BMC50471I | MSCICK ENGINE -- V02.04.01 |
BMC50471I | BMC50471I | BMC STATS API -- V11.01.00 |
BMC50471I | BMC50471I | DB2 UTILITIES COMMON CODE -- V11.01.00 |
BMC50471I | BMC50471I | SOLR=NONE |
BMC50471I | BMC50471I | FORCE_AT=(START,3) |
BMC50471I | BMC50471I | SHRLEVEL=REFERENCE |

BMC50471I | TAPES=NONE |
BMC50470I | OUTPUT = SYSREC |
BMC50470I | UNIT = SYSALDA |
BMC50470I | VOLCNT = 25 |
BMC50470I | GDGLIMIT = 5 |
BMC50470I | GDGEMPTY = NO |
BMC50470I | GDGSCRAT = NO |
BMC50470I | STORCLAS = NONE |
BMC50470I | DATACLAS = NONE |
BMC50470I | MGMTCLAS = NONE |
BMC50470I | UNITCNT = 0 |
BMC50470I | SPACE = CYL |
BMC50470I | PCTPRIM = AUTO |
BMC50470I | MAXPRIM = 0 |
BMC50470I | MAXSEQ = 0 |
BMC50470I | FILESZPCT = 100 |
BMC50470I | NBRSEQ = AUTO |
BMC50470I | DISKRET = NONE |
BMC50470I | DISKEXP = NONE |
BMC50470I | RETPD = NONE |
BMC50470I | EXPDT = 99000 |
BMC50470I | TRTCH = NONE |
BMC50470I | DSNTYPE = NONE |
BMC50483I | SYSREC = VOLUMES=NONE |
BMC50483I | SYSRED = VOLUMES=NONE |
BMC50483I | SYREF = VOLUMES=NONE |

BMC50483I | DSNAME=USERID.ATYPE.SSELECTNUM |
BMC50483I | DSNAME=USERID.ATYPE.SSELECTNUM |
BMC50483I | DSNAME=USERID.ATYPE.SSELECTNUM |
```

Chapter 5  Examples of UNLOAD PLUS jobs  317
Example 1: Unloading partitions to separate data sets

Figure 22  SYSPRINT for example 1 (part 2 of 3)

```
BMC50471I DB2 DSNHDECP MODULE SETTINGS:
BMC50471I VERSION = 1110
BMC50471I SUBSYSTEM DEFAULT = DBU
BMC50471I CHARACTER SET = ALPHANUM
BMC50471I DATE FORMAT = USA
BMC50471I TIME FORMAT = USA
BMC50471I LOCAL DATE LENGTH = 0
BMC50471I LOCAL TIME LENGTH = 0
BMC50471I DECIMAL POINT = PERIOD
BMC50471I DECIMAL ARITHMETIC = 15
BMC50471I DELIMITER = DEFAULT
BMC50471I SQL DELIMITER = 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 UNICODE CCSID = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)

BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL = '/BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC = '/BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST = '/BMCUTIL.CMN_BMCHIST'
BMC50471I BMC_BMCXCOPY = '/BMCUTIL.CMN_BMCXCOPY'

BMC50102I *
BMC50102I UNLOAD SELECT *
BMC50102I INTO
BMC50102I EMPLOYEE_ID DECIMAL EXTERNAL (5,0),
BMC50102I ,
BMC50102I ,
BMC50102I JOB_CODE SMALLINT EXTERNAL (2,0),
BMC50102I ,
BMC50102I SALARY DECIMAL EXTERNAL (9,2),
BMC50102I ,
BMC50102I ,
BMC50102I ,
BMC50102I FROM BMC.EMPLS
BMC50102I *

BMC5164I DIRECT YES IN EFFECT
BMC5167I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 1 IS 8 FROM DB2STATS
BMC5167I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 2 IS 5 FROM DB2STATS
BMC5167I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 3 IS 5 FROM DB2STATS
BMC5167I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 4 IS 2 FROM DB2STATS
BMC5168I ESTIMATED ROWS FOR TABLE BMCEXDB.EMPLS IS 20 FROM DB2STATS
BMC5168I I ESTIMATED ROWS FOR SELECT 1 IS 20

BMC5004I UTILINIT PHASE COMPLETE.  ELAPSED TIME = 00:00:00
BMC5163I FOR DONNAME 'SYSRECO1' DSN=DBU.EXAMPLE1.SYSRECO1.DCB=(RECFM=VB,BLKSIZE=27993,RECL=117)
BMC5163I FOR DONNAME 'SYSRECO2' DSN=DBU.EXAMPLE1.SYSRECO2.DCB=(RECFM=VB,BLKSIZE=27993,RECL=117)
BMC5163I FOR DONNAME 'SYSRECO3' DSN=DBU.EXAMPLE1.SYSRECO3.DCB=(RECFM=VB,BLKSIZE=27993,RECL=117)
BMC5163I FOR DONNAME 'SYSRECO4' DSN=DBU.EXAMPLE1.SYSRECO4.DCB=(RECFM=VB,BLKSIZE=27993,RECL=117)

BMC5084I PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 12:59:34
BMC5065I ESTABLISHING A POINT OF CONSISTENCY FOR:
BMC5065I SPACE BMCEXDB.BMCEXATS PART(0) PSID(0002)
BMC5084I PAGE EXTERNALIZATION PROCESS COMPLETE.  ELAPSED TIME = 00:00:00
BMC504I 0: ZIIP ENABLED (0) USING XBMB SUBSYSTEM XBMB
BMC504I BELOW 16M = 8452K, ABOVE 16M = 1403792K, CPU = 3
BMC5170I MAX TASKS = 4, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC5047I 4: PARTITION = 4, ROWS/KEYS = 2, I/O WAITS = 2, DONNAME = SYS00006
BMC5047I 4: RDB LOCK WAITS = 0
BMC5047I 2: PARTITION = 2, ROWS/KEYS = 5, I/O WAITS = 2, DONNAME = SYS00002
BMC5047I 2: RDB LOCK WAITS = 0
BMC5047I 1: PARTITION = 1, ROWS/KEYS = 8, I/O WAITS = 2, DONNAME = SYS00003
```
### Example 1: Unloading partitions to separate data sets

**Figure 22**  SYSPRINT for example 1 (part 3 of 3)

```
BMC50478I 3: RDB LOCK WAITS = 0
BMC50478I 3: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 2, DDNAME = SYS00004
BMC50476I DDNAME = SYSREC01, 1/05 = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC02, 1/05 = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC03, 1/05 = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC04, 1/05 = 1, I/O WAITS = 1, RDB LOCK WAITS = 0

BMC51671I UNLOAD STATISTICS: 8 ROWS PROCESSED FROM PARTITION 1
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A001' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 2
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A002' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 3
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A003' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A004' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 20 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 0 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 0 RECORDS WRITTEN TO DDNAME 'SYSREC01'
BMC51674I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYSREC02'
BMC51674I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYSREC03'
BMC51674I UNLOAD STATISTICS: 2 RECORDS WRITTEN TO DDNAME 'SYSREC04'
BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPLE1.SYSCNTL,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)
BMC51801I LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'

```
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.

Table 64 describes the key command options for this job.

### Table 64  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>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>

Figure 23 on page 321 shows the JCL for example 2.
Example 2: Unloading columns in clustering order

Chapter 5 Examples of UNLOAD PLUS jobs

Table 65 on page 322 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.

Figure 23 JCL for example 2

```
//        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
//UTPRINT  DD   SYSOUT=*        
//SYSPRINT DD   SYSOUT=*        
```
Example 2: Unloading columns in clustering order

Table 65  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>182864562</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_HRLY</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>

Figure 24 shows the SYSPRINT for example 2.

**Figure 24** SYSPRINT for example 2 (part 1 of 3)

---

 **** B M C   U N L O A D   P L U S   F O R   D B 2   V11R1.00 ****

(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.

BMC50001I UTILITY EXECUTION STARTING 10/28/2013 13:00:09 ...
BMC50002I UTILITY ID = 'ADUXM02'. 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=1413308K,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
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT: BPJ0661 BPJ0667 BPJ0671 BPJ0675 BPJ0676 BPJ0682 BPJ0686 BPJ0689 BPJ0697
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAINT: BPJ0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I MAINT: BPUS409 BPUS534 BPUS674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I MAINT: BPE0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416

BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FORCE_RPT=NO SMAX=16
BMC50471I CENTURY=(1950,2049) FORCE=NONE SMCORE=(0K,0K)
BMC50471I CHANGE_CONSISTENT=NO HISTORY=YES SORTNUM=32
BMC50471I CHANGE_QUESICE=NO IBUFFS=25 SODELAY=3
BMC50471I CMAK=16 INLINI=NO SOLRETRY=100
BMC50471I CMRATIO=50 LOADEDCP=NO TAPEDISP=DELETE
BMC50471I CONSTRAINTS=BMC LDKCROW=YES TASKMAX=200%
BMC50471I CURRENTDEGREE=NONE MAXP=5 UBUFFS=25
BMC50471I DELFILES=YES MSGLEVEL=1 UNLOAD=2000%
BMC50471I DRNDELAY=1 NULLCHAR=X'6F' UNLOADN_ACTIVE=(YES,NO)
BMC50471I DRNRETRY=255 NULLTYPE=T1 UNLOADN=(SYSREC,SYSRED)
BMC50471I DRNWAIT=NONE OPMDB2ID=YES USEREC=NO
BMC50471I DSPLOCKS=DRNFAIL PLAN=ADUQA UXSTATE=SUP
BMC50471I EXCLUDM=(X37,X22,X06) RECFM=AUTO WORKUNIT=SYSALDSD
BMC50471I FILEREDON=SYSREF ROWSETS=100 ZIIP=ENABLED
BMC50471I FILL=NO SDUMP=YES ZFEDDECOPV=(C,D)
BMC50471I FORCE_AT=(START,3) SHRLEVEL=REFERENCE

BMC50471I TAPES=NONE

BMC50470I OUTPUT = SYSREC SYSREC SYSREC
BMC50470I UNIT = SYSALDSD SYSALDSD SYSALDSD
BMC50470I VLOCNT = 25 25 25
BMC50470I GDGMAX = 5 5 5
BMC50470I GDGEMPTY = NO NO NO

---

UNLOAD PLUS for DB2 Reference Manual
Example 2: Unloading columns in clustering order

Chapter 5 Examples of UNLOAD PLUS jobs

Figure 24  SYSPRINT for example 2 (part 2 of 3)

```
BMC50470I GDGSCRAT = NO
BMC50470I STORCLAS = NONE
BMC50470I DATACLAS = NONE
BMC50470I MGMTCLAS = NONE
BMC50470I UNITCNT  = 0
BMC50470I SPACE    = CYL
BMC50470I PCTPRIM  = AUTO
BMC50470I MAXPRIM  = 0
BMC50470I MAXSECD = 0
BMC50470I FILESZPCT  = 100
BMC50470I NBRSECD = AUTO
BMC50483I SYSREC VOLUMES=NONE
BMC50483I SYSRED VOLUMES=NONE
BMC50483I SYSREF VOLUMES=NONE
BMC50483I SYSREC DSNAME=&USERID.&TYPE.S&SELNUM
BMC50483I SYSRED DSNAME=&USERID.&TYPE.S&SELNUM
BMC50483I SYSREF DSNAME=&USERID.BMC
BMC50471I DB2 DSNHDECP MODULE SETTINGS:
BMC50471I VERSION                 = 1110
BMC50471I SUBSYSTEM DEFAULT       = DEHJ
BMC50471I CHARACTER SET           = ALPHANUM
BMC50471I DATE FORMAT             = USA
BMC50471I TIME FORMAT             = USA
BMC50471I LOCAL DATE LENGTH       = 0
BMC50471I LOCAL TIME LENGTH       = 0
BMC50471I DECIMAL POINT           = PERIOD
BMC50471I DECIMAL ARITHMETIC      = 15
BMC50471I DELIMITER               = DEFAULT
BMC50471I SQL DELIMITER           = 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 UNICODE CCSID           = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE      = CURRENT (-05:00)
BMC50102I UNLOAD
BMC50102I ORDER YES
BMC50102I LIMIT 2
BMC50102I FIXEDVARCHAR YES
BMC50102I SELECT DEPT_NO,
BMC50102I  EMPL_LNAME,
BMC50102I  SSN,
BMC50102I  HIRE_DATE,
BMC50102I  JOB_CODE,
BMC50102I  SALARY_HRLY,
BMC50102I  PART_401K
BMC50102I INTO
BMC50102I HIRE_DATE IF NULL VALUE('01/01/1991'),
BMC50102I JOB_CODE IF NULL VALUE(''),
BMC50102I SALARY_HRLY IF NULL VALUE(999.99),
BMC50102I PART_401K
BMC50102I IF PART_401K = '0' VALUE('N')
BMC50102I IF PART_401K = 'Y' VALUE('Y')
BMC50102I IF PART_401K = NULL VALUE ('N')
```
Figure 24  SYSPRINT for example 2 (part 3 of 3)

Example 2: Unloading columns in clustering order

<table>
<thead>
<tr>
<th>BMC50102I</th>
<th>FROM BMC.EMPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC51654I</td>
<td>DIRECT YES IN EFFECT</td>
</tr>
<tr>
<td>BMC51687I</td>
<td>ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 1 IS 8 FROM DB2STATS</td>
</tr>
<tr>
<td>BMC51687I</td>
<td>ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 2 IS 5 FROM DB2STATS</td>
</tr>
<tr>
<td>BMC51687I</td>
<td>ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 3 IS 5 FROM DB2STATS</td>
</tr>
<tr>
<td>BMC51688I</td>
<td>ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 4 IS 2 FROM DB2STATS</td>
</tr>
<tr>
<td>BMC51689I</td>
<td>ESTIMATED ROWS FOR SELECT 1 IS 8</td>
</tr>
<tr>
<td>BMC50004I</td>
<td>UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:01</td>
</tr>
<tr>
<td>BMC51639I</td>
<td>FOR DDNAME 'SYSREC' DSN=ADU.EXAMPLE2.SYSREC,DCB=(RECFM=FB,BLKSIZE=27984,LRECL=53)</td>
</tr>
<tr>
<td>BMC50934I</td>
<td>UNABLE TO LOCATE SORT WORK DATASETS, DDNAME = 'SORTWKNN'</td>
</tr>
<tr>
<td>BMC50004I</td>
<td>UNLOAD PHASE COMPLETE. ELAPSED TIME = 00:00:01</td>
</tr>
<tr>
<td>BMC51639I</td>
<td>FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPLE2.SYSCNTL,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)</td>
</tr>
</tbody>
</table>

DDNAME = SYSCNTL, I/OS = 4, I/O WAITS = 4, RDB LOCK WAITS = 0

DDNAME = SYSCNTL, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0

Figure24 SYSPRINT for example 2 (part3 of3)
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 66 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 66  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>
<tr>
<td>ORDER YES</td>
<td>sorts output records in data-sorting key sequence</td>
</tr>
<tr>
<td>SORTDEVT SYSDA</td>
<td>specifies device SYSDA for dynamically allocated sort work files</td>
</tr>
<tr>
<td>INTO</td>
<td>specifies the table in the LOAD control statements that UNLOAD PLUS generates in SYSCNTL</td>
</tr>
<tr>
<td>FROM</td>
<td>names tables from which to unload data</td>
</tr>
</tbody>
</table>

Figure 25 shows the JCL for example 3.

Figure 25  JCL for example 3 (part 1 of 2)
```sql
// 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_SALCODE
```
Example 3: Unloading multiple tables to a single data set

**Figure 26** shows the SYSPRINT for example 3.

---

---
### Example 3: Unloading multiple tables to a single data set

**Figure 26**  SYSPRINT for example 3 (part 2 of 5)

<table>
<thead>
<tr>
<th>BMC50471I</th>
<th>CHANGE_QUIESCE=NO</th>
<th>IBUFFS=25</th>
<th>SOLQRELAY=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I</td>
<td>CMAx=16</td>
<td>INLINE=NO</td>
<td>SOLQRETRY=100</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>CMRATIO=50</td>
<td>LOADDECP=NO</td>
<td>TAPEDISP=DELETE</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>CONSTRILES=BMC</td>
<td>LOCKROW=YES</td>
<td>TASKMAX=2000</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>CURRENTDEGREE=None</td>
<td>MAXP=5</td>
<td>UBUFFS=25</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DELFILES=YES</td>
<td>MSGLEVEL=1</td>
<td>UNLDMAX=2000</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DRNDELAY=1</td>
<td>NULLCHAR=X’6F’</td>
<td>WORKUNIT=SYSALLDA</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DRNRETRY=255</td>
<td>NULLTYPE=T1</td>
<td>ZIIP=ENABLED</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DRNWAIT=None</td>
<td>OPNDB2ID=YES</td>
<td>ZONEDDECOVP=(C,D)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DSPLOCKS=DRNFAIL</td>
<td>PLAN=AQDA</td>
<td>ZONEDRECOVP=ONE</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>EXCLUMP=(X37,X22,X06)</td>
<td>PLAN=AQDA</td>
<td>ZONEDRECOVP=(C,D)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>FILESETN=SYSREC</td>
<td>RECFM=AUTO</td>
<td>ZONEDRECOVP=ONE</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>FILEREFDN=SYSREF</td>
<td>SDUMP=YES</td>
<td>ZONEDRECOVP=ONE</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>FORCE_AT=(START,3)</td>
<td>SHRLEVEL=REFERENCE</td>
<td></td>
</tr>
<tr>
<td>BMC50471I</td>
<td>TAPES=NONE</td>
<td>SYSREC</td>
<td>VOLUMES=NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>OUTPUT   = SYSREC</td>
<td>SYSRED</td>
<td>VOLUMES=NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNIT     = SYSALLDA</td>
<td>SYSALLDA</td>
<td>VOLUMES=NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>VOLCNT   = 25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGLIMIT = 5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGEMPTY = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GODSCRAT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>STORCLAS = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DATACLAS = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>MGMTCLAS = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNITCNT  = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SPACE    = CYL</td>
<td>CYL</td>
<td>CYL</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>PCTPRIM  = AUTO</td>
<td>AUTO</td>
<td>AUTO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>MAXPRIM  = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>MAXSECQ = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>FILESZPCT 100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>NBRSECO = AUTO</td>
<td>AUTO</td>
<td>AUTO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DISKRETN = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DISKEXPD = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>RETPD = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>EXPDT   = 99000</td>
<td>99000</td>
<td>99000</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>TRTCH = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DSNTYPE = NONE</td>
<td>NONE</td>
<td>POS</td>
</tr>
<tr>
<td>BMC50483I</td>
<td>SYSREC</td>
<td>VOLUMES=NONE</td>
<td></td>
</tr>
<tr>
<td>BMC50483I</td>
<td>SYSCRED</td>
<td>VOLUMES=NONE</td>
<td></td>
</tr>
<tr>
<td>BMC50483I</td>
<td>SYSREF</td>
<td>VOLUMES=NONE</td>
<td></td>
</tr>
<tr>
<td>BMC50483I</td>
<td>DSNAME=&amp;USERID.ATYPE.&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I</td>
<td>DSNAME=&amp;USERID.ATYPE.&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I</td>
<td>DSNAME=&amp;USERID.BMC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BMC50471I**  DB2 DSNHDECP MODULE SETTINGS:

<table>
<thead>
<tr>
<th>BMC50471I</th>
<th>VERSION</th>
<th>DEHJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I</td>
<td>DATE_FORMAT</td>
<td>USA</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>TIME_FORMAT</td>
<td>USA</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>LOCAL_DATE_LENGTH</td>
<td>0</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DECIMAL_POINT</td>
<td>PERIOD</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DECIMAL_ARITHMETIC</td>
<td>15</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DELIMITER</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>SOL_DELIMITER</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>ENCODING_SCHEME</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>APPL. ENCODING_SCHEME</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>MIXED</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>EBCDIC_CSSID</td>
<td>(37,65534,65534)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>ASCII_CSSID</td>
<td>(819,65534,65534)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>UNICODE_CSSID</td>
<td>(367,1208,1200)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>IMPLICIT_TIME_ZONE</td>
<td>CURRENT (-05:00)</td>
</tr>
</tbody>
</table>

**BMC50528I**  DB2 MODE = NFM

<table>
<thead>
<tr>
<th>BMC50471I</th>
<th>BMC_BMCUTIL</th>
<th>'BMCUTIL.CMN_BMCUTIL'</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I</td>
<td>BMC_BMCSYNC</td>
<td>'BMCUTIL.CMN_BMCSYNC'</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>BMC_BMCHIST</td>
<td>'BMCUTIL.CMN_BMCCHIST'</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>BMC_BMCXCOPY</td>
<td>'BMCUTIL.CMN_BMCXCOPY'</td>
</tr>
</tbody>
</table>
Figure 26   SYSPRINT for example 3 (part 3 of 5)
Example 3: Unloading multiple tables to a single data set

BMC51674I UNLOAD STATISTICS:  40 RECORDS WRITTEN TO DDNAME 'SYSRED'

BMC51675I UNLOAD STATISTICS:  0 RECORDS DISCARDED DUE TO ERRORS

BMC50044I UNLOAD PHASE COMPLETE.  ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPLE3.SYSCNTL,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)

BMC51801I DB2 CREATE TABLE and LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'

BMC51805I CREATE TABLE

BMC51807I BMC.ALL_EMPLS

BMC51807I ( ALL_EMPLS_LNAME VARCHAR(15)  NOT NULL
BMC51807I , ALL_EMPLS_FNAME VARCHAR(10)  NOT NULL
BMC51807I , ALL_EMPLS_SALCDE CHARACTER(1)  NOT NULL
BMC51808I , ALL_EMPLS_SALARY DECIMAL(9 , 2)
BMC51807I , ALL_EMPLS_DEPTNO CHARACTER(4)  NOT NULL
BMC51806I , ALL_EMPLS_HIREDT DATE
BMC51807I , ALL_EMPLS_CRDATE TIMESTAMP(6)  NOT NULL WITH DEFAULT
BMC51807I , ALL_EMPLS_STATUS CHARACTER(8)  NOT NULL

BMC51809I)

BMC51809I;

BMC51805I CREATE TABLE

BMC51809I BMC.ALL_EMPLS

BMC51807I ( ALL_EMPLS_LNAME VARCHAR(15)  NOT NULL
BMC51807I , ALL_EMPLS_FNAME VARCHAR(10)  NOT NULL
BMC51807I , ALL_EMPLS_SALCDE CHARACTER(1)  NOT NULL
BMC51808I , ALL_EMPLS_SALARY DECIMAL(9 , 2)
BMC51807I , ALL_EMPLS_DEPTNO CHARACTER(4)  NOT NULL
BMC51806I , ALL_EMPLS_HIREDT DATE
BMC51807I , ALL_EMPLS_CRDATE TIMESTAMP(6)  NOT NULL WITH DEFAULT
BMC51807I , ALL_EMPLS_STATUS CHARACTER(8)  NOT NULL

BMC51809I)

BMC51809I;

BMC51805I CREATE TABLE

BMC51809I BMC.ALL_EMPLS

BMC51807I ( ALL_EMPLS_LNAME VARCHAR(15)  NOT NULL
BMC51807I , ALL_EMPLS_FNAME VARCHAR(10)  NOT NULL
BMC51807I , ALL_EMPLS_SALCDE CHARACTER(1)  NOT NULL
BMC51808I , ALL_EMPLS_SALARY DECIMAL(9 , 2)
BMC51807I , ALL_EMPLS_DEPTNO CHARACTER(4)  NOT NULL
BMC51806I , ALL_EMPLS_HIREDT DATE
BMC51807I , ALL_EMPLS_CRDATE TIMESTAMP(6)  NOT NULL WITH DEFAULT
BMC51807I , ALL_EMPLS_STATUS CHARACTER(8)  NOT NULL

BMC51809I)

BMC51809I;

BMC51810I LOAD DATA INDDN SYSREC

BMC51940I EBCDIC CCSID(37,65534,65534)

BMC51811I INTO TABLE

BMC51809I BMC.ALL_EMPLS

BMC51813I (ALL_EMPLS_LNAME POSITION(1:*+) VARCHAR
BMC51813I ,ALL_EMPLS_FNAME POSITION(*:*) VARCHAR
BMC51813I ,ALL_EMPLS_SALCDE POSITION(*:*+0) CHAR  (1)
BMC51813I ,ALL_EMPLS_SALARY POSITION(*:*+4) DECIMAL
BMC51814I NULLIF BMC_NULL1=X'6F'
BMC51819I , BMC_NULL1 POSITION(*) CHAR(1)
BMC51815I ,ALL_EMPLS_DEPTNO POSITION(*:*+3) CHAR  (4)
BMC51815I ,ALL_EMPLS_HIREDT POSITION(*:*+9) DATE EXTERNAL(10)
BMC51814I NULLIF BMC_NULL2=X'6F'
BMC51819I , BMC_NULL2 POSITION(*) CHAR(1)
BMC51815I ,ALL_EMPLS_CRDATE POSITION(*:*+25) TIMESTAMP EXTERNAL(26)
BMC51815I ,ALL_EMPLS_STATUS POSITION(*:*+7) CHAR  (8)

BMC51809I)

BMC51810I LOAD DATA INDDN SYSREC

BMC51940I EBCDIC CCSID(37,65534,65534)

BMC51811I INTO TABLE

BMC51809I BMC.ALL_EMPLS

BMC51813I (ALL_EMPLS_LNAME POSITION(1:*+) VARCHAR
BMC51813I ,ALL_EMPLS_FNAME POSITION(*:*) VARCHAR
BMC51813I ,ALL_EMPLS_SALCDE POSITION(*:*+0) CHAR  (1)
BMC51813I ,ALL_EMPLS_SALARY POSITION(*:*+4) DECIMAL
BMC51814I NULLIF BMC_NULL1=X'6F'
BMC51819I , BMC_NULL1 POSITION(*) CHAR(1)
BMC51815I ,ALL_EMPLS_DEPTNO POSITION(*:*+3) CHAR  (4)
BMC51815I ,ALL_EMPLS_HIREDT POSITION(*:*+9) DATE EXTERNAL(10)
BMC51814I NULLIF BMC_NULL2=X'6F'
BMC51819I , BMC_NULL2 POSITION(*) CHAR(1)
BMC51815I ,ALL_EMPLS_CRDATE POSITION(*:*+25) TIMESTAMP EXTERNAL(26)
BMC51815I ,ALL_EMPLS_STATUS POSITION(*:*+7) CHAR  (8)
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 50041I 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.

Table 67 describes key command options that this job uses.

<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</td>
<td>invokes XBM or SUF to allow concurrent read and write access during the unload</td>
</tr>
<tr>
<td>CONSISTENT YES</td>
<td></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>
</tbody>
</table>
Example 4: Unloading while allowing read/write access to the table space

Table 67 Command options for example 4 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

Figure 27 shows the JCL for example 4.

Figure 27 JCL for example 4

```acl
// JOB
// UNLOAD04 EXEC PGM=ADUUMAIN,REGION=0M,
// PARM=(DEHJ,'ADUXMO4','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 XBMD ID XBMB
* SELECT EMPLOYEE_LNAME, HIRE_DATE, DEPT_NO, SALARY_CODE
  FROM BMC.EMPLS
  WHERE SALARY_CODE = 'H'
* SELECT EMPLOYEE_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=* 
```

Figure 28 shows the SYSPRINT for example 4.

Figure 28 SYSPRINT for example 4 (part 1 of 4)

```
***** BMC UNLOAD PLUS FOR DB2 VII.R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC5000I UTILITY EXECUTION STARTING 10/28/2013 13:02:42 ...
BMC50002I UTILITY ID = 'ADUXMO4'. DB2 SUBSYSTEM ID = 'DEHJ'. OPTION MODULE = 'ADUXOPTS'.
BMC50471I z/OS 2.1.0,PID=HBB7790,DFSMF FOR 2.05=2.1.0,DB2=11.1.0
BMC50471I REGION=OM,BELOW 16M=9844K,ABOVE 16M=1412360K,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
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT:BPJ0641 BPJ0667 BPJ0671 BPJ0674 BPJ0675 BPJ0682 BPJ0686 BPJ0689 BPJ0697
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAINT:BPJ0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I MAINT:BPJ0549 BPJ0534 BPJ05674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I MAINT:BPE0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416
BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FORCE_RPT=NO SMAX=16
BMC50471I CENTURY=(1950,2049) FORCE=NONE SMCORE=(OK,OK)
```
Example 4: Unloading while allowing read/write access to the table space

Figure 28  SYSPRINT for example 4 (part 2 of 4)
Example 4: Unloading while allowing read/write access to the table space

Figure 28  SYSPRINT for example 4 (part 3 of 4)

```
BMC50102I *
BMC50102I UNLOAD LIMIT 10  INTERVAL 2
BMC50102I *  SHRLEVEL CHANGE CONSISTENT YES XBMID XMB
BMC50102I *
BMC50102I SELECT EMPL_LNAME, HIRE_DATE, DEPT_NO, SALARY_CODE
BMC50102I FROM BMC.EMPLS
BMC50102I WHERE SALARY_CODE = 'H'
BMC50102I *
BMC50102I SELECT EMPL_LNAME,EMPL_HIRE_DATE,EMPL_DEPT_NO,EMPL_SALARY_CODE
BMC50102I FROM BMC.RET_EMPLS
BMC50102I WHERE EMPL_SALARY_CODE = 'S'
BMC50102I *
BMC51644I DIRECT YES IN EFFECT
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 1 IS 8 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 2 IS 5 FROM DB2STATS
BMC51688I ESTIMATED ROWS FOR TABLE BMC.EMPLS IS 20 FROM DB2STATS
BMC51688I ESTIMATED ROWS FOR TABLE BMC.RET_EMPLS IS 6240 FROM TABLESPACE
BMC51688I ESTIMATED ROWS FOR TABLE BMC.HRLY_EMPLS IS 6240 FROM TABLESPACE
BMC51688I ESTIMATED ROWS FOR TABLE BMC.SAL_EMPLS IS 6240 FROM TABLESPACE
BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 40
BMC51689I ESTIMATED ROWS FOR SELECT 2 IS 10
BMC50004I UTILINIT PHASE COMPLETE.  ELAPSED TIME = 00:00:00
BMC51639I FOR DDNAME 'SYSREC01' DSN=ADU.EXAMPLE4.SYSREC01,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=37)
BMC51639I FOR DDNAME 'SYSREC02' DSN=ADU.EXAMPLE4.SYSREC02,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=37)
BMC50894I PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 13:02:42
BMC50650I ESTABLISHING A POINT OF CONSISTENCY FOR:
BMC50651I SPACE BMCEXDB.BMCEXATS PART(0) PSID(0002)
BMC50651I SPACE BMCEXDB.BMCEXATS PART(0) PSID(0007)
BMC50895I PAGE EXTERNALIZATION PROCESS COMPLETE.  ELAPSED TIME = 00:00:00
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM
BMC50476I DDNAME = SYSREC01, I/OS = 3, I/O WAITS = 3, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC02, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51701I MAX TASKS = 4, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50477I 4: PARTITION = 4, ROWS/KEYS = 2, I/O WAITS = 2 , DDNAME = SYS00006
BMC50478I 4: RDB LOCK WAITS = 0
BMC50477I 3: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 2 , DDNAME = SYS00005
BMC50478I 3: RDB LOCK WAITS = 0
BMC50477I 2: PARTITION = 2, ROWS/KEYS = 2 , DDNAME = SYS00004
BMC50478I 2: RDB LOCK WAITS = 0
BMC50477I 1: PARTITION = 1, ROWS/KEYS = 8, I/O WAITS = 2 , DDNAME = SYS00003
BMC50478I 1: RDB LOCK WAITS = 0
BMC51701I MAX TASKS = 1, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50477I 5: PARTITION = 0, ROWS/KEYS = 20, I/O WAITS = 2 , DDNAME = SYS00007
BMC50478I 5: RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC01, I/OS = 3, I/O WAITS = 3, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC02, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 1
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A001' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 1 ROWS PROCESSED FROM PARTITION 2
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A002' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 3
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A003' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 0 ROWS PROCESSED FROM PARTITION 4
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A004' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS'. 15 NOT SELECTED, 0 DISCARDED
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.BMCEXATS.I0001.A001' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 4 ROWS PROCESSED FROM TABLE 'BMC.RET_EMPLS'. 16 NOT SELECTED, 0 DISCARDED
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.BMCEXATS.I0001.A002' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DUMMY 'SYSRECO1'
BMC51674I UNLOAD STATISTICS: 4 RECORDS WRITTEN TO DDNAME 'SYSRECO2'
BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM
BMC50277I XBM STATISTICS: DSN='DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A001'. READS 4, HITS 0, WRITES 0, CACHE 0
BMC50277I XBM STATISTICS: DSN='DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A002'. READS 4, HITS 0, WRITES 0, CACHE 0
BMC50277I XBM STATISTICS: DSN='DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A003'. READS 4, HITS 0, WRITES 0, CACHE 0
```
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 68 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 Appendix C, “Generating control statements for DB2 or other software products.”

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 68  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>
<tr>
<td>CNTLCARDS SAS</td>
<td>generates SAS external file INPUT statements in the SYSCNTL data set</td>
</tr>
<tr>
<td>PART 4,1:3</td>
<td>designates the individual partitions and range of partitions that will be unloaded</td>
</tr>
<tr>
<td></td>
<td>While you can specify a mixture of individual partitions and ranges of partitions in any order, UNLOAD PLUS unloads all partitions in ascending order.</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>OBID</td>
<td>specifies the OBID that will be associated with the selected table</td>
</tr>
<tr>
<td>WHERE</td>
<td>qualifies the data to unload</td>
</tr>
</tbody>
</table>
Figure 29 shows the JCL for example 5.

**Figure 29  JCL for example 5**

```plaintext
// JOB
// UNLOAD05 EXEC PGM=ADUUMAIN,REGION=0M,
// 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.EXMPEOS.SYSREC01,
+ // DISP=(C, CATLG), SPACE=(TRK,(1,1))
+ //SYSREC02 DD DSN=ADU.EXMPEOS.SYSREC02,
+ // DISP=(C, CATLG), SPACE=(TRK,(1,1))
+ //SYSREC03 DD DSN=ADU.EXMPEOS.SYSREC03,
+ // DISP=(C, CATLG), SPACE=(TRK,(1,1))
+ //SYSREC04 DD DSN=ADU.EXMPEOS.SYSREC04,
+ // DISP=(C, CATLG), SPACE=(TRK,(1,1))
+ //SYSCTNL DD DSN=ADU.EXMPEOS.SYSCTNL,
+ // DISP=(C, CATLG), SPACE=(TRK,(1,1))
+ /UTPRINT DD SYSOUT=* 
+ /SYSUDUMP DD SYSOUT=* 
+ /SYSPRINT DD SYSOUT=* 
```

Figure 30 shows the SYSPRINT for example 5.

**Figure 30  SYSPRINT for example 5 (part 1 of 4)**

```plaintext
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC00001I  UTILITY EXECUTION STARTING 10/28/2013 13:03:04 ...
BMC50471I   DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I   MAINT: BPJ0661  BPJ0667  BPJ0671  BPJ0674  BPJ0675  BPJ0676  BPJ0682  BPJ0686  BPJ0689  BPJ0697
BMC50471I   BMC STATS API--V02.04.01
BMC50471I   MAINT: BPJ0691
BMC50471I   EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I   MAINT: BPE0401  BPE0403  BPE0405  BPE0407  BPE0410  BPE0412  BPE0416
BMC50471I   DB2STATS,NOLIMIT
BMC50471I   DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I   MAINT: BPJ0661  BPJ0667  BPJ0671  BPJ0674  BPJ0675  BPJ0676  BPJ0682  BPJ0686  BPJ0689  BPJ0697
BMC50471I   BMC STATS API--V11.01.00
BMC50471I   MAINT: BPJ0691
BMC50471I   EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I   MAINT: BPE0401  BPE0403  BPE0405  BPE0407  BPE0410  BPE0412  BPE0416
BMC50471I   DB2STATS,NOLIMIT
BMC50471I   DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I   MAINT: BPJ0661  BPJ0667  BPJ0671  BPJ0674  BPJ0675  BPJ0676  BPJ0682  BPJ0686  BPJ0689  BPJ0697
BMC50471I   BMC STATS API--V11.01.00
BMC50471I   MAINT: BPJ0691
BMC50471I   EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I   MAINT: BPE0401  BPE0403  BPE0405  BPE0407  BPE0410  BPE0412  BPE0416
```

Chapter 5  Examples of UNLOAD PLUS jobs  335
Example 5: Unloading data from a DSN1COPY

Figure 30  SYSPRINT for example 5 (part 2 of 4)

<table>
<thead>
<tr>
<th>BMC0471I</th>
<th>CONSTRULES=BMC</th>
<th>LDDCRM=YES</th>
<th>TASKMAX=200%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC0471I</td>
<td>CURRENTDEGREE=None</td>
<td>MAXP=5</td>
<td>UBUFFS=25</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>DELFILES=YES</td>
<td>MSGLEVEL=1</td>
<td>UNLOADMAX=200%</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>DRNDELAY=1</td>
<td>NULLCHAR='6F'</td>
<td>UNLOADDN_ACTIVE=(YES,NO)</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>DRNWAIT=None</td>
<td>NULTYPE=TI</td>
<td>UNLOADDN=(SYSREC,SYSRED)</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>DRNWAIT=None</td>
<td>ODPNB2ID=YES</td>
<td>USELRECL=ND</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>DPLOCS=DRNFAL</td>
<td>PLAN=ADUQA</td>
<td>UXSTATE=SUP</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>EXCLDUMP=(X37,X22,X06)</td>
<td>RECMP=AUTO</td>
<td>WORKUNIT=SYSALLDA</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>FILERREFM=SYSREF</td>
<td>ROWSETSZ=100</td>
<td>ZIIP=ENABLED</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>FILL=NO</td>
<td>SDUMP=YES</td>
<td>ZONEDDECOVP=(C,D)</td>
</tr>
<tr>
<td>BMC0471I</td>
<td>FORCE_AT=(START,3)</td>
<td>SHRLEVEL=REFERENCE</td>
<td></td>
</tr>
<tr>
<td>BMC0471I</td>
<td>TAPES=NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0470I</td>
<td>OUTPUT = SYSREC</td>
<td>SYSRED</td>
<td>SYSREF</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>UNIT = SYSALLDA</td>
<td>SYSALLDA</td>
<td>SYSALLDA</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>VOLCNT = 25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>GDGLIMIT = 5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>DDGEMPTY = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>DDGCRT = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>STORCLAS = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>DATACLAS = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>MGMTCLAS = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>UNITCNT = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>SPACE = CYL</td>
<td>CYL</td>
<td>CYL</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>FILESZPCT = 100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>NBRSECD = AUTO</td>
<td>AUTO</td>
<td>AUTO</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>DISKRETN = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>DISKEXPD = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>RETPD = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>EXPOT = 99000</td>
<td>99000</td>
<td>99000</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>TRCH = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC0470I</td>
<td>DSNCMP = NONE</td>
<td>NONE</td>
<td>PDS</td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC VOLUMES=None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC VOLUMES=None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC VOLUMES=None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC VOLUMES=None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC DSNNAME=BUSERID&amp;type.s&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC DSNNAME=BUSERID&amp;type.s&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC0483I</td>
<td>SYSREC DSNNAME=BUSERID&amp;type.s&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BMC0471I  DB2 DSNHDECP MODULE SETTINGS:
| BMC0471I | VERSION = 1110 |
| BMC0471I | SUBSYSTEM DEFAULT = DEHJ |
| BMC0471I | CHARACTER SET = ALPHANUM |
| BMC0471I | DATE FORMAT = USA |
| BMC0471I | TIME FORMAT = USA |
| BMC0471I | LOCAL DATE LENGTH = 0 |
| BMC0471I | LOCAL TIME LENGTH = 0 |
| BMC0471I | DECIMAL POINT = PERIOD |
| BMC0471I | DECIMAL ARITHMETIC = 15 |
| BMC0471I | DELIMITER = DEFAULT |
| BMC0471I | SQL DELIMITER = DEFAULT |
| BMC0471I | ENCODING SCHEME = EBCDIC |
| BMC0471I | APPL. ENCODING SCHEME = EBCDIC |
| BMC0471I | MIXED = NO |
| BMC0471I | EBCDIC CCSID = (37,65534,65534) |
| BMC0471I | ASCII CCSID = (819,65534,65534) |
| BMC0471I | UNICODE CCSID = (367,1208,1200) |
| BMC0471I | IMPLICIT TIME ZONE = CURRENT (-05:00) |

BMC0528I  DB2 MODE = NFM
| BMC0471I | BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL' |
| BMC0471I | BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC' |
| BMC0471I | BMC_BMCXCPY = 'BMCUTIL.CMN_BMCXCPY' |
| BMC0471I | BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY' |

BMC0102I *
BMC0102I UNLOAD INFILE DSN1CPY
BMC0102I CNTDCAIDS SAS

UNLOAD PLUS for DB2 Reference Manual
Example 5: Unloading data from a DSN1COPY

Chapter 5 Examples of UNLOAD PLUS jobs 337

BMC51644 I DIRECT YES IN EFFECT
BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC51639I FOR DDNAME 'SYSREC01' DSN=ADU.EXAMPLE5.SYSREC01,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=111)
BMC51639I FOR DDNAME 'SYSREC02' DSN=ADU.EXAMPLE5.SYSREC02,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=111)
BMC51639I FOR DDNAME 'SYSREC03' DSN=ADU.EXAMPLE5.SYSREC03,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=111)
BMC51639I FOR DDNAME 'SYSREC04' DSN=ADU.EXAMPLE5.SYSREC04,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=111)
BMC50476I DDNAME = DSN1CPY2, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY1, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY3, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY4, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51701I MAX TASKS = 4, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50477I 2: PARTITION = 2, ROWS/KEYS = 5, I/O WAITS = 0 , DDNAME = DSN1CPY2
BMC50477I 2: RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY4, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50477I 1: PARTITION = 1, ROWS/KEYS = 0, I/O WAITS = 2 , DDNAME = DSN1CPY1
BMC51701I MAX TASKS = 4, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50476I DDNAME = DSN1CPY1, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY3, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY4, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50477I 3: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 0 , DDNAME = DSN1CPY3
BMC50478I 3: RDB LOCK WAITS = 0
BMC50476I DDNAME = DSN1CPY4, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51701I MAX TASKS = 4, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50476I DDNAME = SYSREC01, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC02, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC03, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC04, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51671I UNLOAD STATISTICS: 8 ROWS PROCESSED FROM PARTITION 1
BMC51666I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P1' READ 3 PAGES
BMC51667I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 2
BMC51666I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P2' READ 3 PAGES
BMC51667I UNLOAD STATISTICS: 4 ROWS PROCESSED FROM PARTITION 3
BMC51666I UNLOAD STATISTICS: 4 ROWS PROCESSED FROM PARTITION 4
BMC51666I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P3' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51671I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P4' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 19 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 1 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 8 RECORDS WRITTEN TO DDNAME 'SYSREC01'
BMC51671I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYSREC02'
BMC51671I UNLOAD STATISTICS: 4 RECORDS WRITTEN TO DDNAME 'SYSREC03'
BMC51671I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XMB
BMC50476I DDNAME = SYSREC01, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC02, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC03, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYSREC04, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51672I UNLOAD STATISTICS: 8 ROWS PROCESSED FROM PARTITION 1
BMC51666I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P1' READ 3 PAGES
BMC51667I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 2
BMC51666I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P2' READ 3 PAGES
BMC51667I UNLOAD STATISTICS: 4 ROWS PROCESSED FROM PARTITION 3
BMC51666I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P3' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51671I UNLOADING OF DATASET 'ADU.QA.DSN1COPY.ADUEX05.P4' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 19 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 1 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 8 RECORDS WRITTEN TO DDNAME 'SYSREC01'
BMC51671I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYSREC02'
BMC51671I UNLOAD STATISTICS: 4 RECORDS WRITTEN TO DDNAME 'SYSREC03'
BMC51671I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XMB
BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC51639I FOR DDNAME 'SYSREC01' DSN=ADU.EXAMPLES.SYSREC01,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)
BMC51801I SAS STANDARD EXTERNAL FILE 'INFILE' STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'
BMC51836I OPTIONS INVALIDDATA='.';
BMC51821I DATA BMC;
BMC51822I INFILE SYSREC;
BMC51823I INPUT
BMC51832I NULL1 $CHARZB1.
BMC51831I NULL2 $CHARZB1.
BMC51835I IF VARLEN1 ≠ 0 THEN INPUT
BMC51825I IF VARLEN1 ≠ 0 THEN INPUT
BMC51824I PART_401K $CHAR1.   /* NAME TOO LARGE */
BMC51828I        EMPL_ID ?? PD3.0
BMC51824I        SSN $CHAR9.
BMC51824I        HIRE_DATE $CHAR10.   /* NAME TOO LARGE */
BMC51828I        EMPL_LNAME $VARYING15. VARLEN1   /* NAME TOO LARGE */
BMC51822I        DEPT_NO $CHAR4.
BMC51824I        JOB_CODE IB2.
BMC51832I        NULL3 $CHARZB1.
BMC51832I        SALARY_HRLY ?? PD5.2   /* NAME TOO LARGE */
BMC51824I        SALARY_CODE $CHAR1.   /* NAME TOO LARGE */
BMC51832I        PART_401K $CHAR1.   /* NAME TOO LARGE */
BMC51832I        VARLEN1 IB2. @;  /* LENGTH OF FIELD EMPL_LNAME */
BMC51837I          IF VARLEN1 ¬= 0 THEN INPUT
BMC51838I        INPUT   /* RESUME INPUT */

Figure 30  SYSPRINT for example 5 (part 3 of 4)
Example 6: Unloading a full image copy

This example unloads from a full image copy data set that is registered in the DB2 catalog. Table 69 describes the key command options for this job.

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 357. Although example 11 illustrates unloading an incremental image copy, the negative integer in the command option applies to either type of image copy.

### Table 69  Command options for example 6 (part 1 of 2)

<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>TSFMFT</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</td>
</tr>
<tr>
<td></td>
<td>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>
</tbody>
</table>
Example 6: Unloading a full image copy

Chapter 5 Examples of UNLOAD PLUS jobs 339

Table 69 Command options for example 6 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</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>
<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>
</tbody>
</table>

Note: To order by data-sorting key, use the ORDER YES statement as shown in example 2.

Figure 31 shows the JCL for example 6.

Figure 31 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 *
* UNLOAD INFILE IMAGECOPY
TSFMT('MM.DD.YY/HH:MM:SS.NN')
DATEFMT('MM.DD.YY')
PART 3
SELECT *
FROM BMC.EMPLS WHERE SSN > '200000000'
ORDER BY HIRE_DATE DESC, EMPL_LNAME
*
//SYSREC DD DSN=ADU.EXAMPLE6.SYSREC, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSCTRL DD DSN=ADU.EXAMPLE6.SYSCNTL, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//UTPRINT DD SYSOUT=* //SYSPRINT DD SYSOUT=* //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))
```

Figure 32 shows the SYSPRINT for example 6.

Figure 32 SYSPRINT for example 6 (part 1 of 4)

```
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(B) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50471I UTILITY EXECUTION STARTING 10/28/2013 13:03:59 ...
BMC50471I UTILITY ID = 'ADUXM06'. DB2 SUBSYSTEM ID = 'DEHJ'. OPTION MODULE = 'ADU$OPTS'.
BMC50471I DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I SOLUTION COMMON CODE--V11.01.00
```

Table 69 Command options for example 6 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</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>
<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>
</tbody>
</table>

Note: To order by data-sorting key, use the ORDER YES statement as shown in example 2.

Figure 31 shows the JCL for example 6.

Figure 31 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 *
* UNLOAD INFILE IMAGECOPY
TSFMT('MM.DD.YY/HH:MM:SS.NN')
DATEFMT('MM.DD.YY')
PART 3
SELECT *
FROM BMC.EMPLS WHERE SSN > '200000000'
ORDER BY HIRE_DATE DESC, EMPL_LNAME
*
//SYSREC DD DSN=ADU.EXAMPLE6.SYSREC, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSCTRL DD DSN=ADU.EXAMPLE6.SYSCNTL, // DISP=(,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//UTPRINT DD SYSOUT=* //SYSPRINT DD SYSOUT=* //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))
```

Figure 32 shows the SYSPRINT for example 6.

Figure 32 SYSPRINT for example 6 (part 1 of 4)

```
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(B) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50471I UTILITY EXECUTION STARTING 10/28/2013 13:03:59 ...
BMC50471I UTILITY ID = 'ADUXM06'. DB2 SUBSYSTEM ID = 'DEHJ'. OPTION MODULE = 'ADU$OPTS'.
BMC50471I DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I SOLUTION COMMON CODE--V11.01.00
```
Example 6: Unloading a full image copy

Figure 32  SYSPRINT for example 6 (part 2 of 4)
Example 6: Unloading a full image copy

Chapter 5 Examples of UNLOAD PLUS jobs

BMC50471I ASCII CCSID = (819,65534,65534)
BMC50471I UNICODE CCSID = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)

BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST = 'BMCUTIL.CMN_BMCHIST'
BMC50471I BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY'

BMC50102I *
BMC50102I  UNLOAD INFILE IMAGECOPY
BMC50102I         TSFMT(‘MM.DD.YY/HH:MM:SS.NN’)
BMC50102I         DATEFMT(‘MM.DD.YY’)
BMC50102I         PART 3
BMC50102I         SELECT *
BMC50102I FROM BMC.EMPLS WHERE SSN > ‘200000000’
BMC50102I ORDER BY HIRE_DATE DESC, EMPL_LNAME
BMC50102I *

BMC51654I DIRECT YES IN EFFECT
BMC51655I LOCATING FULL IMAGE COPIES
BMC51624I FOR PARTITION ALL, IMAGE COPY ‘ADU.QA.ICOPY.FULL1.BMCEXATS’(2013-10-28-13.03.43.921986) WILL BE USED

BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC50041I 0: ZIIP  ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50476I DDNAME = SYS00002, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51701I MAX TASKS = 1, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 4, MAX OPEN PARTITIONS PER TASK = 1
BMC51671I UNLOAD STATISTICS: 4 ROWS PROCESSED FROM PARTITION 3

BMC51671I FOR DDNAME 'SYSREC' DSN=ADU.EXAMPLE6.SYSREC, DCB=(RECFM=VB, BLKSIZE=27993, LRECL=103)

BMC51604I 1: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 0, DDNAME = SYS00002
BMC51671I UNLOAD STATISTICS: 4 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 1 NOT SELECTED, 0 DISCARDED

BMC51604I 2: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 0
BMC51671I UNLOAD STATISTICS: 4 RECORDS WRITTEN TO DDNAME 'SYSREC'
BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS

BMC50004I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50041I 0: ZIIP  ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC50041I 0: ZIIP  ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPLE6.SYSCNTL, DCB=(RECFM=FB, BLKSIZE=320, LRECL=80)
BMC51810I LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'

BMC51809I BMC.EMPLS
BMC51811I INTO TABLE
BMC51809I BMC.EMPLS
BMC51813I (EMPL_ID POSITION(1:3) DECIMAL,
BMC51815I ,SSN POSITION(4:12) CHAR (9),
BMC51815I ,HIRE_DATE POSITION(13:20) DATE EXTERNAL(8),
BMC51814I ,NULLIF BMC_NULL1=X’6F’,
BMC51819I ,DEPT_NO POSITION(22:25) CHAR (4),
BMC51813I ,JOB_CODE POSITION(26:27) SMALLINT,
BMC51814I ,NULLIF BMC_NULL2=X’6F’,
BMC51819I ,SALARY_CODE POSITION(29:29) CHAR (1),
BMC51814I ,NULLIF BMC_NULL3=X’6F’,
BMC51819I ,PART_401K POSITION(36:36) CHAR (1),
BMC51814I ,NULLIF BMC_NULL4=X’6F’,
BMC51819I ,EMPL_LNAME POSITION(*:*) VARCHAR,
BMC51819I ,EMPL_FNAME POSITION(*:*) VARCHAR,
BMC51819I ,EMPL_MNAME POSITION(*:*) VARCHAR,
BMC51814I ,NULLIF BMC_NULL5=X’6F’,
BMC51819I ,nullif BMC_NULL5=X’6F’)
BMC51819I ,nullif BMC_NULL5=X’6F’)

Figure 32  SYSPRINT for example 6 (part 3 of 4)
Example 7: Using FILTERPART

This example unloads a partitioned table space with four partitions into two dynamically allocated output data sets. Table 70 describes the key command options for this job.

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.

Table 70 Command options for example 7

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SYSSREC 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>

Figure 33 shows the JCL for example 7.
Example 7: Using FILTERPART

Figure 33 JCL for example 7 (part 2 of 2)

```
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
*
UNLOAD
OUTPUT SYSREC DSNAME 'ADU.EXAMPLE7.SYSREC&PART'
*
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=* 
```

Figure 34 SYSPRINT for example 7 (part 1 of 4)

```
**** BMC UNLOAD PLUS FOR DB2 VI11R1.00 ****
BMC50001I UTILITY EXECUTION STARTING 10/28/2013 13:04:13 ...
BMC50002I UTILITY ID = 'ADUXM07'. 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=16M,BELONG 16M=8844K,ABOVE 16M=1413312K,IEFUSI=NO,CPU5=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
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT: BPJ0661 BPJ0667 BPJ0671 BPJ0674 BPJ0675 BPJ0682 BPJ0686 BPJ0689 BPJ0697
BMC50471I BMCSORT ENGINE -- V02.04.01
BMC50471I MAINT: BPJ0691
BMC50471I MAINT: BPSUS49 BPUS534 BPU5674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I MAINT: BPD401 BPD403 BPD405 BPD407 BPD410 BPD412 BPD416
BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FORCE_RPT=NO SMAX=16
BMC50471I CENTURY=(1950,2049) FORCE=NONE SMCORE=(OK,OK)
BMC50471I CHANGE_CONSISTENT=NO HISTORY=YES SORTNUM=12
BMC50471I CHANGE_QUESCE=NO IBUFFS=25 SQLDELAY=3
BMC50471I CMAX=16 INLINE=NO SQLRETRY=100
BMC50471I CMAX=16 LOADDEC=NO TAPEDISP=DELETE
BMC50471I CONSTRUES=NO LOCKROW=YES TASKMAX=200
BMC50471I CURRENTDEGREE=NONE MAX=5 UBUFFS=25
BMC50471I DELFILES=NO MSGLEVEL=1 UNLOADMAX=2000
BMC50471I DRNDDELAY=1 NULLCHAR='X'6F' UNLOADON_ACTIVE=(YES,NO)
BMC50471I DRNKTRY=55 NULLTYPE=TI UNLOADON=(SYSREC,SYSRD)
BMC50471I DRNWAIT=NONE OPMNDBD=10 USERECL=NO
BMC50471I DSPLOCKS=DRNFAIL PLAN=ADUQA UXSTATE=SUP
BMC50471I EXCLUDUMP=(X37,X22,X06) RECNUM=AUTO WORKUNIT=SYSALDA
BMC50471I FILENEXT=SYSREF ROWSETS=100 ZIP=ENABLED
BMC50471I FILE=NO SOUNM=NO ZONEDDECOVP=(C,D)
BMC50471I FORCE_AT=(START,3) SHLEVEL=REFERENCE
BMC50471I TAPES=NONE
BMC50470I OUTPUT = SYSREC SYSRED SYSRD
BMC50470I UNIT = SYSALLDA SYSALLDA SYSREF
```

Figure 34 shows the SYSPRINT for example 7.
Example 7: Using FILTERPART

BMC50470I VOLLNT = 25
BMC50470I GDLIMIT = 5
BMC50470I GDLVOLM = NO
BMC50470I GDLVOLM = NO
BMC50470I STORCLS = NONE
BMC50470I DATACLAS = NONE
BMC50470I UNITCNT = 0
BMC50470I SPACE = CYL
BMC50470I PCTPRIM = AUTO
BMC50470I MAXPRIM = 0
BMC50470I MAXSECD = 0
BMC50470I FILESZPCT = 100
BMC50470I NBRSECD = AUTO
BMC50470I DISKRETN = NONE
BMC50470I DISKEXPD = NONE
BMC50470I RECDP = 0
BMC50470I NBRSECD = AUTO
BMC50470I EXPDP = 0
BMC50470I EXPDT = 99000
BMC50470I TRTCH = NONE
BMC50470I DSNTYPE = NONE
BMC50483I SYSREC VOLUMES=NONE
BMC50483I SYSRED VOLUMES=NONE
BMC50483I SYSREF VOLUMES=NONE
BMC50483I SYSREC DSNAME=&USERID.&TYPE.S&SELNUM
BMC50483I SYSRED DSNAME=&USERID.&TYPE.S&SELNUM
BMC50483I SYSREF DSNAME=&USERID.BMC

BMC50471I DB2 DSNHDECP MODULE SETTINGs:
BMC50471I VERSION = 1110
BMC50471I SUBSYSTEM DEFAULT = DEHJ
BMC50471I CHARACTER SET = ALPHANUM
BMC50471I DATE FORMAT = USA
BMC50471I TIME FORMAT = USA
BMC50471I LOCAL DATE LENGTH = 0
BMC50471I LOCAL TIME LENGTH = 0
BMC50471I DECIMAL POINT = PERIOD
BMC50471I DECIMAL ARITHMETIC = 15
BMC50471I DELIMITER = DEFAULT
BMC50471I SQL DELIMITER = 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 UNICODE CCSID = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)

BMC51654I DIRECT YES IN EFFECT

Figure 34  SYSPRINT for example 7 (part 2 of 4)
Example 7: Using FILTERPART

Figure 34  SYSPRINT for example 7 (part 3 of 4)

### Chapter 5 Examples of UNLOAD PLUS jobs
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.

Table 71 describes the key command options and DD statements for this job.

Table 71 Command options and DD statements for example 8 (part 1 of 2)

<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</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>
</tbody>
</table>
Example 8: Using multiple SELECT statements

Table 71 Command options and DD statements for example 8 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options and DD statements used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>//CNTL2</td>
<td>receives DB2 LOAD control statements for SELECT statements 2, 4, and 6</td>
</tr>
<tr>
<td>//CNTL4</td>
<td></td>
</tr>
<tr>
<td>//CNTL6</td>
<td></td>
</tr>
<tr>
<td>//DATA</td>
<td>receives data unloaded by SELECT statements 1, 3, and 5. The UNLOADDDN option specifies this ddname prefix.</td>
</tr>
<tr>
<td>//DATA2</td>
<td></td>
</tr>
<tr>
<td>//DATA4</td>
<td></td>
</tr>
<tr>
<td>//DATA6</td>
<td></td>
</tr>
</tbody>
</table>

Figure 35 shows the JCL for example 8.

Figure 35 JCL for example 8 (part 1 of 2)

```sql
// JOB
//UNLOAD08 EXEC PGM=ADUUMAIN,REGION=OM,
// PARM=(DEHJ,'ADUXM08','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'
UNLOADDDN 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 *
```
Example 8: Using multiple SELECT statements

Figure 35  JCL for example 8 (part 2 of 2)

```sql
FROM BMC.HRLY_EMPLS
*
//CNTL DO DSN=ADU.EXAMPLE8.CNTL,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//CNTL2 DO DSN=ADU.EXAMPLE8.CNTL2,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//CNTL4 DO DSN=ADU.EXAMPLE8.CNTL4,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//CNTL6 DO DSN=ADU.EXAMPLE8.CNTL6,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA DO DSN=ADU.EXAMPLE8.DATA,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA2 DO DSN=ADU.EXAMPLE8.DATA2,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA4 DO DSN=ADU.EXAMPLE8.DATA4,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//DATA6 DO DSN=ADU.EXAMPLE8.DATA6,
// DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//UTPRINT DO SYSOUT=*  
//SYSPRINT DO SYSOUT=* 
```

Figure 36 shows the SYSPRINT for example 8.

Figure 36  SYSPRINT for example 8 (part 1 of 5)

```sql
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****  
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.  
BMC5000I UTILITY EXECUTION STARTING 10/28/2013 13:04:23 ...  
BMC50471I REGION=0M,BELOW 16M=8844K,ABOVE 16M=1413280K,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  
BMC50471I SOLUTION COMMON CODE--V11.01.00  
BMC50471I MAINT: BPJ0661 BPJ0667 BPJ0671 BPJ0674 BPJ0675 BPJ0682 BPJ0686 BPJ0689 BPJ0697  
BMC50471I BMCSORT ENGINE--V02.04.01  
BMC50471I MAINT: BPJ0691  
BMC50471I BMC STATS API--V11.01.00  
BMC50471I MAINT: BPJ0569  
BMC50471I BMC STATS API--V11.01.00  
BMC50471I MAINT: BPJ0549 BPJ05534 BPJ05674  
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00  
BMC50471I MAINT: BPE0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416  
BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FORCERPT=NO SMAX=16  
BMC50471I CENTURY=(1950,2049) FORCENONE SMCORE=(OK,OK)  
BMC50471I CHANGE_CONSISTENT=NO HISTORY=YES SORTNUM=32  
BMC50471I CHANGE_QUIESCE=NO BUFFFSX=25 SDELAY=3  
BMC50471I CMAX=16 INLINE=NO SOLRTRY=100  
BMC50471I CMRATIO=50 LOADADD=NO TADEDISP=DELETE  
BMC50471I CONNVRULES=BMC LOCKDOWN=YES TASKMAX=200%  
BMC50471I CURRENTBHEADER=NONE MAXF=5 UBUFFS=25  
BMC50471I EXCLDUMP=(X37,X22,X06) MSELVL=1 UNLOAD=200%  
BMC50471I DRNDelay=1 NULLCHAR=X'6F' UNLOAD_ACTIVE=(YES,NO)  
BMC50471I DRNRETRY=255 NULLTYPE=I UNLOADON=(SYSREC,SYSEX)  
BMC50471I DRNWAIT=NONE OPNORID=NO USEREC=NO  
BMC50471I DSNPLOCS=DRNFIL  PLANK=AUDOA UXSTATE=SUP  
BMC50471I EXCLUDUMP=(X37,X22,X06) RECFM=AUTO WUNKIND=SYSALLODA  
BMC50471I FILEFORM=SYSREC RDSIZE=100 ZIIP=ENABLED  
BMC50471I FILE=NO SDUMP=NO ZONEDECOVP=(C,D)  
BMC50471I FORKEAT=(START,3) SHLEVEL=REFERENCE  
BMC50471I TAPES=NONE  
BMC50471I OUTPUT = SYSREC SYSRED SYSEXREF  
BMC50471I UNIT = SYSALLODA SYSALLODA SYSALLODA  
BMC50471I VOLTNT = 25 25  
```
Example 8: Using multiple SELECT statements

Chapter 5 Examples of UNLOAD PLUS jobs

Figure 36 SYSPRINT for example 8 (part 2 of 5)

```sql
BMC50470I GDGLIMIT = 5
BMC50470I GDGEMPTY = NO
BMC50470I GDGSCRAT = NO
BMC50470I DATACLAS = NONE
BMC50470I MGMTCLAS = NONE
BMC50470I UNITCNT  = 0
BMC50470I SPACE    = CYL
BMC50470I PCTPRIM  = AUTO
BMC50470I MAXPRIM  = 0
BMC50470I MAXSECD  = 0
BMC50470I FILESZPCT = 100
BMC50470I NBRSECD = AUTO
BMC50470I DISKRETN = NONE
BMC50470I DISKEXPD = NONE
BMC50470I RETPD   = NONE
BMC50470I EXPDT   = 99000
BMC50470I TRTCH   = NONE
BMC50470I DSNTYPE = NONE
BMC50483I SYSREC     VOLUMES=NONE
BMC50483I SYSRED     VOLUMES=NONE
BMC50483I SYSREF     VOLUMES=NONE
BMC50483I SYSREC     DSNAME=&USERID.&TYPE.S&SELNUM
BMC50483I SYSRED     DSNAME=&USERID.&TYPE.S&SELNUM
BMC50483I SYSREF     DSNAME=&USERID.BMC

BMC50471I DB2 DSNHDECP MODULE SETTINGS:
BMC50471I VERSION                 = 1110
BMC50471I SUBSYSTEM DEFAULT       = DEHJ
BMC50471I CHARACTER SET           = ALPHANUM
BMC50471I DATE FORMAT             = USA
BMC50471I TIME FORMAT             = USA
BMC50471I LOCAL DATE LENGTH       = 0
BMC50471I LOCAL TIME LENGTH       = 0
BMC50471I DECIMAL POINT           = PERIOD
BMC50471I DECIMAL ARITHMETIC      = 15
BMC50471I DELIMITER               = DEFAULT
BMC50471I SQL DELIMITER           = 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 UNICODE CCSID           = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE      = CURRENT (-05:00)

BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL       ='BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC       ='BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST       ='BMCUTIL.CMN_BMCHIST'
BMC50471I BMC_BMCCOPY      ='BMCUTIL.CMN_BMCCOPY'

BMC50102I *
BMC50102I UNLOAD CNTLCARDS DB2 'ENUMROWS' 'REPLACE'
BMC50102I UNLOADON DATA
BMC50102I CNTLDN CNTL
BMC50102I SELECT EMP_LNAME, EMP_DEPT_NO, EMP_HIRE_DATE
BMC50102I INTO RECORDID 'S'
BMC50102I FROM BMC.SALEmpleado
BMC50102I SELECT *
BMC50102I FROM BMC.SALEmpleado
BMC50102I SELECT EMP_LNAME, EMP_DEPT_NO, EMP_HIRE_DATE
BMC50102I INTO RECORDID 'H'
BMC50102I FROM BMC.RETEmpleado
BMC50102I SELECT *
BMC50102I FROM BMC.RETEmpleado
BMC50102I SELECT EMP_LNAME, EMP_DEPT_NO, EMP_HIRE_DATE
BMC50102I INTO RECORDID 'H'
BMC50102I FROM BMC.HREmpleado
BMC50102I SELECT *
BMC50102I FROM BMC.HREmpleado
BMC50102I *
```
Example 8: Using multiple SELECT statements

Figure 36  SYSPRINT for example 8 (part 3 of 5)
### Example 8: Using multiple SELECT statements

#### Chapter 5 Examples of UNLOAD PLUS jobs

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC51819I</td>
<td>, BMC_NULLI POSITION(*) CHAR(1)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_DEPT_NO POSITION(*:+3) CHAR(4)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_HIRE_DATE POSITION(*:+9) DATE EXTERNAL(10)</td>
</tr>
<tr>
<td>BMC51814I</td>
<td>, NULLIF BMC_NULLI=X'6F'</td>
</tr>
<tr>
<td>BMC51819I</td>
<td>, BMC_NULLI2 POSITION(*) CHAR(1)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_REC_CREATED POSITION(*:+25) TIMESTAMP EXTERNAL(26)</td>
</tr>
</tbody>
</table>

**Figure 36** SYSPRINT for example 8 (part 4 of 5)**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50476I</td>
<td>DDNAME = CNTL2, I/O WAIT = 1, RDB LOCK WAIT = 0</td>
</tr>
<tr>
<td>BMC51639I</td>
<td>FOR DDNAME 'CNTL4' DSN=ADU.EXAMPLE8.CNTL4,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)</td>
</tr>
<tr>
<td>BMC51801I</td>
<td>CREATE TABLE AND LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'CNTL4'</td>
</tr>
<tr>
<td>BMC51805I</td>
<td>CREATE TABLE</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>BMC.RET_EMPLS</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_LNAME VARCHAR(15) NOT NULL</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_FNAME VARCHAR(10) NOT NULL</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_SALARY_CODE CHARACTER(1) NOT NULL</td>
</tr>
<tr>
<td>BMC51808I</td>
<td>, EMPL_HRLY_RATE DECIMAL(9,2)</td>
</tr>
<tr>
<td>BMC51806I</td>
<td>, EMPL_DEPT_NO CHARACTER(4) NOT NULL</td>
</tr>
<tr>
<td>BMC51806I</td>
<td>, EMPL_HIRE_DATE DATE</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_REC_CREATED TIMESTAMP(6) NOT NULL WITH DEFAULT</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>;</td>
</tr>
<tr>
<td>BMC51810I</td>
<td>LOAD DATA INDDN DATA4</td>
</tr>
<tr>
<td>BMC51800I</td>
<td>ENUMROWS (.20)</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>REPLACE</td>
</tr>
<tr>
<td>BMC51940I</td>
<td>EBCDIC CCSID(37,65534,65534)</td>
</tr>
<tr>
<td>BMC51811I</td>
<td>INTO TABLE</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>BMC.RET_EMPLS</td>
</tr>
<tr>
<td>BMC51813I</td>
<td>(EMPL_LNAME POSITION(1:*)) VARCHAR</td>
</tr>
<tr>
<td>BMC51813I</td>
<td>,EMPL_FNAME POSITION(<em>:</em>)) VARCHAR</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>,EMPL_SALARY_CODE POSITION(<em>:</em>+0) CHAR(1)</td>
</tr>
<tr>
<td>BMC51813I</td>
<td>,EMPL_HRLY_RATE POSITION(<em>:</em>+4) DECIMAL</td>
</tr>
<tr>
<td>BMC51814I</td>
<td>, NULLIF BMC_NULLI=X'6F'</td>
</tr>
<tr>
<td>BMC51819I</td>
<td>, BMC_NULLI2 POSITION(*) CHAR(1)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_DEPT_NO POSITION(*:+3) CHAR(4)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_HIRE_DATE POSITION(*:+9) DATE EXTERNAL(10)</td>
</tr>
<tr>
<td>BMC51814I</td>
<td>, NULLIF BMC_NULLI=X'6F'</td>
</tr>
<tr>
<td>BMC51819I</td>
<td>, BMC_NULLI2 POSITION(*) CHAR(1)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_REC_CREATED POSITION(*:+25) TIMESTAMP EXTERNAL(26)</td>
</tr>
</tbody>
</table>

**Figure 36** SYSPRINT for example 8 (part 4 of 5)**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC51805I</td>
<td>CREATE TABLE</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>BMC.HRLY_EMPLS</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_LNAME VARCHAR(15) NOT NULL</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_FNAME VARCHAR(10) NOT NULL</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_SALARY_CODE CHARACTER(1) NOT NULL</td>
</tr>
<tr>
<td>BMC51808I</td>
<td>, EMPL_HRLY_RATE DECIMAL(9,2)</td>
</tr>
<tr>
<td>BMC51806I</td>
<td>, EMPL_DEPT_NO CHARACTER(4) NOT NULL</td>
</tr>
<tr>
<td>BMC51806I</td>
<td>, EMPL_HIRE_DATE DATE</td>
</tr>
<tr>
<td>BMC51807I</td>
<td>, EMPL_REC_CREATED TIMESTAMP(6) NOT NULL WITH DEFAULT</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>;</td>
</tr>
<tr>
<td>BMC51810I</td>
<td>LOAD DATA INDDN DATA6</td>
</tr>
<tr>
<td>BMC51800I</td>
<td>ENUMROWS (.11)</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>REPLACE</td>
</tr>
<tr>
<td>BMC51940I</td>
<td>EBCDIC CCSID(37,65534,65534)</td>
</tr>
<tr>
<td>BMC51811I</td>
<td>INTO TABLE</td>
</tr>
<tr>
<td>BMC51809I</td>
<td>BMC.HRLY_EMPLS</td>
</tr>
<tr>
<td>BMC51813I</td>
<td>(EMPL_LNAME POSITION(1:*)) VARCHAR</td>
</tr>
<tr>
<td>BMC51813I</td>
<td>,EMPL_FNAME POSITION(<em>:</em>)) VARCHAR</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>,EMPL_SALARY_CODE POSITION(<em>:</em>+0) CHAR(1)</td>
</tr>
<tr>
<td>BMC51813I</td>
<td>,EMPL_HRLY_RATE POSITION(<em>:</em>+4) DECIMAL</td>
</tr>
<tr>
<td>BMC51814I</td>
<td>, NULLIF BMC_NULLI=X'6F'</td>
</tr>
<tr>
<td>BMC51819I</td>
<td>, BMC_NULLI2 POSITION(*) CHAR(1)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_DEPT_NO POSITION(*:+3) CHAR(4)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_HIRE_DATE POSITION(*:+9) DATE EXTERNAL(10)</td>
</tr>
<tr>
<td>BMC51814I</td>
<td>, NULLIF BMC_NULLI=X'6F'</td>
</tr>
<tr>
<td>BMC51819I</td>
<td>, BMC_NULLI2 POSITION(*) CHAR(1)</td>
</tr>
<tr>
<td>BMC51815I</td>
<td>, EMPL_REC_CREATED POSITION(*:+25) TIMESTAMP EXTERNAL(26)</td>
</tr>
</tbody>
</table>
Example 8: Using multiple SELECT statements

Figure 36  SYSPRINT for example 8 (part 5 of 5)

BMC51801I CREATE TABLE
BMC51809I BMC.SAL_EMPLS
BMC51807I (EMPL_LNAME VARCHAR(15) NOT NULL
BMC51807I , EMPL_DEPT_NO CHARACTER(4) NOT NULL
BMC51806I , EMPL_HIRE_DATE DATE
BMC51809I )
BMC51809I ;
BMC51805I CREATE TABLE
BMC51809I BMC.RET_EMPLS
BMC51807I (EMPL_LNAME VARCHAR(15) NOT NULL
BMC51807I , EMPL_DEPT_NO CHARACTER(4) NOT NULL
BMC51806I , EMPL_HIRE_DATE DATE
BMC51809I )
BMC51809I ;
BMC51805I CREATE TABLE
BMC51809I BMC.HRLY_EMPLS
BMC51807I (EMPL_LNAME VARCHAR(15) NOT NULL
BMC51807I , EMPL_DEPT_NO CHARACTER(4) NOT NULL
BMC51806I , EMPL_HIRE_DATE DATE
BMC51809I )
BMC51809I ;
BMC51810I LOAD DATA INDDN DATA
BMC51800I ENUMROWS (,40)
BMC51809I REPLACE
BMC51940I EBCDIC CCSID(37,65534,65534)
BMC51801I INTO TABLE
BMC51809I BMC.SAL_EMPLS
BMC51812I WHEN (1:1) = 'S'
BMC51813I (EMPL_LNAME POSITION(2:*+) VARCHAR
BMC51815I ,EMPL_DEPT_NO POSITION(*:*+3) CHAR  (4)
BMC51815I ,EMPL_HIRE_DATE POSITION(*:*+9) DATE  EXTERNAL(10)
BMC51814I NULLIF BMC_NULL1=X'6F'
BMC51819I     ,     BMC_NULL1 POSITION(*) CHAR(1)
BMC51809I )
BMC51811I INTO TABLE
BMC51809I BMC.RET_EMPLS
BMC51812I WHEN (1:1) = 'R'
BMC51813I (EMPL_LNAME POSITION(2:*+) VARCHAR
BMC51815I ,EMPL_DEPT_NO POSITION(*:*+3) CHAR  (4)
BMC51815I ,EMPL_HIRE_DATE POSITION(*:*+9) DATE  EXTERNAL(10)
BMC51814I NULLIF BMC_NULL1=X'6F'
BMC51819I     ,     BMC_NULL1 POSITION(*) CHAR(1)
BMC51809I )
BMC51811I INTO TABLE
BMC51809I BMC.HRLY_EMPLS
BMC51812I WHEN (1:1) = 'H'
BMC51813I (EMPL_LNAME POSITION(2:*+) VARCHAR
BMC51815I ,EMPL_DEPT_NO POSITION(*:*+3) CHAR  (4)
BMC51815I ,EMPL_HIRE_DATE POSITION(*:*+9) DATE  EXTERNAL(10)
BMC51814I NULLIF BMC_NULL1=X'6F'
BMC51819I     ,     BMC_NULL1 POSITION(*) CHAR(1)
BMC51809I )
BMC50476I DDNAME = CNTL, I/O WAIT = 1, I/O WAITS = 1, RDB LOCK WAIT = 0
BMC50006I UTILITY EXECUTION COMPLETE, RETURN CODE = 0

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. Table 72 describes the key command options that this job uses.

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.

Table 72  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>

Figure 37 shows the JCL for example 9.

Figure 37  JCL for example 9 (part 1 of 2)

```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
```
Example 9: Unloading from multiple table spaces

Figure 37  JCL for example 9 (part 2 of 2)

```
//SYSIN DD *  
UNLOAD  
OUTPUT SYSREC DSNAME 'ADU.EXAM09.SYSRC&SELNUM'  
*  
ENUMROWS 100  
*  
SELECT *  
FROM BMC.EMPLS  
ORDER BY HIRE_DATE  
*  
SELECT *  
FROM BMC.RET_EMPLS  
*/  
//SYSCNTL DD DSN=ADU.EXAM09.SYSCNTL,DISP=(NEW,CATLG),  
//            UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)  
//SYSPRINT DD SYSOUT=*  
//UTPRINT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
```

Figure 38 shows the SYSPRINT for example 9.

Figure 38  SYSPRINT for example 9 (part 1 of 4)

```
****** B M C   U N L O A D   P L U S   F O R   D B 2    V11R1.00 ******  
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.  
BMC5001I UTILITY EXECUTION STARTING   10/28/2013   13:04:39 ...  
BMC5002I UTILITY ID = 'ADUXM09'.  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=1413308K,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  
BMC50471I SOLUTION COMMON CODE--V11.01.00  
BMC50471I MAINT: BPJ0661  BPJ0667  BPJ0671  BPJ0674  BPJ0675  BPJ0682  BPJ0686  BPJ0689  BPJ0697  
BMC50471I BMCSDRT ENGINE--V02.04.01  
BMC50471I MAINT: BPJ0691  
BMC50471I BMC STATS API--V11.01.00  
BMC50471I MAINT: BPUS409  BPUS534  BPUS674  
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00  
BMC50471I MAINT: BPE0401  BPE0403  BPE0405  BPE0407  BPE0410  BPE0412  BPE0416  
BMC50471I ANALYZE=(DB2STATS,NOLIMIT)  
FDRCRT_PRT=NO  
SDRLEVEL=16  
BMC50471I CENTURY=(1950,2049)  
FDRCRT=NONE  
SMCORE=(OK,OK)  
BMC50471I CHANGE_CONSISTENT=NO  
HISTORY=YES  
SDRNUM=32  
BMC50471I CHANGE QUESSE=NO  
IRUFFS=25  
SOLDELAY=3  
BMC50471I CMX=16  
INLINE=NO  
SOLPRETRY=100  
BMC50471I CMRATIO=50  
LOADDECMP=NO  
TAPEDISP=DELETE  
BMC50471I CONSTRTUES+BMC  
LOCKROW=YES  
TASKMAX=200%  
BMC50471I CURRENTFIGURE=NONE  
MAXP=5  
UMBFFS=25  
BMC50471I DFILES=NONE  
MSELEV=1  
UNLOMAX=200%  
BMC50471I DMNDELAY=1  
NULLCHAR='X'6F'  
UNLOADN_ACTIVE=(YES,NO)  
BMC50471I DRNETXRY=255  
NULLTYPETI=1  
UNLODON=(SYSREC,SYSRED)  
BMC50471I DRNWAIT=NONE  
DPNABDID=YES  
USERECON=NO  
BMC50471I DSPLOCKS=DRNDFAIL  
PLAN=ADUQA  
UXSTATE=SUP  
BMC50471I EXCLUDMP=(X37,X22,X06)  
RECFM=AUTO  
WORKUNIT=SYSALLDA  
BMC50471I FILEFUNCTION=SYSREF  
ROWSETSZ=100  
ZIIP=ENABLED  
BMC50471I FILE=NO  
SDUMP=YES  
ZONEDDECOVP=(C,D)  
BMC50471I FORCE_AT=(START,3)  
SHLEVEL=REFERENCE  
BMC50471I TAPES=NONE  
BMC50471I OUTPUT = SYSREC  
SYSREC  
SYSREF  
BMC50471I UNIT = SYSALLODA  
SYSAULA  
SYSALLODA  
BMC50471I VOLCNT = 25  
25  
25  
BMC50471I GDGLIMIT = 5  
5  
5  
```
Example 9: Unloading from multiple table spaces

Figure 38  SYSPRINT for example 9 (part 2 of 4)
Example 9: Unloading from multiple table spaces

```
BMC51681I ESTIMATED ROWS FOR TABLE BMC.HRLY_EMPLS IS 6240 FROM TABLESPACE
BMC51681I ESTIMATED ROWS FOR TABLE BMC.SAL_EMPLS IS 6240 FROM TABLESPACE
BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 20
BMC51689I ESTIMATED ROWS FOR SELECT 2 IS 6240

BMC50041I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC50441I UNLOAD PLUS DYNAMIC FILE ALLOCATION REPORT

BMC50446I UNIT OR UNIT OR
BMC50447I KBYTES KBYTES ALOC ALOC
BMC50448I DDNAME DSNAME DATACLASS MGMTCLASS STORCLASS PRI SEC PRI SEC
BMC50448I SYS00002 ADU.EXAMPL09.SYSRC001 SYSALLDA 4 0 1 1 CYL
BMC50448I SYS00003 ADU.EXAMPL09.SYSRC002 SYSALLDA 495 0 1 1 CYL

BMC51639I FOR DDNAME 'SYS00002' DSN=ADU.EXAMPL09.SYSRC001,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=111)
BMC51639I FOR DDNAME 'SYS00003' DSN=ADU.EXAMPL09.SYSRC002,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=81)
BMC50394I UNABLE TO LOCATE SORT WORK DATASETS, DDNAME = 'SORTWKNN'

BMC50894I PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 13:04:39

BMC50650I ESTABLISHING A POINT OF CONSISTENCY FOR:
BMC50651I SPACE BMCEXDB.BMCEXATS PART(0) PSID(0002)
BMC50651I SPACE BMCEXDB.BMCEXBTS PART(0) PSID(0007)

BMC50895I PAGE EXTERNALIZATION PROCESS COMPLETE. ELAPSED TIME = 00:00:00

BMC50041I 0: ZIIP ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50474I BELOW 16M = 8476K, ABOVE 16M = 1405300K, CPUS = 3
BMC51701I MAX TASKS = 2, MAX PARTITIONS PER TASK = 4, SORTWKS PER TASK = 32, MAX OPEN PARTITIONS PER TASK = 4
BMC50477I 1: PARTITION = 1, ROWS/KEYS = 8, I/O WAITS = 2, DDNAME = SYS00004
BMC50477I 1: PARTITION = 2, ROWS/KEYS = 5, I/O WAITS = 2, DDNAME = SYS00008
BMC50477I 2: PARTITION = 0, ROWS/KEYS = 20, I/O WAITS = 2, DDNAME = SYS00005
BMC50477I 1: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 2, DDNAME = SYS00009
BMC50478I 2: RDB LOCK WAITS = 0
BMC50477I 1: PARTITION = 4, ROWS/KEYS = 2, I/O WAITS = 2, DDNAME = SYS00010
BMC50478I 1: RDB LOCK WAITS = 0
BMC50476I DDNAME = SYS00002, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYS00003, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0

BMC51671I UNLOAD STATISTICS: 8 ROWS PROCESSED FROM PARTITION 1
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A001' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 2
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A002' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 3
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A003' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A004' READ 3 PAGES
BMC51672I UNLOAD STATISTICS: 20 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 0 NOT SELECTED, 0 DISCARDED
BMC51686I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A001' READ 3 PAGES
BMC51670I UNLOAD STATISTICS: 20 ROWS PROCESSED FROM TABLE 'BMC.RET_EMPLS', 0 NOT SELECTED, 0 DISCARDED
BMC51672I UNLOAD STATISTICS: 20 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 0 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 20 RECORDS WRITTEN TO DDNAME 'SYS00002'
BMC51674I UNLOAD STATISTICS: 20 RECORDS WRITTEN TO DDNAME 'SYS00003'
BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50894I UNLOAD PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPL09.SYSCNTL,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)
BMC51801I LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'
BMC51809I BMC.EMPLS
BMC51940I   INTO TABLE
BMC51810I   INTO TABLE
BMC51811I   INTO TABLE
BMC51809I   INTO TABLE
BMC51809I   INTO TABLE
BMC51813I   INTO TABLE
BMC51815I   INTO TABLE
BMC51814I   INTO TABLE
BMC51813I   INTO TABLE
BMC51815I   INTO TABLE
BMC51814I   INTO TABLE
BMC51813I   INTO TABLE
BMC51815I   INTO TABLE
BMC51814I   INTO TABLE

Figure 38 SYSPRINT for example 9 (part 3 of 4)
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 338. Table 73 on page 358 describes the key command options for this job.

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.
Example 10: Unloading from incremental image copies

Table 73  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 SYSIIBM.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 BMC.EMPLS</td>
<td>names the table to unload</td>
</tr>
</tbody>
</table>

Figure 39 shows the JCL for example 10.

Figure 39  JCL for example 10

```plaintext
// 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)
// SYSREC DD DSN=ADU.EXAMPL10.SYSREC,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
// SYSPRINT DD SYSOUT=* 
// UTPRINT DD SYSOUT=*
// SYSUDUMP DD SYSOUT=*
```

Figure 40 shows the SYSPRINT for example 10.

Figure 40  SYSPRINT for example 10 (part 1 of 4)

```plaintext
***** BMC UNLOAD PLUS FOR DB2 V11R00 ****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50001I UTILITY EXECUTION STARTING 10/28/2013 13:05:33 ...
BMC50002I UTILITY ID = 'ADUXM10'. 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,BELLOW 16M=8848K,ABOVE 16M=1410384K,IEFUSI=NO,CPU5=3
BMC50471I MEMLIMIT=1759216040320M,AVAILABLE=1759216040319M,MEMLIMIT SET BY:REGION=0

BMC50471I UNLOAD PLUS FOR DB2--V11.01.00
```
Figure 40    SYSPRINT for example 10 (part 2 of 4)

Example 10: Unloading from incremental image copies
Example 10: Unloading from incremental image copies

**Figure 40** SYSPRINT for example 10 (part 3 of 4)

```
BMC5047I ENCODING SCHEME = EBCDIC
BMC5047I APPL. ENCODING SCHEME = EBCDIC
BMC5047I MIXED = NO
BMC5047I EBCDIC CCSID = (37,65534,65534)
BMC5047I ASCII CCSID = (819,65534,65534)
BMC5047I UNICODE CCSID = (367,1208,1200)
BMC5047I IMPLICIT TIME ZONE = CURRENT (-05:00)

BMC50028I DB2 MODE = NFM
BMC5047I BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'
BMC5047I BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC'
BMC5047I BMC_BMCHIST = 'BMCUTIL.CMN_BMCHIST'
BMC5047I BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCCOPY'

BMC50102I UNLOAD INFILE IMAGECOPY INCREMENTAL -1
BMC50102I PART 1
BMC50102I UNLOADIN SYSREC ACTIVE NO
BMC50102I SELECT *
BMC50102I FROM BMC.EMPLS

BMC51654I DIRECT YES IN EFFECT
BMC51655I LOCATING INCREMENTAL IMAGE COPIES
BMC51624I FOR PARTITION 1, IMAGE COPY 'ADU.QA.ICOPY.INCR1.BMCEXATS.PTN01'(2013-10-28-13.04.59.607992) WILL BE USED
BMC51655I LOCATING FULL IMAGE COPIES
BMC51624I FOR PARTITION ALL, IMAGE COPY 'ADU.QA.ICOPY.FULL1.BMCEXATS'(2013-10-28-13.04.59.231172) WILL BE USED

BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 0

BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSREC' DSN=ADU.EXAMPL10.SYSREC,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=111)

BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM
BMC51657I DIRECT YES IN EFFECT

BMC51686I LOADING OF DATASET 'ADU.QA.ICOPY.INCR1.BMCEXATS.PTN01'(2013-10-28-13.04.59.607992) WILL BE USED

BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 0

BMC51686I LOADING OF DATASET 'ADU.QA.ICOPY.FULL1.BMCEXATS'(2013-10-28-13.04.59.231172) WILL BE USED

BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 0

BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPL10.SYSCNTL,DCB=(RECFM=F8,BLKSIZE=3120,LRECL=80)

BMC51801I LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'
BMC51809I LOAD DATA INDDN SYSREC
BMC51810I ...
BMC51813I ...
```
Example 11: Using a DDLIN data set

This example uses the OBID option and the INFILE option with a file that contains DDL. Table 74 describes the key command options for this job.

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.

Table 74 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>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>OBID</td>
<td>specifies the OBID that will be associated with the selected table</td>
</tr>
<tr>
<td></td>
<td>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>
Figure 41 shows the JCL for example 11.

**Figure 41  JCL for example 11**

```
// JOB
//UNLOAD11 EXEC PGM=ADUUMAIN, REGION=0M,
// PARM=(DEHJ,'ADUXM11','NEW ','MSGLEVEL(1)')
//*********************************************************************
//STEP18 DD DISP=SHR, DSN=product.libraries
// DO DISP=SHR, DSN=DB2.DSNEXIT
// DO DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
// UNLOAD INFILE BMCCPY
// UNLOADON SYSREC ACTIVE NO
// select 'ADU11TB1', * FROM ADU.ADU11TB1 (OBJID 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.DSN1COPY.ADU11TS1, DISP=SHR
// BMCCPY02 DD DSN=ADU.QA.DSN1COPY.ADU11TSB, DISP=SHR
// SYSCNTL DD DSN=ADU.EXAMPLE11.SYSCNTL, DISP=(NEW,CATLG),
// UNIT=SYSDA, SPACE=(CYL,(3,1),RLSE)
// SYSCREC01 DD DSN=ADU.EXAMPLE11.SYSCREC01, DISP=(NEW,CATLG),
// UNIT=SYSDA, SPACE=(CYL,(3,1),RLSE)
// SYSCREC02 DD DSN=ADU.EXAMPLE11.SYSCREC02, DISP=(NEW,CATLG),
// UNIT=SYSDA, SPACE=(CYL,(3,1),RLSE)
// SYSPRINT DD SYSOUT=*  // UTPRINT DD SYSOUT=*  // SYSUDUMP DD SYSOUT=*  
```

Figure 42 shows the first DDLIN data set that example 11 uses.

**Figure 42  First DDLIN data set for example 11 (part 1 of 2)**

```
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
```
Figure 42  First DDLIN data set for example 11 (part 2 of 2)

```sql
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
  ,COL7_INTEGER       INTEGER               NOT NULL WITH DEFAULT
  )
  IN ADU11DB.ADU11TSA
  AUDIT NONE ;
COMMIT
;
```

Figure 43 shows the second DDLIN data set that example 11 uses.

Figure 43  Second DDLIN data set for example 11

```sql
CREATE TABLESPACE ADU11TSB  IN ADU11DB
  USING      STOGROUP ADUSTG1
  CCSID EBCDIC
  FREEPAGE  10
  PCTFREE   10
  LOCKSIZE  PAGE
  CLOSE  YES
  SEGSIZE  4 ;
COMMIT
;
CREATE TABLE ADU.ADU11TB2
  (COL1_INTEGER       INTEGER               NOT NULL WITH DEFAULT
  ,COL2_CHAR_100      CHARACTER(100)        NOT NULL WITH DEFAULT
  ,COL3_CHAR_100      CHARACTER(100)        NOT NULL WITH DEFAULT
  ,COL4_VCHAR_250     VARCHAR(250)          NOT NULL WITH DEFAULT
  ,COL5_VCHAR_250     VARCHAR(250)          NOT NULL WITH DEFAULT
  ,COL6_TIMESTAMP     TIMESTAMP             NOT NULL WITH DEFAULT
  )
  OBID 1202
  IN ADU11DB.ADU11TSB
  AUDIT NONE ;
COMMIT
;
```

Figure 44 on page 364 shows the SYSPRINT for example 11.
Figure 44  SYSPRINT for example 11 (part 1 of 3)

** **** BMC UNLOAD PLUS FOR DB2 V11R1.00 ****

(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.

BMC50001I UTILITY EXECUTION STARTING  10/29/2013  8:53:11 ...
BMC50002I UTILITY ID = 'ADUXM11'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'ADUOPTS'.
BMC50471I 05 2.1.0, PID=HBB7790, DFSMS FOR 2/05 2.1.0, DB2=11R1.0
BMC50471I REGDUMP, BELOW 16M=BE24X, ABOVE 16M=147704X, IEFUSI=NO, CPUS=3
BMC50471I MEMLIMIT=175921600, AVAILABLE=175921600, MEMLIMIT SET BY: REGION=O

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

BMC50471I SOLUTION COMMON CODE -- V11.01.00

BMC50471I MAINT: BPJ0661 BPJ0667 BPJ0671 BPJ0674 BPJ0675 BPJ0682 BPJ0686 BPJ0689 BPJ0697

BMC50471I MAINT: BPJ0691

BMC50471I MAINT: BPUS409 BPUS534 BPUS5674

BMC50471I EXTENDED BUFFER MANAGER -- V06.01.00

BMC50471I MAINT: BPJ0641 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416

BMC50471I ANALYZE=(DB2STATS,NULLIMIT)  FDOCE_RPT=NO  SMAX=16

BMC50471I CENTURY=(1950,2049)  FDOCE=NONE  SMCORE=(OK, OK)

BMC50471I CHANGE_CONSISTENT=NO  HISTORY=YES  SORTNIM=52

BMC50471I CHANGE_QUESCE=NO  IBUFFS=25  SPOOLDELAY=3

BMC50471I CMAX=16  INLINE=NO  SOLRETRY=100

BMC50471I CMRATIO=50

BMC50471I CMRATIO=50

BMC50471I CONSTRUCTS=MAC

BMC50471I LOCKNOW=YES  TASKMAX=200%

BMC50471I CURRENTDEGREE=NONE  MAXP=5

BMC50471I DFILES=YES  MSGLEVEL=1  UNLDMAX=200%

BMC50471I DRNDELAY.+  NULLCHAR=46'  UNLOADON_ACTIVE=(YES, NO)

BMC50471I RDRNTRY=255  NULLTYPE=11  UNLOADON=(SYSREC, SYSRED)

BMC50471I DRNWAIT=NONE  OMPDB2ID=YES  USEXLRECl=NO

BMC50471I DPMLOCKS=DRNFAIL  PLAN=WDA0A  UXSTATE=SUP

BMC50471I EXCLUDRUMP=X37, X22, X06  RECFM= AUTO  WORKUNIT=SYSALLA

BMC50471I FILEERRORS=SYSRED  ROWSETSZ=100  ZIIP=ENABLED

BMC50471I FILL=NONE  ZONEDDECOVP=(C,D)

BMC50471I FORCE AT=(START,3)  SHRLEVEL=REFERENCE

BMC50471I TAPES=NONE

BMC50470I OUTPUT = SYSREC  SYSREC  SYSREC

BMC50470I UNIT = SYSALLDA  SYSALLDA  SYSALLDA

BMC50470I VOLUME = 25  25  25

BMC50470I DRDLIMIT = 5  5  5

BMC50470I DDGEMPTY = NO  NO  NO

BMC50470I DDGERRAT = NO  NO  NO

BMC50470I STORCLAS = NONE  NONE  NONE

BMC50470I DATACLAS = NONE  NONE  NONE

BMC50470I MMMLCLAS = NONE  NONE  NONE

BMC50470I UNITCNT = 0  0  0

BMC50470I SPACE = CYL  CYL  CYL

BMC50470I PITPRIM = AUTO  AUTO  AUTO

BMC50470I MAXPRIM = 0  0  0

BMC50470I MAXSEC = 0  0  0

BMC50470I FILESIZE=100  100  100

BMC50470I NBRSEC = AUTO  AUTO  AUTO

BMC50470I DISKRETN = NONE  NONE  NONE

BMC50470I DISKEQPD = NONE  NONE  NONE

BMC50470I RETPD = NONE  NONE  NONE

BMC50470I EXPDT = 99000  99000  99000

BMC50470I ITRCH = NONE  NONE  NONE

BMC50470I DSTYPE = NONE  NONE  PDS

BMC50483I SYSREC VOLUMES=NONE

BMC50483I SYSREC VOLUMES=NONE

BMC50483I SYSREC VOLUMES=NONE

BMC50483I SYSREC DNAME=USERID.BTYPE.S&SELNUM

BMC50483I SYSREC DNAME=USERID.BTYPE.S&SELNUM

BMC50483I SYSREC DNAME=USERID.BTYPE.S&SELNUM

BMC50471I DB2 DSLIMUSER MODULE SETTINGS:

BMC50471I VERSION = 1110
Example 11: Using a DDLIN data set

Chapter 5 Examples of UNLOAD PLUS jobs

Figure 44  SYSPRINT for example 11 (part 2 of 3)
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.

Table 75 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>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</td>
<td>disables dynamic allocation for the primary unload data set, overriding the default in the installation options module</td>
</tr>
<tr>
<td>ACTIVE NO</td>
<td></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>

Figure 45 shows the JCL for example 12.
Figure 45  JCL for example 12 (part 2 of 2)

```plaintext
/*SYSLIN DD * *

//SYSPRINT DD SYSOUT=* //SYSCNTL DD DSN=ADU.EXAMPL12.SYSCNTL,DISP=(NEW,CATLG), //UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE) //SYSREC DD DSN=ADU.EXAMPL12.SYSCNTL,DISP=(NEW,CATLG), //UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE) //SYSPRINT DD SYSOUT=* //UTPRINT DD SYSOUT=* //SYSUDUMP DD SYSOUT=*```

Figure 46 shows the SYSPRINT for example 12.

Figure 46  SYSPRINT for example 12 (part 1 of 3)

```plaintext
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****  10/28/2013 13:08:05 ...
```

---

Example 12: Unloading to CSV format

Chapter 5  Examples of UNLOAD PLUS jobs
### Example 12: Unloading to CSV format

#### Figure 46  SYSPRINT for example 12 (part 2 of 3)

<table>
<thead>
<tr>
<th>BMC50470I UNITCNT = 0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50470I SPACE = CYL</td>
<td>CYL</td>
<td>CYL</td>
</tr>
<tr>
<td>BMC50470I PCTPRIM = AUTO</td>
<td>AUTO</td>
<td>AUTO</td>
</tr>
<tr>
<td>BMC50470I MAXPRIM = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I MAXSECD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I FILESZPCT = 100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>BMC50470I NBRSECD = AUTO</td>
<td>AUTO</td>
<td>AUTO</td>
</tr>
<tr>
<td>BMC50470I DISKRETN = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I DISKEXPD = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I RETPD = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I EXPDT = 99000</td>
<td>99000</td>
<td>99000</td>
</tr>
<tr>
<td>BMC50470I TRTCH = NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>BMC50470I DSNTYPE = NONE</td>
<td>NONE</td>
<td>PDS</td>
</tr>
<tr>
<td>BMC50483I SYSREC VOLUMES = NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I SYSRED VOLUMES = NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I SYSREF VOLUMES = NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I SYSREC DSNAME = &amp;USERID.&amp;TYPE.S&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I SYSRED DSNAME = &amp;USERID.&amp;TYPE.S&amp;SELNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50483I SYSREF DSNAME = &amp;USERID.BMC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BMC50471I** DB2 DSNHDECP MODULE SETTINGS:

<table>
<thead>
<tr>
<th>BMC50471I VERSION = 1110</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I SUBSYSTEM DEFAULT = DEHJ</td>
</tr>
<tr>
<td>BMC50471I CHARACTER SET = ALPHANUM</td>
</tr>
<tr>
<td>BMC50471I DATE FORMAT = USA</td>
</tr>
<tr>
<td>BMC50471I TIME FORMAT = USA</td>
</tr>
<tr>
<td>BMC50471I LOCAL DATE LENGTH = 0</td>
</tr>
<tr>
<td>BMC50471I LOCAL TIME LENGTH = 0</td>
</tr>
<tr>
<td>BMC50471I DECIMAL POINT = PERIOD</td>
</tr>
<tr>
<td>BMC50471I DECIMAL ARITHMETIC = IS</td>
</tr>
<tr>
<td>BMC50471I DELIMITER = DEFAULT</td>
</tr>
<tr>
<td>BMC50471I SQL DELIMITER = DEFAULT</td>
</tr>
<tr>
<td>BMC50471I ENCODING SCHEME = EBCDIC</td>
</tr>
<tr>
<td>BMC50471I APPL. ENCODING SCHEME = EBCDIC</td>
</tr>
<tr>
<td>BMC50471I MIXED = NO</td>
</tr>
<tr>
<td>BMC50471I EBCDIC CCSID = (37,65534,65534)</td>
</tr>
<tr>
<td>BMC50471I ASCII CCSID = (819,65534,65534)</td>
</tr>
<tr>
<td>BMC50471I UNICODE CCSID = (367,1208,1200)</td>
</tr>
<tr>
<td>BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)</td>
</tr>
</tbody>
</table>

**BMC50481I** DB2 MODE = NFM

**BMC50471I** BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'

**BMC50471I** BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY'

**BMC50471I** BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC'

**BMC50471I** BMC_BMCHIST = 'BMCUTIL.CMN_BMCHIST'

**BMC50102I** FORMAT CSV ENCLOSEDBY "" AND "" NULLSTRING ""

**BMC50102I** UNLOAD DN SYSREC ACTIVE NO

**BMC50102I** SELECT *

**BMC50102I** FROM BMC.EMPLS

**BMC50102I** *

**BMC5164I** DIRECT YES IN EFFECT

**BMC51639I** FOR DDNAME 'SYSREC' DSN=ADU.EXAMPL12.SYSREC,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=150)

**BMC51639I** FOR DDNAME 'SYSREC' DSN=ADU.EXAMPL12.SYSREC,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=150)

**BMC50102I** PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 13:08:05

**BMC50650I** ESTABLISHING A POINT OF CONSISTENCY FOR:

**BMC50651I** SPACE BMCEXDB.BMCEXATS PART(0) PSID(0002)

**BMC50004I** UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00

**BMC51659I** FOR DDNAME 'SYSREC' DSN=ADU.EXAMPL12.SYSREC,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=150)

**BMC50004I** UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00

**BMC50894I** PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 13:08:05

**BMC50650I** ESTABLISHING A POINT OF CONSISTENCY FOR:

**BMC50651I** SPACE BMCEXDB.BMCEXATS PART(0) PSID(0002)
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.

Table 76 on page 370 describes the key command options for this job.
Table 76  Command options for example 13

<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</td>
</tr>
<tr>
<td></td>
<td>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 ANY</td>
<td>tells UNLOAD PLUS to issue the SET CURRENT DEGREE command before executing your dynamically executed SELECT statements</td>
</tr>
<tr>
<td></td>
<td>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>

Figure 47 shows the JCL for example 13.

Figure 47  JCL for example 13 (part 1 of 2)

```sql
// JOB
//UNLOAD13 EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM=(DEHJ,'ADUXM13','NEW ',,'MSGLEVEL(1)')
//******************************************************************************
//STEPLIB DD DISP=SHR,DSN=*
//                   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,
        EMPL_ID
FROM BMC.EMPLS INNER JOIN BMC.RET_EMPLS
   ON BMC.EMPLS.EMPL_LNAME = BMC.RET_EMPLS.EMPL_LNAME;
```
Example 13: Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism

Chapter 5 Examples of UNLOAD PLUS jobs 371

**Figure 48** shows the SYSPRINT for example 13.

**Figure 47** JCL for example 13 (part 2 of 2)

```sql
* 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)
//SYSCREC01 DD DSN=ADU.EXAMPL13.SYSCREC01,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCREC02 DD DSN=ADU.EXAMPL13.SYSCREC02,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCREC03 DD DSN=ADU.EXAMPL13.SYSCREC03,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=* 
//UTPRINT  DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
```

**Figure 48** SYSPRINT for example 13 (part 1 of 4)

```
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.

BMC50001I UTILITY EXECUTION STARTING   10/28/2013   13:08:17 ...
BMC50002I 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=16M,BELLOW 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
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT: BPJ0661 BPJ0667 BPJ0671 BPJ0674 BPJ0675 BPJ0676 BPJ0682 BPJ0686 BPJ0689 BPJ0697
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAINT: BPJ0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I MAINT: BPUS0409 BPUS534 BPUS674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I MAINT: BPE0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416

BMC50471I ANALYZE=(DB2STATS,NOMAX) FORCE_RPT=NO SMAX=16
BMC50471I CENTURY=(1950,2049) FORCE=NONE SMCORE=(0K,0K)
BMC50471I CHANGE_CONSISTENT=NO HISTORY=YES SORTNUM=32
BMC50471I CHANGE_QUIESCE=NO IBUFFS=25 SPOOL=3
BMC50471I CMAX=16 INLINED=NO SDELAY=100
BMC50471I CMRATIO=50 LOADADCP=NO TATEDISP=DELETE
BMC50471I CONSTRULES=BMC LOCK=NO YES TASKMAX=2001
BMC50471I CURRENTDEGREE=NONE MAXP=5 UNBUFFS=25
BMC50471I DELFILES=NO MSGLEVEL=1 UNLOMAX=2000
BMC50471I DRNDATA=1 NULLCHAR='6F' UNLOADMAX=ACTIVE=(YES,NO)
BMC50471I DRNDEFAULT=255 NULLTYPE=T1 UNLOADMAX=(SYSREC,SYSRED)
BMC50471I DRNWRITE=NONE OPRED=NO USESREC=NO
BMC50471I DSPBOCKS=DNOPFAIL PLAN=ADVQA USSTATE=SUP
BMC50471I EXCLUD=(X37,X22,X06) RECFM=AUTO WORKUNIT=SYSALLDA
BMC50471I FILERETURN=SRYSREF ROWSETSZ=100 ZIIP=ENABLED
BMC50471I FILE=NO SDUMP=YES ZONEDDECOVP=(C,D)
BMC50471I FORCE_AT=(START,3) SHRLEVEL=REFERENCE
BMC50471I TAPES=NONE

BMC50471I OUTPUT = SYSREC SYSEX SYSEX
BMC50471I UNIT = SYSALDA SYSAUX SYSAUX
BMC50471I VOLUMEN = 25 25 25
BMC50471I GDGLIMIT = 5 5 5
```
Example 13: Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism

**Figure 48  SYSPRINT for example 13 (part 2 of 4)**

```
BMC50470I CPUEMPTY = NO  NO  NO
BMC50470I GDGSCRAT = NO  NO  NO
BMC50470I STORCLAS = NONE  NONE  NONE
BMC50470I DATACLAS = NONE  NONE  NONE
BMC50470I MGMTCLAS = NONE  NONE  NONE
BMC50470I UNITCNT = 0  0  0
BMC50470I SPACE = CYL  CYL  CYL
BMC50470I PCTPRIM = AUTO  AUTO  AUTO
BMC50470I MAXPRIM = 0  0  0
BMC50470I MAXSECD = 0  0  0
BMC50470I FILESZPCT = 100  100  100
BMC50470I NBRSECD = AUTO  AUTO  AUTO
BMC50470I DISKRETN = NONE  NONE  NONE
BMC50470I DISKEXPD = NONE  NONE  NONE
BMC50470I RETPD = NONE  NONE  NONE
BMC50470I EXPD = 99000  99000  99000
BMC50470I TRTCH = NONE  NONE  NONE
BMC50470I DSNTYPE = NONE  NONE  PDS
BMC50483I SYSPRINT VOLUMES=NONE
BMC50483I SYSPRINT VOLUMES=NONE
BMC50483I SYSPRINT VOLUMES=NONE
BMC50483I SYSPRINT VOLUMES=NONE
BMC50483I SYSPRINT VOLUMES=NONE
BMC50483I SYSPRINT VOLUMES=NONE

BMC50471I DB2 DSNHDECP MODULE SETTINGS:
BMC50471I VERSION = 1110
BMC50471I SUBSYSTEM DEFAULT = DEHJ
BMC50471I CHARACTER SET = ALPHANUM
BMC50471I DATE FORMAT = USA
BMC50471I TIME FORMAT = USA
BMC50471I LOCAL DATE LENGTH = 0
BMC50471I LOCAL TIME LENGTH = 0
BMC50471I DECIMAL POINT = PERIOD
BMC50471I DECIMAL ARITHMETIC = 15
BMC50471I DELIMITER = DEFAULT
BMC50471I SQL DELIMITER = 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 UNICODE CCSID = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)
BMC50502I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST = 'BMCUTIL.CMN_BMCHIST'
BMC50471I BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY'

BMC50102I UNLOAD
BMC50102I DIRECT NO
BMC50102I CNTLCARDS DB2
BMC50102I CURRENTDEGREE ANY
BMC50102I UNLOADON SYSPRINT ACTIVE NO
BMC50102I *
BMC50102I SELECT EMPL_LNAME,
BMC50102I EMPL_SALARY
BMC50102I FROM BMC.SAL_EMPLS A
BMC50102I GROUP BY EMPL_LNAME
BMC50102I HAVING MAX(EMPL_SALARY) = (SELECT MAX(EMPL_SALARY_HRLY)
BMC50102I FROM BMC.RET_EMPLS
BMC50102I WHERE EMPL_LNAME = A.EMPL_LNAME);
BMC50102I *
BMC50102I SELECT DISTINCT BMC.EMPLS.EMPL_LNAME,
BMC50102I SSN,
BMC50102I EMPL_ID
BMC50102I FROM BMC.EMPLS INNER JOIN BMC.RET_EMPLS
BMC50102I ON BMC.EMPLS.EMPL_LNAME = BMC.RET_EMPLS.EMPL_LNAME;
BMC50102I *
BMC50102I SELECT EMPL_DEPT_NO,
BMC50102I MAX(EMPL_HRLY_RATE),
```
Example 13: Using DB2 dynamic SQL (DIRECT NO) and DB2 parallelism

Figure 48  SYSPRINT for example 13 (part 3 of 4)

BMC50102I  MIN(EMPL_HRLY_RATE)
BMC50102I  FROM BMC.HRLY_EMPLS
BMC50102I  GROUP BY EMPL_DEPT_NO;
BMC50102I *

BMC51654I DIRECT NO IN EFFECT

BMC51699I 'DIRECT NO' OPTION SPECIFIED.  OPTION 'SHRLEVEL' IGNORED

BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 0
BMC51689I ESTIMATED ROWS FOR SELECT 2 IS 0
BMC51689I ESTIMATED ROWS FOR SELECT 3 IS 0

BMC50004I UTILINIT PHASE COMPLETE.  ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSREC01' DSN=ADU.EXAMPL13.SYSREC01,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=27)
BMC51639I FOR DDNAME 'SYSREC02' DSN=ADU.EXAMPL13.SYSREC02,DCB=(RECFM=VB,BLKSIZE=27993,LRECL=33)
BMC51639I FOR DDNAME 'SYSREC03' DSN=ADU.EXAMPL13.SYSREC03,DCB=(RECFM=FB,BLKSIZE=27984,LRECL=16)
BMC50041I 0: ZIIP  ENABLED (0) USING XBM SUBSYSTEM XBMB

BMC51641I 3: SELECT STATEMENT NO. 3, ROWS = 10
BMC51641I 2: SELECT STATEMENT NO. 2, ROWS = 20
BMC51641I 1: SELECT STATEMENT NO. 1, ROWS = 8

BMC50474I BELOW 16M = 8492K, ABOVE 16M = 1405468K, CPUS = 3

BMC51702I MAX TASKS = 3
BMC51664I 3: SELECT STATEMENT NO. 3, ROWS = 10
BMC51664I 2: SELECT STATEMENT NO. 2, ROWS = 20
BMC51664I 1: SELECT STATEMENT NO. 1, ROWS = 8

BMC51676I UNLOAD STATISTICS:  8 ROWS PROCESSED FOR SELECT STATEMENT NO. 1, 0 DISCARDED
BMC51676I UNLOAD STATISTICS:  20 ROWS PROCESSED FOR SELECT STATEMENT NO. 2, 0 DISCARDED
BMC51676I UNLOAD STATISTICS:  10 ROWS PROCESSED FOR SELECT STATEMENT NO. 3, 0 DISCARDED

BMC51674I UNLOAD STATISTICS:  8 RECORDS WRITTEN TO DDNAME 'SYSREC01'
BMC51674I UNLOAD STATISTICS:  20 RECORDS WRITTEN TO DDNAME 'SYSREC02'
BMC51674I UNLOAD STATISTICS:  10 RECORDS WRITTEN TO DDNAME 'SYSREC03'

BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBMB

BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPL13.SYSCNTL,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)
BMC51805I DB2 CREATE TABLE AND LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'

BMC51805I CREATE TABLE
BMC51809I RDAMZL.$TABLE1
BMC51807I   ( EMPL_LNAME VARCHAR(15)  NOT NULL
BMC51807I   , $COLUMN2 DECIMAL(9 , 2)
BMC51809I )

BMC51805I CREATE TABLE
BMC51809I RDAMZL.$TABLE2
BMC51807I   ( EMPL_LNAME VARCHAR(15)  NOT NULL
BMC51807I   , SSN CHARACTER(9)  NOT NULL
BMC51807I   , EMPL_ID DECIMAL(5 , 0) NOT NULL
BMC51809I )

BMC51805I CREATE TABLE
BMC51809I RDAMZL.$TABLE3
BMC51807I   ( EMPL_DEPT_NO CHARACTER(4)  NOT NULL
BMC51807I   , $COLUMN2 DECIMAL(9 , 2)
BMC51807I   , $COLUMN3 DECIMAL(9 , 2)
BMC51809I )

BMC51805I LOAD DATA INDDN SYSREC01
BMC51940I   EBCDIC CCSID(37,65534,65534)
BMC51810I INTO TABLE
BMC51809I RDAMZL.$TABLE1
BMC51813I (EMPL_LNAME POSITION(1:*) VARCHAR
BMC51813I ,COLUMN2 POSITION(*:*+4) DECIMAL
BMC51814I , NULLIF BMC_NULL1=X'6F'
BMC51819I , BMC_NULL1 POSITION(*) CHAR(1)
BMC51809I )

BMC51805I LOAD DATA INDDN SYSREC02
BMC51940I   EBCDIC CCSID(37,65534,65534)
BMC51810I INTO TABLE
BMC51809I RDAMZL.$TABLE2
BMC51813I (EMPL_LNAME POSITION(1:*) VARCHAR
BMC51813I ,SSN POSITION(*:*+8) CHAR  (9)
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:

- If the unload is unsuccessful and you specify ON FAILURE ALL TERMINATE
  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, SYSCNTL, 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.
Table 77 describes the key command options for this job.

**Table 77  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</td>
</tr>
<tr>
<td></td>
<td>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, SYSCNTL, and SORTWK files if a failure occurs</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>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>

**Figure 49** shows the JCL for example 14.

**Figure 49  JCL for example 14**

```sql
//        JOB
//UNLOAD14 EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM=(DEHJ,'ADUXM14','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
    ON FAILURE ALL TERMINATE UTILITY RETCODE 11
    DELETEFILES YES
    UNLOADDDN SYSREC ACTIVE NO
    *
    SELECT *
        FROM SYSIBM.SYSCOLUMNS A
        WHERE TBCREATOR='SYSIBM';
/*
//SYSREC01 DD DSN=ADU.EXAMPL14.SYSREC01,DISP=(NEW,CATLG),
//         UNIT=SYSDA,SPACE=(TRK,(1,0),RLSE)
//SYSPRINT DD SYSOUT=* 
//UTPRINT  DD SYSOUT=* 
//SYSDUMPP DD SYSOUT=*
```
Figure 50 shows the SYSPRINT for example 14.

**SYSPRINT for example 14 (part 1 of 2)**

```
****** BMC UNLOAD PLUS FOR DB2 V11R1.00 ******
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.

BMC5001I UTILITY EXECUTION STARTING 10/28/2013 13:08:33 ...
BMC5001I UTILITY ID = 'ADUXM14'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'ADU$OPTS'.
BMC5047I I/O 2.1.0,P1d=H8779DFMS FOR 2/052.1.0,DB2=11.1.0
BMC5047I REGION=1M,BLLOW=16M,BB44K,ABOVE 16M=1413184K,IEFUSI=NO,CPU5=3
BMC5047I MMLIMIT=17592186040320M,AVAILABLE=17592186040311M,MEMLIMIT SET BY:REGION=0

BMC5047I UNLOAD PLUS FOR DB2--V11.01.00
BMC5047I NO MAINTENANCE TO REPORT
BMC5047I DB2 UTILITIES COMMON CODE--V11.01.00
BMC5047I NO MAINTENANCE TO REPORT
BMC5047I SOLUTION COMMON CODE--V11.01.00
BMC5047I MAINT: BPJ0661 BPJ0667 BPJ0671 BPJ0674 BPJ0675 BPJ0676 BPJ0682 BPJ0686 BPJ0688 BPJ0697
BMC5047I BMCSORT ENGINE--V02.04.01
BMC5047I MAINT: BPJ0693
BMC5047I BMC STATS API--V11.01.00
BMC5047I MAINT: BPUS534 BPUS5674
BMC5047I EXTENDED BUFFER MANAGER--V06.01.00
BMC5047I MAINT: BPE0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416

BMC5047I ANALYZE=(DB2STATS,NOLIMIT)  FDRC=ND NO  SMAX=16
BMC5047I CENTURY=(1950,2049)  FDRC=RD NO  SMCORE=(OK,OK)
BMC5047I CHANGE_CONSISTENT=NO  HISTORY=YES  SORTNUM=32
BMC5047I CHANGE_QUIESCE=NO  IBUFFS=25  SOLDELAY=3
BMC5047I CMACT=1K  INL=1M  SOLRETR=100
BMC5047I CMRARIT=50  LOAD_ADEPC=NO  TAPEDISP=DELETE
BMC5047I CONSTRAINTS=BMC  LDCKRON=YES  TASKMAX=200%
BMC5047I CURRENTDRICT=NONE  MAP=5  UBUFFS=25
BMC5047I DELFILES=NO  MSGLEVEL=1  UNLOAD=200%
BMC5047I DRNDELAY=1  NULCHAR='*6F'  UNLOADMAX=(YES,NO)
BMC5047I DRNRETRY=255  NULTYPE=T1  UNLOAD=(SYSREC,SYSRED)
BMC5047I DRNWAIT=NONE  OPMID=24D  USELRECL=NO
BMC5047I EXCLDUMP=(X37,X22,X06)  PLAN=ADUQA  UXSTATE=SUP
BMC5047I GDGLIMIT = 5  RECFM=AUTO  WORKUNIT=SALLDA
BMC5047I GDGEMPTY = NO  TAPEDISP3=DELETE  ZIIP=ENABLED
BMC5047I GDGSCRAT = NO  TAPEDISP=DELETE  ZEDDECOVP=(C,D)
BMC5047I STORCLAS = NONE  TAPEDISP=DELETE
BMC5047I DATACLAS = NONE
BMC5047I MGMTCLAS = NONE
BMC5047I UNITCNT = 0
BMC5047I SPACE = CYL
BMC5047I PCTPRIM = 0
BMC5047I MAXPRIM = 0
BMC5047I MAXSECD = 0
BMC5047I FILESZPCT = 100
BMC5047I NBRSECD = 0
BMC5047I DISKEXPD = NONE
BMC5047I RETPD = NONE
BMC5047I EXPDT = 99000
BMC5047I TRTCH = NONE
BMC5047I DSNTYPE = PDS
BMC5047I SYSREC     VOLUMES=NONE
BMC5047I SYSRED     VOLUMES=NONE
BMC5047I SYSREF     VOLUMES=NONE
BMC5047I SYSREC     DSNAME=&USERID.&TYPE.S&SELNUM
BMC5047I SYSRED     DSNAME=&USERID.&TYPE.S&SELNUM
BMC5047I SYSREF     DSNAME=&USERID.BMC

376  UNLOAD PLUS for DB2 Reference Manual
Example 14: Handling an abnormal termination

Chapter 5 Examples of UNLOAD PLUS jobs

Figure 50  SYSPRINT for example 14 (part 2 of 2)

BMC50471I DB2 DSNHDECPS MODULE SETTINGS:
BMC50471I VERSION = 1110
BMC50471I SUBSYSTEM DEFAULT = DSNH
BMC50471I CHARACTER SET = ALPHANUM
BMC50471I DATE FORMAT = USA
BMC50471I TIME FORMAT = USA
BMC50471I LOCAL DATE LENGTH = 0
BMC50471I LOCAL TIME LENGTH = 0
BMC50471I DECIMAL POINT = PERIOD
BMC50471I DECIMAL ARITHMETIC = 15
BMC50471I DELIMITER = DEFAULT
BMC50471I SQL DELIMITER = 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 UNICODE CCSID = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)
BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCFILE = 'BMCUTIL.CMN_BMCFILE'
BMC50471I BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY'

BMC50102I UNLOAD
BMC50102I DIRECT NO
BMC50102I ON FAILURE ALL TERMINATE UTILITY RETCODE 11
BMC50102I DELETE FILES YES
BMC50102I UNLOAD ON SYSREC ACTIVE NO
BMC50102I *
BMC50102I SELECT *
BMC50102I FROM SYSIBM.SYSCOLUMNS A
BMC50102I WHERE TBCREATOR = 'SYSIBM';

BMC51654I DIRECT NO IN EFFECT
BMC50109I 'DIRECT NO' OPTION SPECIFIED.  OPTION 'SHRLEVEL' IGNORED
BMC50353I FOR BIT DATA COLUMN HIGH2KEY WILL NOT BE TRANSLATED
BMC50353I FOR BIT DATA COLUMN LOW2KEY WILL NOT BE TRANSLATED
BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 0
BMC50041I 0: ZIIP ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50474I BELOW 16M = 8508K, ABOVE 16M = 1407052K, CPUS = 3
BMC50041U UNLOAD TASK NUMBER 1 ABNORMALLY TERMINATED WITH SYSTEM ABEND CODE = D37, REASON CODE = 00000004
BMC50471I QCATEGORY = 0
BMC50411U UNLOAD TASK NUMBER 1 ABNORMALLY TERMINATED WITH SYSTEM ABEND CODE = D37, REASON CODE = 00000004
BMC50471I QCATEGORY = 0
BMC50411U UNLOAD TASK NUMBER 1 ABNORMALLY TERMINATED WITH SYSTEM ABEND CODE = D37, REASON CODE = 00000004
BMC51676I UNLOAD STATISTICS:  0 ROWS PROCESSED FOR SELECT STATEMENT NO. 1, 0 DISCARDED
BMC51675I UNLOAD STATISTICS:  3415 RECORDS WRITTEN TO DDNAME 'SYSREC01'
BMC51674I UNLOAD STATISTICS:  0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50019I UTILITY REQUESTED TO TERMINATE DUE TO FAILURE IN PHASE 'UNLOAD'

BMC50894I DELETE FILES PROCESS STARTING AT 10/28/2013 13:08:34
BMC50318I DATASET SUCCESSFULLY DELETED, DNAME = 'SYSREC01', DSNAME = 'ADU.EXAMPL14.SYSREC01'
BMC50895I DELETE FILES PROCESS COMPLETE.  ELAPSED TIME = 00:00:00

BMC50006I UTILITY EXECUTION COMPLETE, RETURN CODE = 11
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.

Table 78 describes the key command options for this job.

Table 78  Command options for example 15 (part 1 of 2)

<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>
</tbody>
</table>
Example 15: Unloading LOB and XML data; using nonstandard null indicators

Chapter 5  Examples of UNLOAD PLUS jobs  379

Figure 51 shows the JCL for example 15, case 1.

**Table 78  Command options for example 15 (part 2 of 2)**

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULLCHAR HIVAL</td>
<td>tells UNLOAD PLUS to fill the null indicator field with high values</td>
</tr>
<tr>
<td>CNTLCARDS</td>
<td>tells UNLOAD PLUS to write DB2 LOAD control statements</td>
</tr>
<tr>
<td>DB2LOAD 'REPLACE'</td>
<td>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>

**Figure 51  JCL for example 15, case 1**

```sql
//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
  /*/
```
Figure 52 shows the JCL for example 15, case 2.

```
//UNLOAD15B 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) //SYSCTN 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 DSN=('ADU.EXAMP15B.CLOB') DSNTYPE(PDS) DIR (100) SPACE (10,10) SUBSETS 1 UNIT WORK OUTPUT XML001 DSN=('ADU.EXAMP15B.XML') DSNTYPE(PDS) DIR (100) SPACE (10,10) SUBSETS 1 UNIT WORK 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 INTO 
COL_SMALL, 
COL_INTEGER, 
COL_FLOAT, 
COL_CHAR, 
COL_VCHAR, 
COL_DATE, 
COL_TIME, 
COL_TIMESTAMP, 
COL_XML CHAR (45) CLOBF XML001, 
COL_CLOB CHAR (45) CLOBF CLOB01 FROM ADU.TADU_XML4K_NULL_32K_PLUS_C
//*
```

Figure 53 shows the SYSPRINT for example 15, case 1.

```
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50001I UTILITY EXECUTION STARTING 10/28/2013 13:14:12 ...
BMC50002I UTILITY ID = 'ADUXM15'. 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=8840K,ABOVE 16M=1414792K,IEFUSI=NO,CPU=3
BMC50471I MEMLIMIT=17592186040320M,AVAILABLE=17592186040313M,MEMLIMIT SET BY:REGION=0

```

Figure 53 shows the SYSPRINT for example 15, case 1 (part 1 of 4).
Example 15: Unloading LOB and XML data; using nonstandard null indicators

Chapter 5 Examples of UNLOAD PLUS jobs 381

Figure 53 SYSPRINT for example 15, case 1 (part 2 of 4)
Example 15: Unloading LOB and XML data; using nonstandard null indicators

**Figure 53**  SYSPRINT for example 15, case 1 (part 3 of 4)

```
BMC50471I SQL_DELIMITER = 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 UNICODE CCSID = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE = CURRENT (-05:00)

BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC = 'BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST = 'BMCUTIL.CMN_BMCHIST'
BMC50471I BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY'

BMC50102I UNLOAD
BMC50102I DIRECT NO
BMC50102I FORMAT STANDARD
BMC50102I NULLTYPE L2
BMC50102I NULLCHAR HIVAL
BMC50102I CNTLCARDS DB2LOAD 'REPLACE '
BMC50102I SELECT
BMC50102I COL_SMALL,
BMC50102I COL_INTEGER,
BMC50102I COL_FLOAT,
BMC50102I COL_CHAR,
BMC50102I COL_VCHAR,
BMC50102I COL_DATE,
BMC50102I COL_TIME,
BMC50102I COL_TIMESTAMP,
BMC50102I COL_XML,
BMC50102I COL_CLOB
BMC50102I FROM ADU.TADU_XML4K_NULL_32K_PLUS_C

BMC51639I FOR DDNAME 'SYSREC' DSN=ADU.EXAMP15A.SYSREC,DCB=(RECFM=VBS,BLKSIZE=27993,LRECL=32756)
BMC50041I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC51809I REPLACE
BMC51801I LOAD DATA INDDN SYSREC
BMC51813I ,COL_SMALL POSITION(3:4) SMALLINT
BMC51814I ,NULLIF BMC_NULL1=X'FFFF'
BMC51813I ,COL_INTEGER POSITION(7:10) INTEGER
BMC51814I ,NULLIF BMC_NULL2=X'FFFF'
BMC51813I ,COL_FLOAT POSITION(13:20) FLOAT  (53)
BMC51814I ,NULLIF BMC_NULL3=X'FFFF'
BMC51813I ,COL_DATE POSITION(23:32) CHAR  (10)
BMC51814I ,NULLIF BMC_NULL4=X'FFFF'
BMC51813I ,COL_CLOB POSITION(33) CHAR(2)
```
Example 15: Unloading LOB and XML data; using nonstandard null indicators

Chapter 5 Examples of UNLOAD PLUS jobs

Figure 53  SYSPRINT for example 15, case 1 (part 4 of 4)

```sql
BMC51813I .COL_VCHAR POSITION(35:*+)VARCHAR
BMC51814I  NULLIF BMC_NULL5=X'FFFF'
BMC51819I .COL_DATE POSITION(*:*+9) DATE EXTERNAL(18)
BMC51814I  NULLIF BMC_NULL6=X'FFFF'
BMC51819I .COL_TIME POSITION(*:*+7) TIME EXTERNAL(26)
BMC51814I  NULLIF BMC_NULL7=X'FFFF'
BMC51819I .COL_TIMESTAMP POSITION(*:*+25) TIMESTAMP EXTERNAL(26)
BMC51814I  NULLIF BMC_NULL8=X'FFFF'
BMC51819I .COL_XML POSITION(*:*+)XML
BMC51809I  PRESERVE WHITESPACE
BMC51814I  NULLIF BMC_NULL9=X'FFFF'
BMC51819I .COL_CLOB POSITION(*:*+)CLOB
BMC51814I  NULLIF BMC_NULL10=X'FFFF'
BMC51809I )
```

Figure 53 shows the SYSPRINT for example 15, case 2.

Figure 54  SYSPRINT for example 15, case 2 (part 1 of 4)

```sql
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50001I UTILITY EXECUTION STARTING 10/28/2013 13:14:13 ...
BMC50002I UTILITY ID = 'ADUXM15'. DB2 SUBSYSTEM ID = 'DEHJ'. OPTION MODULE = 'ADU$OPTS'.
BMC50471I z/OS 2.1.0.PID=HBB7790,DFSMS FOR 2.1/0=2.1.0,0=DB2=11.1.0
BMC50471I REGION=16M,68400K,ABOVE 16M=1614792K,1EFFU=NO,CPU=3
BMC50471I MCLIMIT=17592186040320M,AVAILABLE=17592186040313M,REGION 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 SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT: BPJ0661 BPJ0662 BPJ0663 BPJ0664 BPJ0665 BPJ0666 BPJ0667 BPJ0668 BPJ0669 BPJ0670
BMC50471I BMCORT ENGINE--V02.04.01
BMC50471I MAINT: BPJ0691
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAINT: BPJ0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I MAINT: BPJ5409 BPJ5454 BPJ5474
BMC50471I EXTENDED BUFFER MGR--V06.01.00
BMC50471I MAINT: BPJ0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416
BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FORCE_RPT=NO SMAX=16
BMC50471I CENTURY=(1950,2049) FORCE=NONE SMAX=(OK,OK)
BMC50471I CHANGE_CONSISTENT=NO FOREC=0 HISTORY=YES SORTINUM=3Z
BMC50471I CHANGE_QUESCE=NO IBUFFS=25 SSOKEY=3
BMC50471I CMAX=16 INLNAME=NO SORDER=100 TAPEDISP=DELETE
BMC50471I CMATIO=10 LOADDEFcrop=NO TASKMAX=2000
BMC50471I CONSTRULES=BMC TASKMAX=NO TASKM=1000
BMC50471I CURRENTDEGREE=NONE UBUFFS=5 TAPEDISP=DELETE
BMC50471I DELFILES=NO UBUFF=25 TAPEDISP=DELETE
BMC50471I DRNDELAY=1 UBUFFS=25 TAPEDISP=DELETE
BMC50471I DRNRETRY=255 UBUFFS=25 TAPEDISP=DELETE
BMC50471I DRNWAIT=NONE UBUFFS=25 TAPEDISP=DELETE
BMC50471I DSPLOCKS=DRNFAIL USELREC=NO TAPEDISP=DELETE
BMC50471I EXCLUDM=(X37,X22,X06) USELDECO=(C,D)
BMC50471I EXCLUDM=(X37,X22,X06) USELDECO=(C,D)
BMC50471I FILEESCON=SYSREF USELDECO=(C,D)
BMC50471I FILEESCON=SYSREF ZONEDDECO=(C,D)
BMC50471I FILEESCON=SYSREF ZIP=DISABLED
BMC50471I FORCE_AT=(START,3) ZIP=DISABLED
BMC50471I TAPES=NONE
BMC50471I OUTPUT = SYSREC SYSREF
BMC50471I UNIT = SYSSDA SYSSDA
BMC50471I VOLCNT = 25
```

Chapter 5  Examples of UNLOAD PLUS jobs 383
Figure 54  SYSPRINT for example 15, case 2 (part 2 of 4)
Example 15: Unloading LOB and XML data; using nonstandard null indicators

**Figure 54**  SYSPRINT for example 15, case 2 (part 3 of 4)
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.
Table 79 describes the key command options for this job.

### Table 79 Command options for example 16

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SYSRECMn</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>
<tr>
<td>DSNNAME ADU.EXAMPL16.BKUP.P&amp;PART</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>UNLOADDN SYSREC, BACKUP</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>ACTIVE (YES,YES)</td>
<td>tells UNLOAD PLUS to dynamically allocate both the primary and secondary unload data sets.</td>
</tr>
<tr>
<td>ANALYZE DB2STATS</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>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>

Figure 55 shows the JCL for example 16.

### Figure 55 JCL for example 16 (part 1 of 2)

```plaintext
// 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 *
EXEC PGM=ADUUMAIN,REGION=0M,
PARM=(DEHJ,'ADUXM16','NEW ',,'MSGLEVEL(1)')
```
Example 16: Using dynamic allocation

Figure 55  JCL for example 16 (part 2 of 2)

```plaintext
UNLOAD
OUTPUT SYSREC01 UNIT SYSDA
  SPACE (5,1)
  DSNAME 'ADU.EXAMPLE16.SYSREC01'
OUTPUT SYSREC02 UNIT SYSDA
  SPACE (5,1)
  DSNAME 'ADU.EXAMPLE16.SYSREC02'
OUTPUT SYSREC03 UNIT SYSDA
  SPACE (3,1)
  DSNAME 'ADU.EXAMPLE16.SYSREC03'
OUTPUT SYSREC04 UNIT SYSDA
  SPACE (3,1)
  DSNAME 'ADU.EXAMPLE16.SYSREC04'
OUTPUT BACKUP UNIT SYSDA
  SPACE (5,1)
  DSNAME 'ADU.EXAMPLE16.BKUP.P&PART'
*
UNLOADON SYSREC,BACKUP ACTIVE(YES, YES)
ANALYZE DB2STATS
*
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.EXAMPLE16.SYSCNTL,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSPRINT DD SYSOUT=* 
```

Figure 56  SYSPRINT for example 16 (part 1 of 4)

```
***** BMC UNLOAD PLUS FOR DB2 V11R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50001I UTILITY EXECUTION STARTING   10/28/2013   13:15:32 ...
BMC50002I UTILITY ID = 'ADUXM16'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'ADU$OPTS'.
BMC50471I z/OS 2.1.0,PID=HBB7790,DFSMS FOR Z/OS=2.1.0,OB2=11.1.0
BMC50471I REGION=0M,BELOW 16M=8844K,ABOVE 16M=1415044K,IEFUSI=NO,CPUS=3
BMC50471I MEMLIMIT=17592186040320M,AVAILABLE=17592186040313M,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
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAIN: BP0661 BP0661 BP0661 BP0667 BP0671 BP0674 BP0675 BP0682 BP0686 BP0689 BP0697
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAIN: BP0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I MAIN: BP05409 BP05534 BP05674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I MAIN: BPE0401 BPE0403 BPE0405 BPE0407 BPE0410 BPE0412 BPE0416
BMC50471I ANALYZE=(DB2STATS,NOLIMIT) FDRCE_RPT=NO SMAX=16
BMC50471I CENTRY= (1950,2049) FORCE=NONE SMCRE=(OK,OK)
BMC50471I CHANGE_CONSISTENT=NO HISTORY=YES SORTNUM=32
BMC50471I CHANGE_QUSESC=NO IBUFFS=25 SOLDelay=3
BMC50471I CMAX=16 INLINE=NO SQLRETRY=100
```
### Example 16: Using dynamic allocation

#### Chapter 5 Examples of UNLOAD PLUS jobs

<table>
<thead>
<tr>
<th>BMC50471I</th>
<th>CRMAIDI=50</th>
<th>LOADDECP=NO</th>
<th>TAPIOIS=DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I</td>
<td>CONSTRULES=BMC</td>
<td>LOCKROW=YES</td>
<td>TASKMAX=200</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>CURRENTDEGREE=MINE</td>
<td>MAXP=5</td>
<td>UBUFFS=25</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DELFILEY=YES</td>
<td>MSGLEVEL=1</td>
<td>UNLDMAX=200</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DRNDELAY=1</td>
<td>NULLCHAR=X'6F'</td>
<td>UNLOADDN_ACTIVE=(YES,NO)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DRNRETRY=255</td>
<td>NULLTYPE=T1</td>
<td>UNLOADDN=(SYSREC,SYSRED)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DRNWAIT=NONE</td>
<td>OPNDB2ID=YES</td>
<td>USELRECL=NO</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DSPLOCKS=DRNFAIL</td>
<td>PLAN=ADUQA</td>
<td>UXSTATE=SUP</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>EXCLDUMP=(X37,X22,X06)</td>
<td>RECEM=AUTO</td>
<td>WORKUNIT=SYSALLDA</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>FILEREDWN=SYSREF</td>
<td>ROWSETSZ=100</td>
<td>ZIP=ENABLED</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>FILL=NO</td>
<td>SDUMP=YES</td>
<td>ZONEDDECOVP=(C,D)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>FORCE_AT=(START,3)</td>
<td>SHRLEVEL=REFERENCE</td>
<td></td>
</tr>
</tbody>
</table>

#### BMC50470I TAPES=NONE

<table>
<thead>
<tr>
<th>BMC50470I</th>
<th>OUTPUT = SYSREC</th>
<th>SYSRD</th>
<th>SYSREF</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50470I</td>
<td>UNIT = SYSALLDA</td>
<td>SYSRD</td>
<td>SYSALLDA</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>VOLCNT=25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGLIMIT = 5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGEMPTY = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGSCRAT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNITCNT = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SPACE = CYL</td>
<td>CYL</td>
<td>CYL</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DSNTYPE = NONE</td>
<td>NONE</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SYSCODE = SYSREC</td>
<td>VOLUMES=NONE</td>
<td>VOLUMES=NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SYSRED = SYSRD</td>
<td>VOLUMES=NONE</td>
<td>VOLUMES=NONE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SYSREF = SYSREF</td>
<td>VOLUMES=NONE</td>
<td>VOLUMES=NONE</td>
</tr>
</tbody>
</table>

#### BMC50483I SYSPRINT for example 16 (part 2 of 4)

<table>
<thead>
<tr>
<th>BMC50471I</th>
<th>DB2 DSNHDECP MODULE SETTINGS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I</td>
<td>VERSION = 1110</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>SUBSYSTEM DEFAULT = DEHJ</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>CHARACTER SET = ALPHANUM</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DATE FORMAT = USA</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DATE FORMAT = USA</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>LOCAL DATE LENGTH = 0</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>LOCAL TIME LENGTH = 0</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DECIMAL POINT = PERIOD</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DECIMAL ARITHMETIC = 15</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>DELIMITER = DEFAULT</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>SQL DELIMITER = DEFAULT</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>ENCODING SCHEME = EBCDIC</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>APPL ENCODING SCHEME = EBCDIC</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>MIXED = NO</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>EBCDIC CCSID = (37,65534,65534)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>ASCI CCSID = (819,65534,65534)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>UNICODE CCSID = (367,1208,1200)</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>IMPLICIT TIME Zone = CURRENT (-05:00)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BMC505028I</th>
<th>DB2 MODE = NFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50471I</td>
<td>BMC_BMCUTIL = 'BMCUTIL.CMN_BMCUTIL'</td>
</tr>
<tr>
<td>BMC50471I</td>
<td>BMC_BMCXCOPY = 'BMCUTIL.CMN_BMCXCOPY'</td>
</tr>
</tbody>
</table>

#### BMC50471I FORCE_AT=(START,3) SHRLEVEL=REFERENCE
Example 16: Using dynamic allocation

```sql
UNLOAD PLUS for DB2 Reference Manual

BMC50102I    OUTPUT SYSREC01 UNIT SYSDA
BMC50102I    SPACE (5,1)
BMC50102I    DSNAME 'ADU.EXAMPL16.SYSREC01'
BMC50102I    OUTPUT SYSREC02 UNIT SYSDA
BMC50102I    SPACE (5,1)
BMC50102I    DSNAME 'ADU.EXAMPL16.SYSREC02'
BMC50102I    OUTPUT SYSREC03 UNIT SYSDA
BMC50102I    SPACE (3,1)
BMC50102I    DSNAME 'ADU.EXAMPL16.SYSREC03'
BMC50102I    OUTPUT SYSREC04 UNIT SYSDA
BMC50102I    SPACE (3,1)
BMC50102I    DSNAME 'ADU.EXAMPL16.SYSREC04'
BMC50102I    OUTPUT BACKUP UNIT SYSDA
BMC50102I    SPACE (5,1)
BMC50102I    DSNAME 'ADU.EXAMPL16.BKUP.P001'
BMC50102I *
BMC50102I    UNLOADDN SYSREC,BACKUP ACTIVE(YES, YES)
BMC50102I    ANALYZE DB2STATS
BMC50102I *
BMC50102I    SELECT * INTO
BMC50102I         EMPL_ID DECIMAL EXTERNAL (5,0),
BMC50102I         ,
BMC50102I         ,
BMC50102I         ,
BMC50102I         JOB_CODE SMALLINT EXTERNAL (2,0),
BMC50102I         ,
BMC50102I         SALARY DECIMAL EXTERNAL (9,2),
BMC50102I         ,
BMC50102I         ,
BMC50102I         FROM BMC.EMPLS
BMC50102I *

BMC51654I DIRECT YES IN EFFECT
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 1 IS 8 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 2 IS 5 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 3 IS 5 FROM DB2STATS
BMC51687I ESTIMATED ROWS FOR TABLE SPACE BMCEXDB.BMCEXATS PART 4 IS 2 FROM DB2STATS
BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 20
BMC50046I UNLOAD PLUS DYNAMIC FILE ALLOCATION REPORT

BMC50446I DDNAME DNAME
BMC50447I UNIT OR DATACLAS MGMTCLAS STORCLAS KBYTES PRI KBYTES SEC ALOC PRI ALOC SEC
BMC51639I FOR DDNAME 'SYS00002' DSN='ADU.EXAMPL16.SYSREC01'
BMC51639I FOR DDNAME 'SYS00006' DSN='ADU.EXAMPL16.BKUP.P001'
BMC51639I FOR DDNAME 'SYS00003' DSN='ADU.EXAMPL16.SYSREC02'
BMC51639I FOR DDNAME 'SYS00007' DSN='ADU.EXAMPL16.BKUP.P002'
BMC51639I FOR DDNAME 'SYS00004' DSN='ADU.EXAMPL16.SYSREC03'
BMC51639I FOR DDNAME 'SYS00008' DSN='ADU.EXAMPL16.BKUP.P003'
BMC51639I FOR DDNAME 'SYS00005' DSN='ADU.EXAMPL16.SYSREC04'
BMC51639I FOR DDNAME 'SYS00009' DSN='ADU.EXAMPL16.BKUP.P004'

BMC50894I PAGE EXTERNALIZATION PROCESS STARTING AT 10/28/2013 13:15:33
BMC50650I ESTABLISHING A POINT OF CONSISTENCY FOR:
BMC50651I SPACE BMCEXDB.BMCEXATS PART(0) PSID(0002)
BMC50895I PAGE EXTERNALIZATION PROCESS COMPLETE. ELAPSED TIME = 00:00:00

Figure 56 SYSPRINT for example 16 (part 3 of 4)
Example 16: Using dynamic allocation

Figure 56  SYSPRINT for example 16 (part 4 of 4)

```
BMC50041I 0: ZIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM8
BMC50474I BELOW 16M = 8392K, ABOVE 16M = 1405476K, CPUS = 3
BMC51701I MAX TASKS = 4, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50477I 1: PARTITION = 1, ROWS/KEYS = 8, I/O WAITS = 2, DDNAME = SYS00011
BMC50477I 3: PARTITION = 3, ROWS/KEYS = 5, I/O WAITS = 2, DDNAME = SYS00013
BMC50478I 1: ROB LOCK WAITS = 0
BMC50477I 4: PARTITION = 4, ROWS/KEYS = 2, I/O WAITS = 2, DDNAME = SYS00013
BMC50478I 4: ROB LOCK WAITS = 0
BMC50478I 2: PARTITION = 2, ROWS/KEYS = 5, I/O WAITS = 2, DDNAME = SYS00012
BMC50478I 2: ROB LOCK WAITS = 0
BMC50476I DDNAME = SYS00002+SYS00006, I/Os = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYS00003+SYS00007, I/Os = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYS00004+SYS00008, I/Os = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50476I DDNAME = SYS00005+SYS00009, I/Os = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC51671I UNLOAD STATISTICS: 8 ROWS PROCESSED FROM PARTITION 1
BMC51666I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A001' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 2
BMC51666I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A002' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 5 ROWS PROCESSED FROM PARTITION 3
BMC51666I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A003' READ 3 PAGES
BMC51671I UNLOAD STATISTICS: 2 ROWS PROCESSED FROM PARTITION 4
BMC51666I UNLOADING OF DATASET 'DEHJCAT.DSNDBD.BMCEXDB.BMCEXATS.I0001.A004' READ 3 PAGES
BMC51672I UNLOAD STATISTICS: 20 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXATS', 0 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 8 RECORDS WRITTEN TO DDNAME 'SYS00002'
BMC51674I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYS00003'
BMC51674I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYS00004'
BMC51674I UNLOAD STATISTICS: 5 RECORDS WRITTEN TO DDNAME 'SYS00005'
BMC51674I UNLOAD STATISTICS: 2 RECORDS WRITTEN TO DDNAME 'SYS00006'
BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50041I 0: ZIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM8
BMC50006I UTILITY EXECUTION COMPLETE, RETURN CODE = 0
BMC50476I DDNAME = SYSCNTL, I/Os = 1, I/O WAITS = 1, ROB LOCK WAITS = 0
BMC50006I UTILITY EXECUTION COMPLETE, RETURN CODE = 0
```

Example 17: Changing overpunch values in zoned decimal data

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).

<table>
<thead>
<tr>
<th>Source Value</th>
<th>Target</th>
<th>SYSREC value with overpunch C,D (default)</th>
<th>SYSREC value with overpunch F,F</th>
</tr>
</thead>
<tbody>
<tr>
<td>21111</td>
<td>COL1 DECIMAL ZONED (12)</td>
<td>'F0F0F0F0F0F0F2F1F1C1'</td>
<td>'F0F0F0F0F0F0F2F1F1F1F1'</td>
</tr>
<tr>
<td>-21117</td>
<td>COL1 DECIMAL ZONED (12)</td>
<td>'F0F0F0F0F0F0F2F1F1D7'</td>
<td>'F0F0F0F0F0F0F2F1F1F1F7'</td>
</tr>
</tbody>
</table>

In this example, any data that is in zoned decimal format is assigned an overpunch value during the unload process.

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 81 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>FORMAT EXTERNAL</td>
<td>tells UNLOAD PLUS to use external format for all output data types</td>
</tr>
<tr>
<td>UNLOADADDN 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</td>
</tr>
</tbody>
</table>

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.
Example 17: Changing overpunch values in zoned decimal data

Table 81  Command options for example 17 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

Figure 57 shows the JCL for example 17.

Figure 57  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)
FIXEDVARCHAR 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_TBEXM17
/*
//SYSREC DD DSN=ADU.EXAMPLE17.SYSREC,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,(3,1),RLSE)
//SYSCNTL 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

Figure 58 shows the SYSPRINT for example 17.

Figure 58 SYSPRINT for example 17 (part 1 of 3)
Figure 58  SYSPRINT for example 17 (part 2 of 3)

```sql
BMC50471I DB2 DSNHDECP MODULE SETTINGS:
BMC50471I VERSION                 = 1110
BMC50471I SUBSYSTEM DEFAULT       = DEHJ
BMC50471I CHARACTER SET           = ALPHANUM
BMC50471I DATE FORMAT             = USA
BMC50471I TIME FORMAT             = USA
BMC50471I LOCAL DATE LENGTH       = 0
BMC50471I LOCAL TIME LENGTH       = 0
BMC50471I DECIMAL POINT           = PERIOD
BMC50471I DECIMAL ARITHMETIC      = 15
BMC50471I DELIMITER               = DEFAULT
BMC50471I SQL DELIMITER           = 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 UNICODE CCSID           = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE      = CURRENT (-05:00)
BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL       = 'BMCUTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC       = 'BMCUTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST       = 'BMCUTIL.CMN_BMCHIST'
BMC50471I BMC_BMCXCOPY      = 'BMCUTIL.CMN_BMCXCOPY'

BMC50102I UNLOAD FORMAT EXTERNAL
BMC50102I     UNLOADDN SYSREC ACTIVE NO
BMC50102I     ZONEDDECOVP(F,F)
BMC50102I     FIXEDVARCHAR YES
BMC50102I     CNTLCARDS 'REPLACE COPYDDN (SYSCPY1)'
BMC50102I     SELECT *
BMC50102I     INTO
BMC50102I     ,COL1 DECIMAL ZONED (12)
BMC50102I     ,COL2 VARCHAR (10)
BMC50102I     ,COL3 DECIMAL ZONED (12)
BMC50102I     ,COL4 VARCHAR (10)
BMC50102I     ,COL5 DECIMAL ZONED (12)
BMC50102I     ,COL6 VARCHAR (10)
BMC50102I     ,COL7 DECIMAL ZONED (12)
BMC50102I     ,COL8 VARCHAR (10)
BMC50102I     ,COL9 DECIMAL ZONED (12)
BMC50102I     ,COL10 VARCHAR (10)
BMC50102I     ,COL11 DECIMAL ZONED (12)
BMC50102I     ,COL12 VARCHAR (10)
BMC50102I     FROM BMC.LONGNAME_TABLE_ON_EXAMPLE17_TBEXMP17

BMC51654I DIRECT YES IN Effect
BMC51687I ESTIMATED ROWS FOR TABLE SPACE DBEXMP17.TSEXMP17 PART 0 IS 6230 FROM HURBA
BMC51688I ESTIMATED ROWS FOR TABLE BMC.LONGNAME_TABLE_ON_EXAMPLE17_TBEXMP17 IS 6230 FROM TABLESPACE
BMC51689I ESTIMATED ROWS FOR SELECT 1 IS 6230
BMC50004I UTILINIT PHASE COMPLETE. ELAPSED TIME = 00:00:00
BMC51693I FOR DDNAME 'SYSREC' DSN=ADU.EXAMPLE17.SYSREC,DCB=(RECFM=VB,BLKSIZ=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(0002)
BMC50895I PAGE EXTERNALIZATION PROCESS COMPLETE. ELAPSED TIME = 00:00:00
BMC50041I 0: ZIIP ENABLED (0) USING XBM SUBSYSTEM XBMB
BMC50474I BELOW 16M = 8492K, ABOVE 16M = 1406204K, CPUS = 3
BMC51703I MAX TASKS = 1, MAX PARTITIONS PER TASK = 1, SORTWKS PER TASK = 0, MAX OPEN PARTITIONS PER TASK = 1
BMC50471I 1: PARTITION = 0, ROWS/KEYS = 3, I/O WAITS = 2, DONAME = SYS00002
BMC50478I 1: ROB LOCK WAITS = 0
BMC50476I DONAME = SYSSRC, 1/O = 1, I/O WAITS = 1, ROB LOCK WAITS = 0
BMC51686I UNLOAD TOTAL OF DATASET 'BEWICAT.DSNDBD,DBEXMP17.TSEXMP17.I0001.A001' READ 3 PAGES
BMC51672I UNLOAD STATISTICS: 3 ROWS PROCESSED FROM SPACE 'DBEXMP17.TSEXMP17', 0 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 3 RECORDS WRITTEN TO DONAME 'SYSREC'
```
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.
Example 18: Unloading a table space and unloading data as Unicode

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.

Table 82 describes the UNLOAD PLUS key command options for this job.

Table 82  UNLOAD PLUS command options for example 18

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
</table>
| UNLOAD TABLESPACE           | unloads table space BMCEXDB.BMCEXBTS and generates SELECT statements for the following tables (see message BMC51657I and BMC51658I in the SYSPRINT output):
  |   | ■ BMC.RET_EMPLS
  |   | ■ BMC.HRLY_EMPLS
  |   | ■ BMC.SAL_EMPLS
| UNICODE                     | tells UNLOAD PLUS to use the DB2 installation default (DSNHDECP) CCSIDs to encode the unloaded data in Unicode
| AUTOTAG YES                 | precedes each output record with a unique four-byte value for each generated SELECT statement
| CNTLCARDS REPLACE           | tells UNLOAD PLUS to write DB2 LOAD control statements
  |   | ‘REPLACE’ is added to the DB2 LOAD control statement.
| UNLOADDN SYSREC ACTIVE NO   | disables dynamic allocation for the primary unload data set, overriding the default in the installation options module

Table 83 describes the LOADPLUS key command options for this job.

Table 83  LOADPLUS command options for example 18

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
</table>
| REPLACE                     | tells LOADPLUS to delete the existing data in the tables before loading
| UNICODE                     | specifies that the data that you are loading from the SYSREC data set is in Unicode format
| CCSID                       | specifies the three CCSIDs used to encode the data coming from UNLOAD PLUS
| INTO TABLE                  | identifies the data to load by specifying the table name
| WHEN                        | specifies the condition that must be true for LOADPLUS to select a row
Example 18: Unloading a table space and unloading data as Unicode

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.

Figure 59 shows the UNLOAD PLUS JCL for example 18.

Figure 59  UNLOAD PLUS JCL for example 18

```
// 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.SYSREC,
// DD DISP=(NEW,CATLG),SPACE=(TRK,(5,1)),UNIT=SYSDA
//SYSCTNL DD DSN=ADU.EXAML18.UNICODE.SYSCNTL,
// DD DISP=(NEW,CATLG),SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSIN DD   *
* UNLOAD
TABLESPACE BMCEXDB.BMCEXBTS
UNICODE
AUTOTAG YES
CNTLCARDS 'REPLACE'
UNLOAD DN SYSREC ACTIVE NO
```

Figure 60 shows the UNLOAD PLUS SYSPRINT for example 18.

Figure 60  UNLOAD PLUS SYSPRINT for example 18 (part 1 of 4)

```
***** BMC UNLOAD PLUS FOR DB2 VI1R1.00 *****
(C) COPYRIGHT 1991 - 2013 BMC SOFTWARE, INC.
BMC50001I UTILITY EXECUTION STARTING   10/28/2013   13:18:16 ...
BMC50021I UTILITY ID = 'ADUXM18A'. DB2 SUBSYSTEM ID = 'DEHJ'. OPTION MODULE = 'ADU$OPTS'.
BMC50471I   No maintenance to report
```
Example 18: Unloading a table space and unloading data as Unicode

BMC50471I DRNRETRY=255                                NULLTYPE=T1                             UNLOADDN=(SYSREC, SYSRED)
BMC50471I DRNWAIT=NONE                                 OPNOB2ID=YES                            USELRECL=NO
BMC50471I DLSPLOCKS=ODRFAIL                           PLAN=ADVDA                               UXSTATE=SUP
BMC50471I EXCLUDUMP=('X37', 'X22', 'X06')               RECFM=AUTO                               WORKUNIT=SYSALLDA
BMC50471I FILEREFER=SYSREF                             ROWSETSZ=100                            ZIIP=ENABLED
BMC50471I FILL=NO                                      SDUMP=YES                               ZONEDDECOVP=(C, D)
BMC50471I FORCE_AT=(START, 3)                         SHRLEVEL=REFERENCE
BMC50471I TAPES=NONE

BMC50470I OUTPUT = SYSREC                             SYSRED                               SYSREF
BMC50470I UNIT = SYSALLDA                             SYSALLDA                             SYSALLDA
BMC50470I VOLCNT = 25                                 25                                   25
BMC50470I GDGLIMIT = 5                                  5                                    5
BMC50470I GDGEMPTY = NO                                NO                                   NO
BMC50470I GENDSR = NO                                  NO                                   NO
BMC50470I SDUMP = YES                                   0                                    0
BMC50470I SPACE = CYL                                  CYL                                  CYL
BMC50470I DISKRETN = NONE                               NONE                                 NONE
BMC50470I DISKEXPD = NONE                               NONE                                 NONE
BMC50470I RETPD = NONE                                  NONE                                 NONE
BMC50470I EXPDT = 99000                                99000                                99000
BMC50470I TRTCH = NONE                                  NONE                                 NONE
BMC50470I DSNTYPE = NONE                                NONE                                 PDS

Figure 60  UNLOAD PLUS SYSPRINT for example 18 (part 2 of 4)

BMC50471I DB2 DSNHDECL MODULE SETTINGS:
BMC50471I VERSION  = 1110
BMC50471I SUBSYSTEM DEFAULT  = DEHJ
BMC50471I CHARACTER SET  = ALPHANUM
BMC50471I DATE FORMAT  = USA
BMC50471I TIME FORMAT  = USA
BMC50471I LOCAL DATE LENGTH  = 0
BMC50471I LOCAL TIME LENGTH  = 0
BMC50471I DECIMAL POINT  = PERIOD
BMC50471I DECIMAL ARITHMETIC  = BLEND
BMC50471I DELIMITER  = DEFAULT
BMC50471I SQL DELIMITER  = 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 UNICODE CCSID  = (367, 1208, 1200)
BMC50471I IMPLICIT TIME ZONE  = CURRENT (-05:00)
BMC50470I *
BMC50470I UNLOAD
BMC50470I TABLESPACE BMCEXDB.BMCEXBTS
BMC50470I UNICODE
BMC50470I AUTOTAG YES
BMC50470I CNTLCARDS 'REPLACE'
BMC50470I UNLOADDN SYSREC ACTIVE NO
Example 18: Unloading a table space and unloading data as Unicode

Figure 60  UNLOAD PLUS SYSPRINT for example 18 (part 3 of 4)
Example 18: Unloading a table space and unloading data as Unicode

Figure 60  UNLOAD PLUS SYSPRINT for example 18 (part 4 of 4)

Figure 61 shows the LOADPLUS JCL for example 18.

Figure 61  LOADPLUS JCL for example 18

Figure 62 on page 402 shows the LOADPLUS SYSPRINT for example 18.
Figure 62  LOADPLUS SYSPRINT for example 18 (part 1 of 6)

**** BMC LOADPLUS FOR DB2 V11R1.00 ****
(C) COPYRIGHT 1990 - 2013 BMC SOFTWARE, INC.
LOADPLUS TECHNOLOGY IS PROTECTED BY U.S. PATENT NUMBER 7,664,790

BMC50001I UTILITY EXECUTION STARTING 10/28/2013 14:17:09 ...
BMC50002I UTILITY ID = 'ADUXM18B'. DB2 SUBSYSTEM ID = 'DEHJ'. OPTION MODULE = 'AMU$OPTS'.
BMC50471I RCSYS:V11.01.00 DB2=11.1.0
BMC50471I REGION=OM,BELOW 16M=8816K,ABOVE 16M=1414436K,IEFUSI=NO,CPUS=3
BMC50471I MEMLIMIT=1759218604320M,AVAILABLE=17592186043138M,MEMLIMIT SET BY:REGION=0

BMC50471I LOADPLUS FOR DB2--V11.01.00
BMC50471I NO MAINTENANCE TO REPORT
BMC50471I DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I NO MAINTENANCE TO REPORT
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT: BPJ661 BPJ667 BPJ667 BPJ667 BPJ667 BPJ667 BPJ668 BPJ668 BPJ668 BPJ668 BPJ669
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAINT: BPJ661
BMC50471I BM C STATE API--V11.01.00
BMC50471I MAINT: BPUS409 BPUS534 BPUS674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
BMC50471I MAINT: BPED401 BPED403 BPED405 BPED407 BPED410 BPED412 BPED416
BMC50471I HIGH-SPEED APPLY ENGINE--V11.01.00
BMC50471I MAINT: BPUS559 BPUS548 BPUS672 BPUS626

BMC50471I APCCOMMIT=2500
BMC50471I APDOPTS= IBUFFS=Z0 SDUMP=YES
BMC50471I APMULTROW=100 IDCACHE=1000 SHORTMEMORY=CONTINUE
BMC50471I APMAXAGT=10 DDCOMD=SYSDICIN SHLEVEL=(NONE,NONE)
BMC50471I APRETLMOUNT=IDERROR=DISCARD SMAX=16
BMC50471I APRETVLS=INDON=SYSREC SMCORE=(OK,OK)
BMC50471I AUTOENMROWS=YES INFOI=NO SORDERB='(SYSALLDA)
BMC50471I AVAILPAGECT=0 INLINEC=NONE SORTNUM=32
BMC50471I CBUFFS=30 IXNEX=NONE SODELAY=3
BMC50471I CENTURY=(1950,2049) IXRANDOM=NO SOLRETRY=100
BMC50471I CHECKPNO=YES KEEPDICTIONARY=NO STOPDUMP=YES
BMC50471I COPYDUM=(BMC,C,SYMBMC) LBUFFS=Z0 STOPDELAY=1
BMC50471I COPYVLY=FALL LOADDUM=SORTOUT STOPPRTY=300
BMC50471I COPYPEND=YES LOAD=NONE TAPEPRT=DELETE
BMC50471I COPYSUBSET=NO LOBAGPCT=50 TEMPDATA=YES
BMC50471I DELFILES=NO LOCKROW=YES TOTALPAGEPCT=30
BMC50471I DISCAGRMRC=8 LONGNAMELEN=MIDDLE TSSAMPLEPCT=100
BMC50471I DISCARDCRC=0 MADCOMD=SYSMAP UNIQUEO=YES
BMC50471I DISCOMD=SYSDISC MAXP=5 UPDMAMA_AUTHID=USER
BMC50471I DPODELAY=1 MAXSORTMEMOY=0 UPDMAA=NO
BMC50471I DRENDRTO=95 MAXTAME=3 UTILB_COLCCSID=UTILB
BMC50471I DRNWAIT=NONE MGEXTENT=CONTINUE UTILB_NULLIX=UTILB
BMC50471I DSNEXIT=NONE,ASM() MINSORTMEMOY=0 UXSTATE=SUP
BMC50471I DSNUTIL=NO MSGLEVEL=0 WRBUFFS=(20,10)
BMC50471I ENFORCE=CHECK OMPORDID=NONE WRHANMM=STAT1
BMC50471I ERRORM=SYSSER ORIGDISP=DELETE WORKUNIT=SYSALLDA
BMC50471I EXCLUD=(X37,X22,X06) PAUSEDISCARDCRC=4 XBLK=10
BMC50471I FASTSMITCH=NO PLAN=AMUQA XML=YES
BMC50471I FILECHAR=WARN PREFORMAT=NO XMLAVGSIZE=10240
BMC50471I FILECHAR=Y EPRALOAD=NO ZEROROWRC=0
BMC50471I FORCE_AT=(START,3) RCYON=(BMCRCY,BMCRCZ) ZIIP=DISABLED
BMC50471I FORCE_RPT=NO REDEFINE=YES
BMC50471I FORCE=NONE RENMMAX=30
BMC50471I APOWNER=
BMC50471I FORT%=NONE RENMMAX=30
BMC50471I FORCE=NONE RENMMAX=30
BMC50471I FORCE_RPT=NO REDEFINE=YES
BMC50471I FORCE=NONE RENMMAX=30

BMC50471I DTYPE = LOAD WORK SRTWORK
BMC50471I ACTIVE = YES YES NO
BMC50471I IFALLOC = USE USE USE
BMC50471I SMS = NO NO NO
BMC50471I SMSUNIT = NO NO NO
BMC50471I SIZECT = (100,100) (100,100) (100,100)
BMC50471I UNIT = (SYSALLDA,SYSDALLA) (SYSALLDA,SYSDALLA) (SYSALLDA,SYSDALLA)
BMC50471I UNITCTN = (0,0) (0,0) N/A
BMC50471I VOLCNT = (25,25) (25,25) N/A
BMC50471I AVGWLP = ((30000,TRK),(30000,TRK)) (30000,TRK) N/A
BMC50471I OSTYPE = (NONE,NONE) (NONE,NONE) (NONE,NONE)
BMC50471I DATACLS = (NONE,NONE) (NONE,NONE) (NONE,NONE)
BMC50471I MGMTCLS = (NONE,NONE) (NONE,NONE) (NONE,NONE)
BMC50471I STORCLS = (NONE,NONE) (NONE,NONE) (NONE,NONE)
### LOADPLUS SYSPRINT for example 18 (part 2 of 6)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DDTYPE</strong></td>
<td>LOCBFCPY</td>
</tr>
<tr>
<td><strong>ACTIVE</strong></td>
<td>USE</td>
</tr>
<tr>
<td><strong>IFALLOC</strong></td>
<td>USE</td>
</tr>
<tr>
<td><strong>SMS</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>SMSUNIT</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>SMS</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>IFALLOC</strong></td>
<td>USE</td>
</tr>
<tr>
<td><strong>SMS</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>SMSUNIT</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>SMS</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>SIZEPCT</strong></td>
<td>(100,100)</td>
</tr>
<tr>
<td><strong>SIZEPCT</strong></td>
<td>(100,100)</td>
</tr>
<tr>
<td><strong>SIZEPCT</strong></td>
<td>(100,100)</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>DSNTYPE</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DSNTYPE</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DSNTYPE</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DATACLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DATACLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DATACLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>MGMTCLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>MGMTCLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>MGMTCLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DATACLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DATACLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DATACLAS</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DSNTYPE</strong></td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td><strong>DSNUNIT</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>DSNUNIT</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>DSNUNIT</strong></td>
<td>NO</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>THRESHLD</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>THRESHLD</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>THRESHLD</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>AVGWOLSP</strong></td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALDA</td>
</tr>
<tr>
<td><strong>UNIT</strong></td>
<td>SYSALLDA</td>
</tr>
<tr>
<td><strong>THRESHLD</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>THRESHLD</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>THRESHLD</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

**Example 18: Unloading a table space and unloading data as Unicode**

Chapter 5 Examples of UNLOAD PLUS jobs 403
Example 18: Unloading a table space and unloading data as Unicode

Figure 62  LOADPLUS SYSPRINT for example 18 (part 3 of 6)

<table>
<thead>
<tr>
<th>BMC50470I</th>
<th>IFALLOC = USE</th>
<th>USE</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50470I</td>
<td>SMS = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SMSUNIT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SIZEPCT = (100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNIT = (SYSALLDA,SYSALLDA)</td>
<td>(SYSALLDA,SYSALLDA)</td>
<td>(SYSALLDA,SYSALLDA)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNITCNT = (0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>AVGVOLSP = ((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DATACLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DATAUNIT = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>THRESHLD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>MAXEXTSZ = ((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>EXPDT =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>RETPD =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGLIMIT = 5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGEMPTY = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DDTYPE = LOCBXCPY</td>
<td>REMPXCPY</td>
<td>REMBXCPY</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>ACTIVE = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>IFALLOC = USE</td>
<td>USE</td>
<td>USE</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SMS = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SMSUNIT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SIZEPCT = (100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNIT = (SYSALLDA,SYSALLDA)</td>
<td>(SYSALLDA,SYSALLDA)</td>
<td>(SYSALLDA,SYSALLDA)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>UNITCNT = (0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>AVGVOLSP = ((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DATACLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DATAUNIT = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>THRESHLD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>MAXEXTSZ = ((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>EXPDT =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>RETPD =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGLIMIT = 5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>GDGEMPTY = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DDTYPE = LOCBXCPY</td>
<td>REMPXCPY</td>
<td>REMBXCPY</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>ACTIVE = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOAD DSNPAT=UID..BMC..ATS..&amp;DDNAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>WORK DSNPAT=UID..BMC..ATS..&amp;DDNAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>DISCARD DSNPAT=UID..BMC..ATS..&amp;DDNAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>SYSMAP DSNPAT=UID..BMC..ATS..&amp;DDNAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCPXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCBXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMPXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMBXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCPFCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCBFCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMPFCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMBFCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCPLCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCBLCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMPLCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMBLCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCPXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCBXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMPXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMBXCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCPLCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>LOCLTCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMPLTCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I</td>
<td>REMLTCPY DSNPAT=UID..BMC..ATS..&amp;DDNAME..&amp;PART..&amp;TIME</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

404  UNLOAD PLUS for DB2 Reference Manual
Figure 62  LOADPLUS SYSPRINT for example 18 (part 4 of 6)
Example 18: Unloading a table space and unloading data as Unicode

Figure 62  LOADPLUS SYSPRINT for example 18 (part 5 of 6)
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 BMCUNLOAD 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 SYSPRT, the LOADPLUS JCL and SYSPRT, and output showing the loaded table.

Table 84 describes the UNLOAD PLUS key command options for this job.

**Table 84  UNLOAD PLUS command options for example 19 (part 1 of 2)**

<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>
</tbody>
</table>

The control cards that are specified with this option will generate a LOAD REPLACE run with dynamic allocation.

**Note:** When you specify FORMAT BMCLOAD in UNLOAD PLUS, UNLOAD PLUS forces CNTLCARDS BMCLOAD, here overriding the default of DB2LOAD.
Example 19: Unloading data to LOADPLUS

Table 84  UNLOAD PLUS command options for example 19 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<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</td>
</tr>
<tr>
<td></td>
<td>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>
</tbody>
</table>

The UNLOAD PLUS SYSPRINT (Figure 64 on page 410) shows the LOADPLUS control cards that UNLOAD PLUS writes to the SYSCNTL data set.

Table 85 describes the key command options for loading the data using LOADPLUS.

Table 85  LOADPLUS command options for example 19 (part 1 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDSN</td>
<td>names the SYSREC data set that UNLOAD PLUS created</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>This example shows that UNLOAD PLUS calculated the number of new rows that LOADPLUS will load as 29</td>
</tr>
<tr>
<td></td>
<td>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>DELETEFILES 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>
</tbody>
</table>
Example 19: Unloading data to LOADPLUS

Chapter 5 Examples of UNLOAD PLUS jobs

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.

Figure 63 shows the UNLOAD PLUS JCL for example 19.

**Table 85** LOADPLUS command options for example 19 (part 2 of 2)

<table>
<thead>
<tr>
<th>Command options used in JCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>INTO TABLE</td>
<td>identifies the table to load. This example contains the WHEN TABLE=obid option, which is required when using FORMAT BMCUNLOAD. The OBID is the identifier that UNLOAD PLUS 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.

Figure 63 shows the UNLOAD PLUS JCL for example 19.

**Figure 63** UNLOAD PLUS JCL for example 19

```sql
//        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,
//                     DD 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
```
Figure 64 shows the UNLOAD PLUS SYSPRINT output for example 19.
Example 19: Unloading data to LOADPLUS

Chapter 5 Examples of UNLOAD PLUS jobs

Figure 64  UNLOAD PLUS SYSPRINT for example 19 (part 2 of 3)
Example 19: Unloading data to LOADPLUS

Figure 64  UNLOAD PLUS SYSPRINT for example 19 (part 3 of 3)

```plaintext
BMC51672I UNLOAD STATISTICS: 20 ROWS PROCESSED FROM SPACE 'BMCEXDB.BMCEXBTS', 0 NOT SELECTED, 0 DISCARDED
BMC51674I UNLOAD STATISTICS: 29 RECORDS WRITTEN TO DDNAME 'SYS00008'
BMC51675I UNLOAD STATISTICS: 0 RECORDS DISCARDED DUE TO ERRORS
BMC50441I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM5
BMC50004I UNLOAD PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC51639I FOR DDNAME 'SYSCNTL' DSN=ADU.EXAMPL19.SYSCNTL,DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)
BMC51801I LOAD TABLE STATEMENTS WRITTEN TO DDNAME 'SYSCNTL'
BMC51936I LOAD DATA INDSN ('ADU.EXAMPL19.SYSREC'
BMC51809I REPLACE
BMC51809I DELETEFILES YES SYSDISC YES
BMC51955I      FORMAT BMCUNLOAD
BMC51811I   INTO TABLE
BMC51809I BMC.TBEXB19
BMC51956I        WHEN TABLE=1

BMC50041I 0: ZIIP NOT ENABLED (0) USING XBM SUBSYSTEM XBM5
BMC50004I UNLOAD PHASE COMPLETE. ELAPSED TIME = 00:00:00

BMC50476I DDNAME = SYSCNTL, I/OS = 1, I/O WAITS = 1, RDB LOCK WAITS = 0
BMC50006I UTILITY EXECUTION COMPLETE, RETURN CODE = 0

*****  B M C   L O A D P L U S   F O R   D B 2    V11R1.00  *****
(C) COPYRIGHT 1990 - 2013 BMC SOFTWARE, INC.
LOADPLUS TECHNOLOGY IS PROTECTED BY U.S. PATENT NUMBER 7,664,790
BMC50001I UTILITY EXECUTION STARTING   10/28/2013   13:21:31 ...
BMC50002I UTILITY ID = 'ADUXM19'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'AMU$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=8836K,ABOVE 16M=1412296K,IEFUSI=NO,CPUS=3
BMC50471I MEMLIMIT=17592186040320M,AVAILABLE=17592186040311M,MEMLIMIT SET BY:REGION=0
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I   MAINT: BPJ0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I   MAINT: BPU5409  BPU5534  BPU5674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
```

Figure 65 shows the LOADPLUS JCL for example 19.

```plaintext
// JOB
// LOADEX19 EXEC PGM=AMUUMAIN,
//   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=*
// SYSOUT   DD SYSOUT=*
// UTPRINT  DD SYSOUT=*
// SYSUDUMP DD DSN=ADU.EXAMPL19.SYSERR,
//            DISP=(NEW,CATLG),SPACE=(CYL,(5,2)),
//            UNIT=SYSDA
// SYSIN    DD DISP=SHR,DSN=ADU.EXAMPL19.SYSCNTL
```

Figure 66 shows the LOADPLUS SYSPRINT for example 19.

```plaintext
**** J O B **** UNLOAD PLUS FOR DB2 V11R1.00 ****
BMC50001I UTILITY EXECUTION STARTING   10/28/2013   13:21:33 ...
BMC50002I UTILITY ID = 'ADUXM19'.  DB2 SUBSYSTEM ID = 'DEHJ'.  OPTION MODULE = 'AMU$OPTS'.
BMC50471I z/OS 2.1.0,PID=HBB7790,DFSMS FOR Z/OS=2.1.0,B2=11.1.0
BMC50471I REGION=0M,BELOW 16M=8836K,ABOVE 16M=1412296K,IEFUSI=NO,CPUS=3
BMC50471I MEMLIMIT=17592186040320M,AVAILABLE=17592186040311M,MEMLIMIT SET BY:REGION=0
BMC50471I LOADPLUS FOR DB2--V11.01.00
BMC50471I NO MAINTENANCE TO REPORT
BMC50471I DB2 UTILITIES COMMON CODE--V11.01.00
BMC50471I NO MAINTENANCE TO REPORT
BMC50471I SOLUTION COMMON CODE--V11.01.00
BMC50471I MAINT: BPJ0661  BPJ0667  BJ0671  BJ0674  BJ0675  BJ0676  BJ0682  BJ0686  BJ0689  BJ0697
BMC50471I BMCSORT ENGINE--V02.04.01
BMC50471I MAINT: BPJ0691
BMC50471I BMC STATS API--V11.01.00
BMC50471I MAINT: BPUS408  BPUS534  BPUS674
BMC50471I EXTENDED BUFFER MANAGER--V06.01.00
```

Figure 65  LOADPLUS JCL for example 19

Figure 66  LOADPLUS SYSPRINT for example 19 (part 1 of 6)
Chapter 5  Examples of UNLOAD PLUS jobs 413

Example 19: Unloading data to LOADPLUS

Figure 66  LOADPLUS SYSPRINT for example 19 (part 2 of 6)
### Example 19: Unloading data to LOADPLUS

**Figure 66 LOADPLUS SYSPRINT for example 19 (part 3 of 6)**

<table>
<thead>
<tr>
<th>BMC50470I STORAGE = (NONE,NONE)</th>
<th>(NONE,NONE)</th>
<th>(NONE,NONE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC50470I THRESHLD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I MAXEXTSZ = ((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
</tr>
<tr>
<td>BMC50470I EXPDT =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I RETPD =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I GDGLIMIT = N/A</td>
<td>N/A</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I GDGEMPTY = N/A</td>
<td>N/A</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I GDGSCRAT = N/A</td>
<td>N/A</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I DTYPE = LDCBFCPY</td>
<td>REMBCPY</td>
<td>REMBCPY</td>
</tr>
<tr>
<td>BMC50470I ACTIVE = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I FILELOC = USE</td>
<td>USE</td>
<td>USE</td>
</tr>
<tr>
<td>BMC50470I SMS = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I SMSUNIT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I SIZEPCT = (100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
</tr>
<tr>
<td>BMC50470I UNIT = (SYSALDA,SYSAUX)</td>
<td>(SYSALDA,SYSAUX)</td>
<td>(SYSALDA,SYSAUX)</td>
</tr>
<tr>
<td>BMC50470I UNITCNT = (0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMC50470I AVGEXTSP = ((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td>BMC50470I DSTYPE = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I DATACLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I MGMTCLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I STORCLASS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I THRESHLD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I MAXEXTSZ = ((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
</tr>
<tr>
<td>BMC50470I EXPDT =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I RETPD =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I GDGLIMIT = N/A</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I GDGEMPTY = N/A</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I GDGSCRAT = N/A</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I DTYPE = SYSMAP</td>
<td>LDCPLCPY</td>
<td>LDCPLCPY</td>
</tr>
<tr>
<td>BMC50470I ACTIVE = YES</td>
<td>YES</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I FILELOC = USE</td>
<td>USE</td>
<td>USE</td>
</tr>
<tr>
<td>BMC50470I SMS = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I SMSUNIT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I SIZEPCT = (100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
</tr>
<tr>
<td>BMC50470I UNIT = (SYSALDA,SYSAUX)</td>
<td>(SYSALDA,SYSAUX)</td>
<td>(SYSALDA,SYSAUX)</td>
</tr>
<tr>
<td>BMC50470I UNITCNT = (0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMC50470I AVGEXTSP = ((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td>BMC50470I DSTYPE = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I DATACLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I MGMTCLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I STORCLASS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I THRESHLD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I MAXEXTSZ = ((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
</tr>
<tr>
<td>BMC50470I EXPDT =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I RETPD =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I GDGLIMIT = N/A</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I GDGEMPTY = N/A</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I GDGSCRAT = N/A</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I DTYPE = LDCBFCPY</td>
<td>REMBCPY</td>
<td>REMBCPY</td>
</tr>
<tr>
<td>BMC50470I ACTIVE = NO</td>
<td>ND</td>
<td>YES</td>
</tr>
<tr>
<td>BMC50470I FILELOC = USE</td>
<td>USE</td>
<td>USE</td>
</tr>
<tr>
<td>BMC50470I SMS = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I SMSUNIT = NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BMC50470I SIZEPCT = (100,100)</td>
<td>(100,100)</td>
<td>(100,100)</td>
</tr>
<tr>
<td>BMC50470I UNIT = (SYSALDA,SYSAUX)</td>
<td>(SYSALDA,SYSAUX)</td>
<td>(SYSALDA,SYSAUX)</td>
</tr>
<tr>
<td>BMC50470I UNITCNT = (0,0)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>BMC50470I AVGEXTSP = ((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
<td>((30000,TRK),(30000,TRK))</td>
</tr>
<tr>
<td>BMC50470I DSTYPE = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I DATACLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I MGMTCLAS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I STORCLASS = (NONE,NONE)</td>
<td>(NONE,NONE)</td>
<td>(NONE,NONE)</td>
</tr>
<tr>
<td>BMC50470I THRESHLD = 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BMC50470I MAXEXTSZ = ((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
<td>((0,K),(0,K))</td>
</tr>
<tr>
<td>BMC50470I EXPDT =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I RETPD =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC50470I GDGLIMIT = 5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BMC50470I GDGEMPTY = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I GDGSCRAT = NO</td>
<td>ND</td>
<td>ND</td>
</tr>
<tr>
<td>BMC50470I DTYPE = LDCBFCPY</td>
<td>REMBCPY</td>
<td>REMBCPY</td>
</tr>
</tbody>
</table>

*UNLOAD PLUS for DB2 Reference Manual*
Example 19: Unloading data to LOADPLUS

Chapter 5 Examples of UNLOAD PLUS jobs

BMC50470I ACTIVE = NO
BMC50470I IFALLOC = USE
BMC50470I SMS = NO
BMC50470I SMSUNIT = NO
BMC50470I SIZEPCT = (100,100)
BMC50470I UNIT = (SYSALLDA,SYSDALLDA)
BMC50470I UNITCNT = (0,0)
BMC50470I VOLCNT = (25,25)
BMC50470I AVGVOLSP = ((30000,TRK),(30000,TRK))
BMC50470I DSNTYPE = (NONE,NONE)
BMC50470I DATACLAS = (NONE,NONE)
BMC50470I MGMTCLAS = (NONE,NONE)
BMC50470I STORCLAS = (NONE,NONE)
BMC50470I THRESHLD = 0
BMC50470I MAXEXTSZ = ((0,K),(0,K))
BMC50470I EXPDT =
BMC50470I RETPD =
BMC50470I GDGLIMIT = 5
BMC50470I GDGEMPTY = NO
BMC50470I GDSCRAM = NO
BMC50483I LOAD       DSNPAT=&UID..BMC.&TS.&DDNAME
BMC50483I WORK       DSNPAT=&UID..BMC.&TS.&DDNAME
BMC50483I SORTWORK   DSNPAT=NONE
BMC50483I ERROR      DSNPAT=&UID..BMC.&TS.&DDNAME
BMC50483I DISCARD    DSNPAT=&UID..BMC.&TS.&DDNAME
BMC50483I SYSMAP     DSNPAT=&UID..BMC.&TS.&DDNAME
BMC50483I LOCPLCPY   DSNPAT=&UID.&DDNAME.&TS..T&TIME
BMC50483I LOCBLCPY   DSNPAT=&UID.&DDNAME.&TS..T&TIME
BMC50483I REMPLCPY   DSNPAT=&UID.&DDNAME.&TS..T&TIME
BMC50483I REMBLCPY   DSNPAT=&UID.&DDNAME.&TS..T&TIME
BMC50483I LOCPXCPY   DSNPAT=&UID.&DDNAME.&TS..F&PART..T&TIME
BMC50471I DB2 DSNHDECP MODULE SETTINGS:
BMC50471I VERSION                 = 1110
BMC50471I SUBSYSTEM DEFAULT       = DEHJ
BMC50471I CHARACTER SET           = ALPHANUM
BMC50471I DATE FORMAT             = USA
BMC50471I TIME FORMAT             = USA
BMC50471I LOCAL DATE LENGTH       = 0
BMC50471I LOCAL TIME LENGTH       = 0
BMC50471I DECIMAL POINT           = PERIOD
BMC50471I DECIMAL ARITHMETIC      = 15
BMC50471I DELIMITER               = DEFAULT
BMC50471I SQL DELIMITER           = 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 UNICODE CCSID           = (367,1208,1200)
BMC50471I IMPLICIT TIME ZONE      = CURRENT (-05:00)
BMC50028I DB2 MODE = NFM
BMC50471I BMC_BMCUTIL       ='BM UTIL.CMN_BMCUTIL'
BMC50471I BMC_BMCSYNC       ='BM UTIL.CMN_BMCSYNC'
BMC50471I BMC_BMCHIST       ='BM UTIL.CMN_BMCHIST'
BMC50471I BMC_BMCDICT       ='BM UTIL.CMN_BMCDICT'

Figure 66 LOADPLUS SYSPRINT for example 19 (part 4 of 6)
Example 19: Unloading data to LOADPLUS

Figure 66  LOADPLUS SYSPRINT for example 19 (part 5 of 6)
Figure 67 shows the output of a SELECT on the loaded table for example 19.

**Figure 67**

**SELECT output from example 19 (part 1 of 2)**

```sql
SELECT * FROM BMC.TBEXB19;
```
### Example 19: Unloading data to LOADPLUS

**Figure 67  SELECT output from example 19 (part 2 of 2)**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12_</td>
<td>Britain</td>
<td>Michelle</td>
<td>H</td>
<td>?</td>
<td>4200</td>
<td>12/24/1994</td>
<td></td>
</tr>
<tr>
<td>13_</td>
<td>Franklin</td>
<td>Erica</td>
<td>S</td>
<td>32000.00</td>
<td>4300</td>
<td>04/18/1990</td>
<td></td>
</tr>
<tr>
<td>14_</td>
<td>Williamson</td>
<td>Floyd</td>
<td>H</td>
<td>5.75</td>
<td>5100</td>
<td>04/03/1992</td>
<td></td>
</tr>
<tr>
<td>15_</td>
<td>George</td>
<td>Hugh</td>
<td>H</td>
<td>7.50</td>
<td>5100</td>
<td>05/19/1989</td>
<td></td>
</tr>
<tr>
<td>16_</td>
<td>Slaughter</td>
<td>Johnath</td>
<td>H</td>
<td>6.25</td>
<td>5200</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>17_</td>
<td>Johnson</td>
<td>Robert</td>
<td>H</td>
<td>?</td>
<td>6300</td>
<td>07/31/1991</td>
<td></td>
</tr>
<tr>
<td>18_</td>
<td>Peterson</td>
<td>Barbara</td>
<td>H</td>
<td>6.90</td>
<td>6700</td>
<td>12/19/1989</td>
<td></td>
</tr>
<tr>
<td>19_</td>
<td>Kennedy</td>
<td>Jennifer</td>
<td>S</td>
<td>36000.00</td>
<td>8200</td>
<td>03/31/1993</td>
<td></td>
</tr>
<tr>
<td>20_</td>
<td>Wilson</td>
<td>Lindsay</td>
<td>S</td>
<td>45000.00</td>
<td>8200</td>
<td>04/25/1992</td>
<td></td>
</tr>
</tbody>
</table>

SUCCESSFUL RETRIEVAL OF 20 ROW(S)
Tuning UNLOAD PLUS jobs

This chapter presents the following topics:

Tuning for performance .......................................................... 419
  Setting installation options for optimal performance ................. 419
  Specifying command options for optimal performance .......... 421
  Additional performance information for installation and command options . 423
  Enabling multitasking for performance .................................. 427
  Additional performance tuning recommendations .................... 429
  Performance tuning for specific scenarios .............................. 430
  Interpreting performance-related messages ............................ 431
Tuning to improve memory use .............................................. 434
  Tuning sort processing ....................................................... 434
  Tuning when defining a large number of partitions ................. 434

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 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. Table 86 on page 420 describes these recommendations. For a complete description of each option, see Appendix A, “UNLOAD PLUS installation options.”
If you modify these options after installation, you must rerun the installation job for these modifications to take effect.

Table 86   Installation option changes for performance (part 1 of 2)

<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>“Controlling memory usage (SMCORE)” on page 425</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>
Specifying command options for optimal performance

Table 86  Installation option changes for performance (part 2 of 2)

<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 states, specify this value to obtain significant savings in CPU overhead. | “UXSTATE installation option” on page 427 |

Specifying command options for optimal performance

Table 87 and Table 88 on page 422 describe command option recommendations to obtain optimal performance when running UNLOAD PLUS.

NOTE

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.

Table 87 lists UNLOAD command options that correspond to the installation options listed in Table 86 on page 420. 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 Table 86 for comments about these command options.

Table 87  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)</td>
<td>DRNRETRY=(n)</td>
</tr>
<tr>
<td>where (n) is based on your environment</td>
<td>where (n) is based on your environment</td>
</tr>
<tr>
<td>RETRY_DELAY (n)</td>
<td>DRNDELAY=(n)</td>
</tr>
<tr>
<td>where (n) is based on your environment</td>
<td>where (n) is based on your environment</td>
</tr>
<tr>
<td>ACTIVE(YES,YES)</td>
<td>UNLOADDN_ACTIVE=(YES,YES)</td>
</tr>
</tbody>
</table>
Table 88 describes additional recommended command options. (No installation options correspond to these command options.)

### Table 88 Command option values for performance (part 1 of 2)

<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 <em>not</em> specify this option unless your job requires it. CCSID translation can degrade performance.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>CNTLCARDS BMLOAD '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 BMLOAD, 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 429</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 <em>not</em> 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>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 “Output format” on page 71.</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></td>
</tr>
<tr>
<td></td>
<td>This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- FORMAT EXTERNAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This option limits the ability of UNLOAD PLUS to maximize the number of records that will fit in a block.</td>
<td></td>
</tr>
<tr>
<td>MAXBLKSIZE</td>
<td>Do <em>not</em> 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 429</td>
</tr>
</tbody>
</table>
Additional performance information for installation and command options

The following sections 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.

**Buffer options (IBUFFS and UBUFFS installation options)**

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 430.
**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.

**NOTE**

If you specify the INFILE option, UNLOAD PLUS can use any of the following copy data sets as input instead of the DB2 table space:

- full or incremental image copies
- DSN1COPY data sets
- inline copies
- encrypted copies
- cabinet copies
- Instant Snapshot copies
- online consistent copies
- VSAM FlashCopy 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 430.

**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 SYSRECnn and SYSREDnn data sets contain the output data that UNLOAD PLUS unloaded from one or more DB2 tables.)

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.
**Sort options (SMCORE and SMAX installation options and MAXSORTS command option)**

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)**

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 as discussed in “Controlling the number of sort processes (SMAX)” on page 426

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 (SMAX)**

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 SMAX installation option.

Under normal circumstances, you should allow UNLOAD PLUS to control the number of sorts that it processes concurrently. The value of the SMAX 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 SMAX 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. See “MAXSORTS” on page 154.
UXSTATE installation option

UNLOAD PLUS invokes DB2 user exits during processing as described in “DB2 user exits” on page 84. 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.

Enabling multitasking for performance

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 Table 86 on page 420, 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 199.

### 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.

**NOTE**

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. For more information about this command option, see “FILTERPART” on page 116.

Additional performance tuning recommendations

The following sections 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

Table 89 on page 431 describes the steps that you can take to tune specific types of unload jobs to improve performance.
Interpreting performance-related messages

UNLOAD PLUS issues performance-related messages if you specify MSGLEVEL(1) in the execution JCL. See “Utility parameters on the EXEC statement” on page 273 for more details about specifying this utility parameter. Information in these messages can help you monitor performance of UNLOAD PLUS and fine-tune future runs.

### Table 89 Performance tuning scenarios

<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: <code>CNTLCARDS BMCLOAD 'ENUMROWS' 'PRELOAD LOAD' 'SORTNUM 32'</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unloading LOB or XML data (DIRECT YES only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></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>
<tr>
<td>multitasking</td>
<td>running a multitasking job that unloads into a single SYSREC data set</td>
<td>This job can cause buffer contention.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If message BMC50478I reports RDBLOCK WAITS greater than zero, set UBUFFS to 255 to improve performance. Create a separate options module to use for this type of unload job.</td>
</tr>
<tr>
<td>high availability</td>
<td>appropriate SHRLEVEL option</td>
<td>Specify SHRLEVEL CHANGE or SHRLEVEL REFERENCE depending on your goals.</td>
</tr>
</tbody>
</table>
Interpreting performance-related messages

**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 430 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 “Buffer options (IBUFFS and UBUFFS installation options)” on page 423 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.
Tuning to improve memory use

BMC50478I  \textit{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

To help you determine how to improve memory usage, consider the information in the following sections.

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 defining 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.
Tuning when defining a large number of partitions
UNLOAD PLUS installation options

This appendix presents the following topics:

Overview ................................................................. 437
Basic UNLOAD PLUS installation options ......................... 438
Dynamic allocation installation options ............................... 464
  Default output descriptor options for dynamic allocation ....... 465
  Options that are common to disk and tape data sets .......... 466
Options for disk data sets ............................................. 474
Options for tape data sets ............................................. 478
DYNALOC installation option ......................................... 480

Overview

When you use the BMC Installation System to install the UNLOAD PLUS for DB2 product, the customization 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:

- $C30DOPT 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.)

- $C32SOPT 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 customization 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 $C30DOPT or $C32SOPT 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 Chapter 4, “Building and executing UNLOAD PLUS jobs.” For more information about customizing your installation of UNLOAD PLUS, see the *BMC Products and Solutions for DB2 Configuration Guide*.

To find a description of each options macro, use Table 90.

### Table 90  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>$C30DOPT</td>
<td>$ADUOPTS</td>
<td>basic options</td>
<td>“Basic UNLOAD PLUS installation options”</td>
</tr>
<tr>
<td></td>
<td>$ADUOUTP</td>
<td>options for dynamic allocation</td>
<td>“Dynamic allocation installation options” on page 464</td>
</tr>
<tr>
<td>$C32SOPT</td>
<td>$AUPSMAC</td>
<td>options for BMCSORT</td>
<td>“DYNALOC installation option” on page 480</td>
</tr>
</tbody>
</table>

### Basic UNLOAD PLUS installation options

Table 91 shows the options contained in the $ADUOPTS macro in $C30DOPT. 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 91  Basic UNLOAD PLUS installation options (part 1 of 4)

<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>page 441</td>
</tr>
<tr>
<td>CENTURY</td>
<td>No value</td>
<td>century specification for two-digit years</td>
<td>page 442</td>
</tr>
</tbody>
</table>
Table 91  Basic UNLOAD PLUS installation options (part 2 of 4)

<table>
<thead>
<tr>
<th>Option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE_CONSISTENT</td>
<td>NO</td>
<td>SHRLEVEL CHANGE CONSISTENT default</td>
<td>page 442</td>
</tr>
<tr>
<td>CHANGE QUIESCE</td>
<td>NO</td>
<td>QUIESCE default</td>
<td>page 443</td>
</tr>
<tr>
<td>CMAX</td>
<td>16</td>
<td>maximum number of concurrent tasks when using dynamic SQL</td>
<td>page 443</td>
</tr>
<tr>
<td>CMRATIO</td>
<td>50</td>
<td>compression ratio</td>
<td>page 444</td>
</tr>
<tr>
<td>CONSTRULES</td>
<td>BMC</td>
<td>how to handle character constants in SELECT lists</td>
<td>page 444</td>
</tr>
<tr>
<td>CURRENTDEGREE</td>
<td>NONE</td>
<td>whether to issue SET CURRENT DEGREE before dynamically executed SELECT statements</td>
<td>page 444</td>
</tr>
<tr>
<td>DELFILES</td>
<td>YES</td>
<td>whether to delete work files on failure</td>
<td>page 445</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>page 446</td>
</tr>
<tr>
<td>DRNRETRY</td>
<td>255</td>
<td>maximum number of times to attempt to obtain a drain before terminating</td>
<td>page 446</td>
</tr>
<tr>
<td>DRNWAIT</td>
<td>NONE</td>
<td>timeout value for object drains</td>
<td>page 446</td>
</tr>
<tr>
<td>DSPLOCKS</td>
<td>DRNFAIL</td>
<td>how to display locks if the drain times out</td>
<td>page 447</td>
</tr>
<tr>
<td>EXCLUDUMP</td>
<td>(X37,X22,X06)</td>
<td>system codes to ignore during dump processing</td>
<td>page 448</td>
</tr>
<tr>
<td>FILEREFDN</td>
<td>SYSREF</td>
<td>default output descriptor for referenced output files</td>
<td>page 448</td>
</tr>
<tr>
<td>FILL</td>
<td>NO</td>
<td>whether to fill numeric external data types with zeros</td>
<td>page 448</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>page 449</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>page 449</td>
</tr>
<tr>
<td>FORCE RPT</td>
<td>NO</td>
<td>whether to display a report of the canceled threads</td>
<td>page 450</td>
</tr>
<tr>
<td>HISTORY</td>
<td>YES</td>
<td>whether to update the BMCHIST table for each table space</td>
<td>page 451</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>page 451</td>
</tr>
<tr>
<td>INLINE</td>
<td>NO</td>
<td>whether to unload from inline image copies</td>
<td>page 451</td>
</tr>
<tr>
<td>KEYDSNAM</td>
<td>No value</td>
<td>example value: ACP.EKEYDS</td>
<td>page 452</td>
</tr>
<tr>
<td>LOADDECP</td>
<td>NO</td>
<td>whether to use a DSNHDECP other than the local subsystem default</td>
<td>page 452</td>
</tr>
</tbody>
</table>
### Table 91  Basic UNLOAD PLUS installation options (part 3 of 4)

<table>
<thead>
<tr>
<th>Option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCKROW</td>
<td>YES</td>
<td>serialization method for BMCSYNC and BMCUTIL</td>
<td>page 452</td>
</tr>
<tr>
<td>MAXP</td>
<td>5</td>
<td>maximum number of partitions to unload concurrently</td>
<td>page 453</td>
</tr>
<tr>
<td>MSGLEVEL</td>
<td>1</td>
<td>default for the message-level execution parameter</td>
<td>page 453</td>
</tr>
<tr>
<td>NULLCHAR</td>
<td>?</td>
<td>default character for the null indicator</td>
<td>page 453</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>page 454</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>page 454</td>
</tr>
<tr>
<td>PLAN</td>
<td>ADU1110</td>
<td>product plan name</td>
<td>page 455</td>
</tr>
<tr>
<td>RECFM</td>
<td>AUTO</td>
<td>record format for unload data sets</td>
<td>page 455</td>
</tr>
<tr>
<td>ROWSETSZ</td>
<td>100</td>
<td>size of rowset for a single FETCH request</td>
<td>page 455</td>
</tr>
<tr>
<td>SDUMP</td>
<td>(ALLPSA,CSA, RGN, SQA, LSQA, SUM, TRT, IO)</td>
<td>system dump parameters</td>
<td>page 456</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>REFERENCE</td>
<td>SHRLEVEL option default</td>
<td>page 456</td>
</tr>
<tr>
<td>SMAX</td>
<td>16</td>
<td>maximum number of sort processes to invoke concurrently</td>
<td>page 456</td>
</tr>
<tr>
<td>SMCORE</td>
<td>(0K,0K)</td>
<td>maximum amount of sort memory</td>
<td>page 457</td>
</tr>
<tr>
<td>SORTNUM</td>
<td>32</td>
<td>number of sort work files per task for BMCSORT to allocate dynamically</td>
<td>page 458</td>
</tr>
<tr>
<td>SQLDELAY</td>
<td>3</td>
<td>number of seconds between retry attempts after SQLCODE -911</td>
<td>page 458</td>
</tr>
<tr>
<td>SQLRETRY</td>
<td>100</td>
<td>number of retry attempts after SQLCODE -911</td>
<td>page 458</td>
</tr>
<tr>
<td>TAPEDISP</td>
<td>DELETE</td>
<td>tape file disposition</td>
<td>page 459</td>
</tr>
<tr>
<td>TAPES</td>
<td>NONE</td>
<td>list of default tape units</td>
<td>page 459</td>
</tr>
<tr>
<td>TASKMAX</td>
<td>200%</td>
<td>global setting for the maximum number of concurrent tasks to start</td>
<td>page 459</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>page 459</td>
</tr>
<tr>
<td>UNLDMAX</td>
<td>*</td>
<td>maximum number of tasks to start per CPU during the UNLOAD phase</td>
<td>page 459</td>
</tr>
<tr>
<td>UNLOADDN_ACTIVE</td>
<td>(YES,NO)</td>
<td>whether to activate dynamic allocation for unload data sets</td>
<td>page 460</td>
</tr>
<tr>
<td>UNLOADDN</td>
<td>(SYSREC,SYSRED)</td>
<td>default ddnames or output descriptors for unload data sets</td>
<td>page 460</td>
</tr>
<tr>
<td>USELRECL</td>
<td>NO</td>
<td>whether to use the LCRECL value in the JCL</td>
<td>page 461</td>
</tr>
<tr>
<td>UXSTATE</td>
<td>SUP</td>
<td>how to invoke DB2 user exits</td>
<td>page 462</td>
</tr>
<tr>
<td>WORKUNIT</td>
<td>SYSALLDA</td>
<td>temporary unit for work data sets</td>
<td>page 462</td>
</tr>
</tbody>
</table>
This section describes each of the options. For more information about setting the values of some of these options at runtime, see Chapter 3, “Syntax of the UNLOAD command.” For more information about performance implications of these options, see Chapter 6, “Tuning UNLOAD PLUS jobs.”

**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 “HURBA” on page 132.

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)

For more information about the priority and hierarchy that UNLOAD PLUS uses, see “Using dynamically allocated unload data sets” on page 293.

**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 (see page 131).

**CENTURY**

The CENTURY option specifies the 100-year range that determines the century for the DATE and TIMESTAMP 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.

NOTE
UNLOAD PLUS does not supply a value for this option. If you do not supply a value, your installation options module does not assemble successfully.

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 page 183) or the CENTURY field specification option (see page 249).

**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 “SHRLEVEL CHANGE CONSISTENT YES considerations” on page 61.
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 110 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**

**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 110 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. For more information, see “DIRECT” on page 106.
**Overriding this option**
You can override the value for this option by using the MAXCONNECT command option (see page 155).

**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 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 command option. See “CMRATIO” on page 133 for more information.

**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 (see page 109).

**DELFILES=YES**

The DELFILES option tells UNLOAD PLUS whether to delete the SYSREC, SYSRED, SYSCNTL, SORTWK, and referenced files when the unload is unsuccessful. (For more information, see “ON FAILURE” on page 178.)

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, SYSCNTL, 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 DELETEFILES 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 DELETEFILES command option which is described on page 181.

**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, which is described on page 185.

**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, which is described on page 185.

**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. For more information, see page 449.

---

**Overriding this option**

You can override the value for this option by using the DRAIN_WAIT command option, which is described on page 184.

**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, which is described on page 186.

EXCLDUMP=(X37,X22,X06)

The EXCLDUMP option allows you to limit the conditions under which UNLOAD PLUS generates a system dump when the SDUMP option (see page 456) 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 UNLOADDDN 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 (see page 164) or by using the FILL option on the individual field specifications (see the data type descriptions starting on page 240).
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.

**NOTE**
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” on page 59.

**Overriding this option**
You can override the value for this option by using the FORCE command option (page 187). 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.

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 (page 185) or DRNRETRY installation option (page 446).
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 (page 187). You can override the value for the second parameter by using the FORCE_DELAY suboption of the FORCE command option (page 188).

**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 (page 188).
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 more information, see “IBUFFS installation option” on page 424.

- 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 (page 118).
**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 (page 118). 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.

*Overriding 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 "Message level (MSGLEVEL)" on page 275.

**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.
Basic UNLOAD PLUS installation options

**NOTE**
When you specify or default to ?, the job SYSPRINT displays a value of X'6F'.

Overriding this option
You can override the value for this option by using the NULLCHAR command option (see page 162).

**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)

**NOTE**
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 (see page 163).

**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 “Data set authorization” on page 52.
PLAN=ADU1110

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 (see page 136).

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 (see page 108).
**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. For details, see the EXCLDUMP option on page 448.

**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. See “SHRLEVEL” on page 110 for more information.

**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 “Controlling the number of sort processes (SMAX)” on page 426.

**Overriding this option**
You can override the value for this option by using the MAXSORTS command option. See “MAXSORTS” on page 154 for more information.

**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 “Controlling memory usage (SMCORE)” on page 425.

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.

Table 92 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>
<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 480.</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” on page 281.

**Overriding this option**
You can override the value for this option by using the SORTNUM command option (page 149).

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 “UBUFFS installation option” on page 424.

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.

NOTE
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

NOTE
During installation, the names that you specify for this option become the default output descriptor names (see page 466) 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 (see page 129).
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.
Basic UNLOAD PLUS installation options

**Overriding this option**
You can override the value for this option by specifying the USELRECL command option (see page 134).

**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 “SHRLEVEL CHANGE CONSISTENT YES considerations” on page 61.
Overriding this option
You can override the value for this option by using the XBMID command option (page 114).

ZIIP=ENABLED

The ZIIP option tells UNLOAD PLUS whether to attempt to use IBM System 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. For more information, see “XBMID” on
page 114 or page 462.

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.
Dynamic allocation installation options

Overriding this option
You can override the value for this option by using the ZIIP command option (page 113).

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.

**NOTE**
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. See “ZONEDDECOVP” on page 171 for more information.

Dynamic allocation installation options

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 section 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 474) and others only with tape data sets (“Options for tape data sets” on page 478).

Table 93 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 93  UNLOAD PLUS installation options for dynamic allocation (part 1 of 2)

<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>page 473</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>page 478</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>page 477</td>
</tr>
<tr>
<td></td>
<td>example value: 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNAME</td>
<td>Value depends on the type of data set</td>
<td>default name for the output data set</td>
<td>page 467</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>(for SYSREF) PDS</td>
<td>type of referenced file or unload data set being used or allocated</td>
<td>page 470</td>
</tr>
<tr>
<td></td>
<td>(for SYSREC and SYSRED) NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPDT</td>
<td>99000</td>
<td>expiration date for tape data sets</td>
<td>page 479</td>
</tr>
<tr>
<td>FILESZPCT</td>
<td>100</td>
<td>percentage of calculated data set size to allocate when using automatic sizing</td>
<td>page 476</td>
</tr>
<tr>
<td>GDGEMPTY</td>
<td>NO</td>
<td>whether to uncatalog all data sets when the limit is reached</td>
<td>page 472</td>
</tr>
<tr>
<td>GDGLIMIT</td>
<td>5</td>
<td>maximum number of generation data sets in a group</td>
<td>page 472</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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDGSCRATCH</td>
<td>NO</td>
<td>whether to keep DSCB information when the data sets are uncataloged</td>
<td>page 472</td>
</tr>
<tr>
<td>MAXPRIM</td>
<td>0</td>
<td>maximum primary allocation</td>
<td>page 475</td>
</tr>
<tr>
<td>MAXSECD</td>
<td>0</td>
<td>maximum secondary allocation</td>
<td>page 475</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>No value</td>
<td>SMS management class name</td>
<td>page 473</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>page 476</td>
</tr>
<tr>
<td>PCTPRIM</td>
<td>AUTO</td>
<td>primary allocation percentage of disk space</td>
<td>page 474</td>
</tr>
<tr>
<td>RETPD</td>
<td>No value</td>
<td>retention period for tape data sets</td>
<td>page 479</td>
</tr>
<tr>
<td></td>
<td>example value: 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>CYL</td>
<td>space unit of allocation</td>
<td>page 474</td>
</tr>
<tr>
<td>STORCLAS</td>
<td>No value</td>
<td>SMS storage class name</td>
<td>page 473</td>
</tr>
<tr>
<td></td>
<td>example value: TEMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRTCH</td>
<td>NONE</td>
<td>tape compression</td>
<td>page 479</td>
</tr>
<tr>
<td>UNIT</td>
<td>SYSALLDA</td>
<td>default device name</td>
<td>page 467</td>
</tr>
<tr>
<td>UNICTNT</td>
<td>0</td>
<td>number of devices to dynamically allocate</td>
<td>page 474</td>
</tr>
<tr>
<td>VOLCNT</td>
<td>25</td>
<td>largest number of volumes to process</td>
<td>page 471</td>
</tr>
<tr>
<td>VOLUMES</td>
<td>No value</td>
<td>default list of volumes</td>
<td>page 477</td>
</tr>
<tr>
<td></td>
<td>example value: (DEV1,DEV2,DEV3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 93  UNLOAD PLUS installation options for dynamic allocation (part 2 of 2)
Options that are common to disk and tape data sets

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 (see page 192).

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 &TS.XX are valid, but &TSXX is invalid. Table 95 on page 468 contains the symbolic variables, a definition of each, and limits on length.
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:

- 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)¹,²</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 c,d</td>
<td>current date (in the format YYMMDD)</td>
<td>6</td>
</tr>
<tr>
<td>&amp;DAY c,d</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 c,d</td>
<td>current hour (in the format HH)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;JDATE c,d</td>
<td>current Julian date (in the format YYDDD)</td>
<td>5</td>
</tr>
<tr>
<td>&amp;JDAY c,d</td>
<td>current Julian day (in the format DDDD)</td>
<td>3</td>
</tr>
<tr>
<td>&amp;JOBNAME</td>
<td>job name</td>
<td>8 maximum</td>
</tr>
<tr>
<td>&amp;MIN c,d</td>
<td>current minute (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;MINUTE c,d</td>
<td>current minute (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;MONTH c,d</td>
<td>current month (in the format MM)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;OBIDc</td>
<td>OBID of the table that UNLOAD PLUS is unloading</td>
<td>4 bytes maximum</td>
</tr>
</tbody>
</table>
Table 95  Symbolic variables for data set names within SYSIN (part 2 of 3)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;PARTc,d</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;POSNUMc</td>
<td>position number of the XML or LOB column within the select list</td>
<td>3</td>
</tr>
<tr>
<td>&amp;SEC c,d</td>
<td>current second (in the format SS)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;SECONDc,d</td>
<td>current second (in the format SS)</td>
<td>2</td>
</tr>
<tr>
<td>&amp;SELNUMc</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 c,d</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>
<tr>
<td>&amp;TYPE</td>
<td>data set type that UNLOAD PLUS is allocating</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Note: This symbol resolves to a value of SYSREC, SYSRED, or SYSREF (where SYSREF indicates that this is a referenced file).</td>
<td></td>
</tr>
<tr>
<td>&amp;USERID</td>
<td>job or TSO user</td>
<td>7 maximum</td>
</tr>
</tbody>
</table>

Note: UNLOAD PLUS ignores PROC names.
Options that are common to disk and tape data sets

Overriding this option
You can override the value for this option by using the DSNAME command option (see page 193).

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 62.

- For unload data sets (SYSREC and SYSRED), the remaining values for DSNTYPE tell UNLOAD PLUS what type of data set to allocate dynamically.

Table 95  Symbolic variables for data set names within SYSIN (part 3 of 3)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Length (bytes)a,b</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;UTIL</td>
<td>utility ID</td>
<td>8 maximum</td>
</tr>
<tr>
<td></td>
<td>Note the following information about the utility ID variable:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- UNLOAD PLUS truncates values longer than 8 bytes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Utility IDs that include special characters might cause UNLOAD PLUS to generate invalid data set names.</td>
<td></td>
</tr>
<tr>
<td>&amp;YEARc,d</td>
<td>current year (in the format YY)</td>
<td>2</td>
</tr>
</tbody>
</table>

a  UNLOAD PLUS removes trailing blanks in the result.

b  The maximum total length that UNLOAD PLUS allows for a data set name is 44 bytes.

c  You must prefix symbols with a numeric result with one or more alphabetic characters.

d  UNLOAD PLUS assigns values for these variables when the utility allocates the output data set. All dynamically allocated data sets have the same value.
Table 96 describes the valid values for this option.

**Table 96 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: The only other valid dynamic allocation installation options with DSNTYPE HFS are the output descriptor name, DSNAME, and SUBSETS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</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 (page 467) supports extended attributes.</td>
</tr>
</tbody>
</table>

**Overriding this option**

You can override the value for this option by using the DSNTYPE command option (see page 197).

**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.
Options that are common to disk and tape data sets

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 (see page 201).

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 (see page 202).

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.

---

**NOTE**

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 (see page 203).

**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.

---

**NOTE**

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 (see page 203).

**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.
**Options for disk data sets**

**NOTE**
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.

**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 (see page 205).

**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 (see page 207).

**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.
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 (see page 207).

**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 (see page 208).

**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.
Options for disk data sets

**Overriding this option**
You can override the value for this option by using the MAXSECD command option in an OUTPUT descriptor statement (see page 209).

**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:

\[
\text{resultantFileSize} = \text{ROUND} \left( \frac{\text{calculatedFileSize} \times \text{FILESZPCT}}{100} \right)
\]

**NOTE**
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 (page 211).

**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.
NOTE

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 (see page 210).

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 (see page 212).

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.
Options for tape data sets

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.

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.

Options for tape data sets

You can use the following options for tape data sets only.

NOTE
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 (see page 215).

**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.

**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.
DYNALOC installation option

The DYNALOC installation option 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 $C32SOPT 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

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 97 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 either 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>
<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 UNLOAD PLUS should dynamically allocate SORTWK data sets. This parameter applies only when the Data Facility Storage Management Subsystem (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>position 2</td>
<td>Do not change this value. UNLOAD PLUS 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>
## Table 97 DYNALOC parameters (part 2 of 3)

<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 3</td>
<td>This parameter tells UNLOAD PLUS whether to dynamically allocate SORTWK files.</td>
<td>ON</td>
<td>• ON dynamically allocates SORTWK.</td>
</tr>
<tr>
<td></td>
<td>Note: BMC recommends that you not change this value.</td>
<td></td>
<td>• OFF does not dynamically allocate SORTWK.</td>
</tr>
<tr>
<td>position 4</td>
<td>Do not change this value. UNLOAD PLUS does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>ON</td>
<td>Do not change this value.</td>
</tr>
<tr>
<td>position 5</td>
<td>Do not change this value. UNLOAD PLUS does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>6000000</td>
<td>Do not change this value.</td>
</tr>
<tr>
<td>position 6</td>
<td>Do not change this value. UNLOAD PLUS does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>3000000</td>
<td>Do not change this value.</td>
</tr>
<tr>
<td>position 7</td>
<td>This parameter specifies the DASD type with the smallest track capacity that a dynamically allocated SORTWK data set might encounter at your site.</td>
<td>3390</td>
<td>• 3380, track capacity of 47968</td>
</tr>
<tr>
<td></td>
<td>Note: If your installation has an automatic class selection (ACS) routine, it can override this specification.</td>
<td></td>
<td>• 3390, track capacity of 56664</td>
</tr>
<tr>
<td></td>
<td>Note: 9345, track capacity of 46456</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td>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, UNLOAD PLUS uses the value from the first DYNALOC parameter.</td>
<td>blank</td>
<td>Use any valid DFSMS storage class.</td>
</tr>
</tbody>
</table>
**Table 97  DYNALOC parameters (part 3 of 3)**

<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 UNLOAD PLUS to handle retry attempts for SORTWK dynamic allocation:  
  - The first subparameter indicates the number of times that you want UNLOAD PLUS 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 UNLOAD PLUS to retry the request.  
  - 0 through 15 for the second subparameter. 0 indicates that you do not want UNLOAD PLUS to retry the request. |
Appendix B: Common utility tables

This appendix presents the following topics:

Overview ............................................................... 483
  Considerations and warnings for common utility tables ................. 484
  Managing common utility tables .................................... 485
BMCDICT table ........................................................ 487
  Considerations for the BMCDICT table ................................ 487
  Maintaining the BMCDICT table .................................... 488
BMCHIST table ......................................................... 488
  Maintaining the BMCHIST table .................................... 490
BMCLGRNX table ....................................................... 490
BMCSYNC table ........................................................ 491
  Considerations for the BMCSYNC table ............................... 493
  Maintaining the BMCSYNC table .................................... 494
BMCTRANS table ....................................................... 495
BMCUTIL table ......................................................... 496
  Maintaining the BMCUTIL table .................................... 498
BMCXCOPY table ....................................................... 499
  Maintaining the BMCXCOPY table .................................... 503

Overview

The BMC common utility tables contain information about the BMC utilities that you generate and submit through a BMC utility product. Table 98 on page 484 lists the tables that each utility uses and each table’s default name and synonym.
Considerations and warnings for common utility tables

Note the following considerations when using the common utility tables:

- 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:
  - BMCUUTIL
  - BMCHIST
  - BMCSYNC
  - BMCXCOPY

### Table 98 Common utility tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Default name</th>
<th>Synonym</th>
<th>CHECK PLUS</th>
<th>COPY PLUS</th>
<th>RECOVER PLUS</th>
<th>REORG PLUS</th>
<th>UNLOAD PLUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCDICT</td>
<td>CMN_BMCDICT</td>
<td>BMC_BMCDICT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMCHIST</td>
<td>CMN_BMCHIST</td>
<td>BMC_BMCHIST</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BMCLGRNX</td>
<td>CMN_BMCLGRNX</td>
<td>BMC_BMCLGRNX</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BMCSYNC</td>
<td>CMN_BMCSYNC</td>
<td>BMC_BMCSYNC</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BMCTRANS</td>
<td>CMN_BMCTRANS</td>
<td>BMC_BMCTRANS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BMCUTIL</td>
<td>CMN_BMCUTIL</td>
<td>BMC_BMCUTIL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BMCXCOPY</td>
<td>CMN_BMCXCOPY</td>
<td>BMC_BMCXCOPY</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Managing common utility tables

This section 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. 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 Table 98 on page 484):

  ```sql
  SELECT CREATOR, NAME FROM SYSCAT.SYSTABLES
  WHERE TSNAME = 'tableName';
  ```

- Get the names from your DB2 system administrator.

**WARNING**

The following warnings apply:

- 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 the BMC tables, with the exception of 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.
To query the tables

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

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

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 TERM as the restart parameter.

**BMCDICT table**

Table 99 describes the BMCDICT table, which stores the compression dictionary during load or reorganization processing.

<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>

**Considerations for the BMCDICT table**

Note the following considerations:

- 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 and 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.

If you need to control the expansion of this table, use the following procedure:

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

Table 100 on page 489 describes the BMCHIST table, which 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.
### Table 100  BMCHIST 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 that contains the table or index space</td>
</tr>
<tr>
<td>SPNAME</td>
<td>CHAR(8)</td>
<td>name of the table or index space</td>
</tr>
<tr>
<td>UTILNAME</td>
<td>CHAR(8)</td>
<td>name of the 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>• 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>UTILID</td>
<td>CHAR(16)</td>
<td>utility identifier</td>
</tr>
<tr>
<td>AUTHID</td>
<td>CHAR(8)</td>
<td>user ID that ran the utility</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>date that the utility completed</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>time that the utility completed</td>
</tr>
<tr>
<td>ELAPSED</td>
<td>TIME</td>
<td>elapsed time of the utility</td>
</tr>
<tr>
<td>PARTITION</td>
<td>LONG VARCHAR</td>
<td>ALL, or the partition numbers as specified by the DSNUM option (for COPY PLUS) or the PART option</td>
</tr>
<tr>
<td>OBJNAME</td>
<td>VARCHAR(27)</td>
<td>fully qualified object name</td>
</tr>
<tr>
<td>PHASE_1</td>
<td>CHAR(8)</td>
<td>name of utility phase 1</td>
</tr>
<tr>
<td>ELAPSED_1</td>
<td>TIME</td>
<td>elapsed time of phase 1</td>
</tr>
<tr>
<td>PHASE_2</td>
<td>CHAR(8)</td>
<td>name of utility phase 2</td>
</tr>
<tr>
<td>ELAPSED_2</td>
<td>TIME</td>
<td>elapsed time of phase 2</td>
</tr>
<tr>
<td>PHASE_3</td>
<td>CHAR(8)</td>
<td>name of utility phase 3</td>
</tr>
<tr>
<td>ELAPSED_3</td>
<td>TIME</td>
<td>elapsed time of phase 3</td>
</tr>
<tr>
<td>PHASE_4</td>
<td>CHAR(8)</td>
<td>name of utility phase 4</td>
</tr>
<tr>
<td>ELAPSED_4</td>
<td>TIME</td>
<td>elapsed time of phase 4</td>
</tr>
<tr>
<td>PHASE_5</td>
<td>CHAR(8)</td>
<td>name of utility phase 5</td>
</tr>
<tr>
<td>ELAPSED_5</td>
<td>TIME</td>
<td>elapsed time of phase 5</td>
</tr>
</tbody>
</table>

Note the following conditions:

- This column lists only three-digit partitions (any loaded partitions 1 through 999). Four-digit partitions (any loaded partitions from 1000 through 4096) are not stored in this column. For jobs that load only four-digit partitions, this column is empty.

- If the list of partitions exceeds 1011 bytes, the utility truncates the value that is stored in this column.

- For UNLOAD PLUS, if you specified LOGICAL PART, these partitions are the physical partitions that correspond to the logical partitions that you specified.
Maintaining the BMCHIST table

When a utility completes successfully, it inserts a row into the BMCHIST table. Periodically, review BMCHIST and delete old rows to control its expansion.

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';
```

BMCLGRRX table

Table 101 describes the contents of the BMCLGRRX table, which contains log ranges that show when a table space was open for updates.

Table 101  BMCLGRRX 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>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></td>
<td></td>
<td>For non-data-sharing, the value is X’000000000000’.</td>
</tr>
<tr>
<td>LGRELRSN</td>
<td>CHAR(6)</td>
<td>ending LRSN of update log records for data sharing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For non-data-sharing, the value is X’000000000000’.</td>
</tr>
<tr>
<td>LGRMEMBER</td>
<td>CHAR(2)</td>
<td>data sharing member ID of the modifying DB2 subsystem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For non-data-sharing, the value is X’0001’.</td>
</tr>
</tbody>
</table>
BMCSYNC table

Table 102 describes the BMCSYNC table, which contains information about the status of the objects that the currently executing utilities are accessing. 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 102  BMCSYNC table  (part 1 of 3)

<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></td>
<td></td>
<td>For 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>NAME1</td>
<td>CHAR(8)</td>
<td>database name or creator name⁴</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For DASD MANAGER PLUS, the value is the database name.</td>
</tr>
<tr>
<td>NAME2</td>
<td>CHAR(18)</td>
<td>space, table, or index name⁴</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For DASD MANAGER PLUS, the BMCSTATS utility always inserts the space name (limited to a maximum of 8 characters).</td>
</tr>
<tr>
<td>KIND</td>
<td>CHAR(2)</td>
<td>type of object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ IP (index partition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ IX (index)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TB (table)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TP (table space partition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TS (table space)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ DD, DW, D1, D2 (dynamic work file allocation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ CI (copy information)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ RD (restart data set block)</td>
</tr>
<tr>
<td>PARTITION</td>
<td>SMALLINT</td>
<td>physical partition number:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ null or 0 for a single data set nonpartitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ data set number for a multi-data-set, nonpartitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ partition number for a partitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY PLUS, LOADPLUS, UNLOAD PLUS, CHECK PLUS, DASD MANAGER PLUS, and REORG PLUS use null or 0 for any nonpartitioned space.</td>
</tr>
<tr>
<td>BMCID</td>
<td>SMALLINT</td>
<td>internal identifier of the object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
</tbody>
</table>
### Table 102 BMCSYNC table (part 2 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| UTILNAME    | CHAR(8)   | name of the executing utility:  
  - CHECK  
  - COPY  
  - STATS  
  - LOAD  
  - RECOVER  
  - REORG  
  - UNLOAD  
| SHRLEVEL   | CHAR(1)   | degree to which utilities can share this object:  
  - Blank means that no status is requested, and any other utility can obtain any status.  
  - S allows sharing among any number of SHRLEVEL S utilities.  
  - X indicates that exclusive control is required. No other utility can run with SHRLEVEL X.  
  For more information, see Table 7 on page 55. |
| STATUS      | CHAR(1)   | status of the utility or object:  
  - blank (indicates no processing has been done)  
  - C (for CHECK PLUS, indicates checked)  
  - L (for LOADPLUS, indicates loaded)  
  - U (for UNLOAD PLUS, indicates unloaded)  
  - R (for REORG PLUS, indicates reloaded)  
  DASD MANAGER PLUS does not use this column. |
| XCOUNT      | INTEGER   | number of rows or keys processed in the current phase  
  DASD MANAGER PLUS does not use this column. |
| DDNAME      | CHAR(8)   | check, load, unload, or work ddname  
  DASD MANAGER PLUS does not use this column. |
| BLOCKS      | INTEGER   | number of blocks for the check, load, unload, or work data set  
  DASD MANAGER PLUS does not use this column. |
| ORIG_STATUS | CHAR(8)   | encoded representation of the original DB2 status of the space  
  For RECOVER PLUS, this column restores the DB2 status of a space after recovery, if necessary.  
  DASD MANAGER PLUS does not use this column. |
| EXTRBA      | CHAR(6)   | (RECOVER PLUS) log point at which this space was externalized  
  Note: RECOVER PLUS no longer uses EXTRBA. |
Considerations for the BMCSYNC table

Note the following considerations:

- 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 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

---

Table 102  BMCSYNC table (part 3 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>LONG VARCHAR</td>
<td>restart information for the space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, the STATE indicates the object state and sync information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td>INSTANCE</td>
<td>SMALLINT</td>
<td>(RECOVERY MANAGER and RECOVER PLUS) the instance number of the current base objects (table and index)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default value is 1. The other utilities do not use this column.</td>
</tr>
</tbody>
</table>

* (LOADPLUS, UNLOAD PLUS, CHECK PLUS, and REORG 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 contains the table OBID or index ISOBID of the object in hexadecimal format.
— 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

- 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.

**Maintaining the BMCSYNC table**

When a utility abends, rows might remain in the BMCSYNC table. If you need to control expansion of this table, use one of the following methods to delete rows:

- Use the TERM restart parameter on the EXEC statement to delete rows from 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.
Table 103 describes the contents of the BMCTRANS table, which contains information that RECOVERY MANAGER and Log Master use for transaction recovery. The table contains one row for each execution of Log Master (one row for each log scan).

<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>STATE³</td>
<td>SMALLINT NOT NULL</td>
<td>level of recovery analysis performed</td>
</tr>
<tr>
<td>PITTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>timestamp for the PIT RBA</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>
</tbody>
</table>
Table 103  BMCTRANS table  (part 2 of 2)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

a  If STATE is 0, only UNDO analysis has been performed. If STATE is 1 through 9999, UNDO and PIT analysis have been performed. If STATE is greater than 10000, UNDO, PIT, and REDO analysis have been performed.

BMCUTIL table

Table 104 describes the BMCUTIL table, which contains information about utilities that are currently running or started. 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 104  BMCUTIL table  (part 1 of 3)

<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>STATUS</td>
<td>CHAR(1)</td>
<td>execution status of the utility:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- A (active, not executing command)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- I (initializing)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- P (pausing or pause-stopped)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- S (stopped)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- T (terminating)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- X (executing command)</td>
</tr>
</tbody>
</table>
Table 104  BMCUTIL table  (part 2 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>PHASE</td>
<td>CHAR(8)</td>
<td>current phase of the utility</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>user ID executing the utility</td>
</tr>
<tr>
<td>SSID</td>
<td>CHAR(4)</td>
<td>DB2 subsystem where the utility is running</td>
</tr>
<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>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></td>
<td></td>
<td>Although UNLOAD PLUS accepts the RESTART, RESTART(PHASE), NEW/RESTART, and NEW/RESTART(PHASE) parameters, the utility executes as though you had specified the NEW parameter.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td>(RECOVER PLUS and REORG PLUS) name of the database containing the table or 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>SPNAME</td>
<td>CHAR(8)</td>
<td>(RECOVER PLUS and REORG PLUS) name of the table or 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>SPSTATUS</td>
<td>CHAR(5)</td>
<td>(REORG PLUS) space status before the utility stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
</tbody>
</table>
Maintaining the BMCUTIL table

When a utility abends, rows might remain in the BMCUTIL table. If you need to control expansion of this table, use one of the following methods to delete rows:

- Use the TERM restart parameter on the EXEC statement to delete rows from 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.
Table 105 describes the contents of the BMCXCOPY table, which the BMC utilities use for tracking 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 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

UNLOAD PLUS can unload data from an encrypted copy that COPY PLUS created. This encrypted copy is registered in the BMCXCOPY table with STYPE e. For more information, see the following documentation:

- Encrypted copy requirements in “IMAGECOPY” on page 120
- COPY PLUS for DB2 Reference Manual

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>
<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>
</tbody>
</table>
### Table 105  BMCXCOPY table (part 2 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICTYPE</td>
<td>CHAR(1)</td>
<td>operation type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- F (COPY FULL YES; for COPY PLUS version 8.1 and later, online consistent copies)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- I (COPY FULL NO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- W (REORG LOG NO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- B (REBUILD INDEX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- P (POINT-IN-TIME RECOVERY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- C (for COPY PLUS version 7.3 and earlier, online consistent copies)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- s (used by COPY PLUS to track system pages)</td>
</tr>
<tr>
<td>ICDATE</td>
<td>CHAR(6)</td>
<td>date of the entry (ymmmdd)</td>
</tr>
<tr>
<td>START_RBA</td>
<td>VARCHAR(10)</td>
<td>the relative byte location of a point in the DB2 recovery log</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The indicated point is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- for ICTYPE F, the starting point for all updates since the image copy was taken</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- for COPY_TYPE O, the minimum of the consistent point and the oldest inflight URID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- (RECOVERY MANAGER) for ICTYPE C, the consistent log point for the copy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— RBA for non-data-sharing systems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— LRSN for data sharing systems</td>
</tr>
<tr>
<td>FILESEQNO</td>
<td>INTEGER</td>
<td>tape file sequence number of the copy</td>
</tr>
<tr>
<td>DEVTYPE</td>
<td>CHAR(8)</td>
<td>type of device on which the copy resides</td>
</tr>
<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>DSNNAME</td>
<td>CHAR(44)</td>
<td>name of the data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If STYPE V, DSNNAME 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>
</tbody>
</table>
### Table 105  BMCXCOPY table (part 3 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSVOLSER</td>
<td>VARCHAR(1784)</td>
<td>volume serial numbers of the data set. 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. This is 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>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'000000000000' (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). 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>CPYPAGESF</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>
</tbody>
</table>
### Table 105  BMCXCOPY table (part 4 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHID</td>
<td>CHAR(8)</td>
<td>authorization ID</td>
</tr>
</tbody>
</table>
| OLDEST_VERSION | SMALLINT  | 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. |
| LOGICAL_PART   | INTEGER   | logical partition number |
| LOGGED         | CHAR(1)   | logging attribute of the table space:  
**Y** (logged)  
**N** (not logged)  
**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. |
| TTYPE          | CHAR(8)   | row format for the table space or partition:  
**RRF** (reordered row format)  
**BRF** (basic row format) |
| INSTANCE       | SMALLINT  | instance number of the current base objects (table and index)  
The default value is 1. |
| RELCREATED     | CHAR(1)   | DB2 release that created the object  
If the release is earlier than version 9, the value is blank. |
| COPY_TYPE      | CHAR(1)   | type of copy:  
**C** (cabinet copy)  
**O** (online consistent copy)  
**blank** (default value) |
| NOTE_VALUE     | CHAR(4)   | encoded value that quickly locates data for a specific space in a cabinet copy  
The default value is blank. |
| NOTE_TYPE      | CHAR(1)   | type of NOTE (issued by COPY PLUS):  
**A** (ABS— tape)  
**R** (REL— disk)  
**F** (frame)  
**blank** (default value) |
| OCC_COPY_RBA   | VARCHAR(10) | original START_RBA of an online consistent copy  
The default value is blank. |
| OCC_LOCKRULE   | CHAR(1)   | locking rule for a table space (not used for indexes):  
**A** (for page level)  
**R** (for row level)  
**blank** (default value) |
Maintaining the BMCXCOPY table

Periodically, you should review BMCXCOPY and delete old rows to control its expansion. To delete all rows from the BMCXCOPY table that are older than 30 days, use the following statement as an example:

```sql
DELETE
FROM creatorName.CMN_BMCXCOPY
WHERE DAYS(CURRENT_TIMESTAMP) - DAYS(TIMESTAMP) > 30;
```

### Table 105 BMCXCOPY table (part 5 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
Generating control statements for DB2 or other software products

This appendix presents the following topics:

Overview ......................................................... 505
Inserting additional command options into utility control statements ............... 506
Control statements to create and load DB2 tables ........................................... 507
  DB2LOAD option .............................................. 508
  DB2 option .................................................. 509
  BMCLOAD option for LOADPLUS for DB2 .............................................. 512
  DB2DDL option .............................................. 514
Control statements for FOCUS processing .................................................... 515
Control statements for SAS processing ......................................................... 518
Control statements to create and load SQL/DS tables .................................... 520
  SQL/DS-LOAD option .......................................... 520
  SQL/DS option .............................................. 522
  SQL/DS-DDL option ........................................... 524
Control statements for Easytrieve processing .............................................. 525
Control statements to load Teradata tables .................................................. 527
  TERADATA-BULK option ....................................... 528
  TERADATA-FAST option ....................................... 531
  TERADATA-MULT option ....................................... 533
  TERADATA option ........................................... 534
Control statements for NOMAD processing .................................................. 535

Overview

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 9 or later
- LOADPLUS version 9.3 or later
- Easytrieve version 5.3
- FOCUS version 6
- NOMAD version 5.5
- SAS version 5
- SQL/DS
- Teradata versions 4.1.1 and 4.1.2

**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 through the use of the following string parameters:

- 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 137.

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.TB1OPART

BMC51810I LOAD DATA INDDN SYSREC
BMC51809I RESUME YES LOG NO NOCOPYPEND
BMC51940I   EBCDIC CCSID(37,65534,65534)
BMC51811I   INTO TABLE RDGTMS.TB1OPART
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:

- load a table that was previously created (DB2LOAD or BMCLOAD option)
- create a new DB2 table and load it (DB2 option)
- create 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 506.
**DB2LOAD option**

The DB2LOAD option allows you to construct a job that can unload data and load it directly with no intervention (see Figure 68 on page 509). 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 Software LOADPLUS product or the IBM® DB2 LOAD utility.

**INTO 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 506.

**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.
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 69 on page 511.

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.
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 506.

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.
Figure 69  Using the DB2 option

1. **UNLOAD PLUS**
   - DB2 table
   - Unloaded data
   - DDL and LOAD control statements

2. User intervention
   - LOAD control statements
   - DDL
   - SPUFI or other tool
   - DB2 catalog

3. **LOADPLUS or DB2 LOAD utility**
   - DB2 table
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 (see Figure 70 on page 514). 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.

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 506.

**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 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.

**Dynamic allocation**

If you specify dynamic allocation, the unload 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.
If you specify CNTLCARDS DB2DDL, UNLOAD PLUS generates appropriate CREATE TABLE DDL. Figure 71 on page 515 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.
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. See Figure 72 on page 517 for an illustration.

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.

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. See the FIXEDVARCHAR option in Chapter 3, “Syntax of the UNLOAD command.”

Figure 72 Using the FOCUS option
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. Figure 73 on page 519 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 VARLENn, where n is incremented for each usage as shown in the following example:

```
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 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.

**Figure 73 Using the SAS option**
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:

- load a table that was previously created (SQL/DS-LOAD option)
- create a new SQL/DS table and load it (SQL/DS option)
- create 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, as illustrated in Figure 74 on page 521.

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 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 question mark (?) character, its associated field is null.

Figure 74  Using the SQL/DS-LOAD 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 75 on page 523.

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 n 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 75 Using the SQL/DS option

1. UNLOAD PLUS
   - DB2 table
   - Unloaded data
   - MVS
   - VM

2. SQL/DS
   - DATA LOAD control statements
   - DDL
   - SQL/DS catalog
   - User intervention

3. SQL/DS
   - DATA LOAD control statements
   - SQL/DS table

Step 1
Step 2
Step 3
### SQL/DS-DDL option

If you specify CNTLCARDS SQL/DS DDL, UNLOAD PLUS generates appropriate CREATE TABLE DDL. The unloading of the data and the creation of the DDL is illustrated in Figure 76.

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.

**Figure 76  Using the SQL/DS-DDL option**
Control statements for Easytrieve processing

If you specify CNTLCARDS EASYTRIEVE, UNLOAD PLUS generates appropriate data definition statements for use with CA Technologies’ CA Easytrieve product. See Figure 77 on page 526 for an illustration of this process.

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 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.
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. See the FIXEDVARCHAR option in Chapter 3, “Syntax of the UNLOAD command” for more information.

Figure 77  Using the EASYTRIEVE option
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 79 on page 530.

When the TERADATA-BULK option is specified, UNLOAD PLUS creates control statements similar to those in Figure 78 for use with the Bulk Data Load utility.

**Figure 78  Example control statements for Teradata Bulk Data Load utility**

```sql
CREATE TABLE tablename
( colname1 coltype(length) NOT NULL,
  colname2 coltype(length)
);

LOGON 0/USERID,PASSWORD;

*```
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 79  Using the TERADATA-BULK option

1. UNLOAD PLUS
2. Teradata control statements
3. Teradata Bulk Data Load
4. Teradata table
5. User intervention
6. Teradata catalog
7. DBC/SQL DDL
8. Bulk Data Load statements
9. Unloaded data
10. DB2 table

Step 1
Step 2
Step 3
TERADATA-FAST option

As illustrated in Figure 81 on page 532, the TERADATA-FAST option generates the statements required to create a Teradata table and load it using the Teradata Fast Data Load utility.

Because this utility is always used to load a new table, the CREATE statement is embedded 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 Figure 80 for use with the Fast Data Load utility.

**Figure 80  Example control statements for Teradata Fast Data Load utility**

```plaintext
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.

Figure 81  Using the TERADATA-FAST option
The TERADATA-MULT option generates the statements needed to create a Teradata table and load it using the Teradata Multiload utility. This option is illustrated in Figure 83 on page 534.

When the TERADATA-MULT option is specified, UNLOAD PLUS creates control statements similar to those in Figure 82 to be used with the Multiload utility:

**Figure 82  Example control statements for the Teradata Multiload utility**

```plaintext
.LOGON 0/USERID,PASSWORD;
/* */
CREATE TABLE tablename
  ( colname1           coltype(length) NOT NULL
 , colname2           coltype(length)
);
/* */
.LAYOUT ddname;
.FIELD colname1         inputposition coltype(length);
.FIELD colname2         inputposition coltype(length);
.DML LABEL INSERT;
INSERT INTO tablename.*;
.IMPORT INFILE ddname LAYOUT ddname APPLY INSERT;
/* */
.END MLOAD;
.LOGOFF;
```

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 Multiload 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.
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. See Figure 84 on page 536 for an illustration of 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 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.
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. See the “FIXEDVARCHAR” on page 174 for more information.

Figure 84 Using the NOMAD option
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. For details, see page 176.
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

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.

NOTE

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.

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.
Sample user exits

Source code for the sample UNLOAD PLUS user exits is distributed in the UNLOAD PLUS HLQ.LLQ.SAMP library (where HLQ is the high-level qualifier specified during installation and LLQ is the low-level qualifier or prefix set during installation). Table 106 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>page 548</td>
</tr>
<tr>
<td>ADUEXIC</td>
<td>C (standard) sample exit</td>
<td>UNLOADEXIT ADUEXIC C</td>
<td>page 565</td>
</tr>
<tr>
<td>ADUEXIL</td>
<td>LE C sample exit</td>
<td>UNLOADEXIT ADUEXIL LE_C</td>
<td>page 585</td>
</tr>
<tr>
<td>ADUEXITC</td>
<td>LE COBOL sample exit</td>
<td>UNLOADEXIT ADUEXITC LE_COBOL</td>
<td>page 605</td>
</tr>
<tr>
<td>ADUEXTC2</td>
<td>COBOL II sample exit</td>
<td>UNLOADEXIT ADUEXTC2 COBOL2</td>
<td>page 608</td>
</tr>
</tbody>
</table>

Copy the appropriate member as the base code for your exit routine, then 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 address and miscellaneous data items that can be used in the exit, message areas for user messages, and 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 on page 540. For pseudo-SQLDA for each language, see the following references:

- The Assembler pseudo-SQLDA is illustrated on page 546.
- The C pseudo-SQLDA is illustrated on page 563.
- The COBOL II and LE COBOL pseudo-SQLDA is illustrated on page 603.
Function codes and return codes

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 page 560). For a COBOL exit, the value of the function code is contained in the first field of the EXITPARMS record (see page 601). 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 page 544.

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. Table 107 describes the function codes and the appropriate return codes.

Table 107  Function and return codes

<table>
<thead>
<tr>
<th>Function code</th>
<th>Return code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (process a record)</td>
<td>0 Write this record.  4 Discard this record.  Any other code terminates the utility.</td>
</tr>
<tr>
<td>1 (table initialization call)</td>
<td>0 Process this table with the exit.  4 Do not use the exit for this table.  Any other code terminates the utility.</td>
</tr>
<tr>
<td>2 (termination call)</td>
<td>Return codes are ignored.</td>
</tr>
</tbody>
</table>

See Figure 98 on page 601 for an illustration of the COBOL II and LE COBOL EXITPARM record relationships, and Figure 88 on page 548 for an illustration of 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:

- The Assembler pseudo-SQLDA is described on Figure 87 on page 546.
- The C pseudo-SQLDA is described in Figure 91 on page 563 and Figure 95 on page 584.
The COBOL pseudo-SQLDA is described in Figure 99 on page 603 and Figure 100 on page 603.

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 108 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 109 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 110 on page 542 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>
<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>an &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 110 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 108 on page 541 for a description of the 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>
Figure 85  User exit pointer relationships to SQLDA

Field is Null

Field 1  Field 2  Field 3  ?  .......

Output record

SQLVARN (1)
SQLTYPE
SQLLEN
SQLPRCSN
SQLSCALE
SQLDATA
SQLIND
SQLNAME

SQLVARN (2)
SQLTYPE
SQLLEN
SQLPRCSN
SQLSCALE
SQLDATA
SQLIND
SQLNAME

SQLVARN (3)
SQLTYPE
SQLLEN
SQLPRCSN
SQLSCALE
SQLDATA
SQLIND
SQLNAME

Occurs SQLD times

User exit pointer to SQLDA

SQLDA
SQLN
SQLD

UNLOAD PLUS pseudo-SQLDA
UNLOAD PLUS Assembler user exit

This section provides a sample exit parameter block and exit that you can use to write a user exit in assembler.

Exit parameter block DSECT

Figure 86 illustrates the Assembler exit parameter block DSECT:

Figure 86  Assembler exit parameter block (part 1 of 2)

```assembly
* ADUXEXITP 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.
**********************************************************************
*
ADUXEXITP 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.
*
* 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
```
**Figure 86  Assembler exit parameter block (part 2 of 2)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPSQLDA@</td>
<td>the address of the pseudo-SQLDA for this table as shown in Figure 87 on page 546</td>
</tr>
<tr>
<td></td>
<td>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</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>XPRE#</td>
<td>an 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 XPRE# 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>XPU#</td>
<td>pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility</td>
</tr>
<tr>
<td>XP#</td>
<td>pointer to the 16-byte utility ID</td>
</tr>
</tbody>
</table>

---

**Table 111 explains the major DSECT fields and their uses.**

**Table 111  Assembler user exit parameter block DSECT fields (part 1 of 2)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPSQLDA@</td>
<td>the address of the pseudo-SQLDA for this table as shown in Figure 87 on page 546</td>
</tr>
<tr>
<td></td>
<td>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</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>XPRE#</td>
<td>an 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 XPRE# 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>XPU#</td>
<td>pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility</td>
</tr>
<tr>
<td>XP#</td>
<td>pointer to the 16-byte utility ID</td>
</tr>
</tbody>
</table>
Figure 87 illustrates the Assembler exit pseudo-SQLDA DSECT.

** 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    OF                  BEGIN VARS
SQLDSIZ EQU  *-SQLDA             SIZE OF FIXED PART
SQLVARN DSECT ,                   NTH VARIABLE
SQLTYPE DS    H                   TYPE CODE
SQLLEN DS    OH                  NAME LENGTH
SQLPRCSN DS    X                  DEC PRECISION
SQLSCALE DS    X                  DEC SCALE
SQLDATA DS    A                  ADDR OF DATA IN OUTPUT RECORD
SQLIND DS    A                  ADDR OF NULL FIELD BYTE (?) IF NULL
SQLNAME DS    H,CL30             DESCRIBE NAME
ORG    SQLNAME
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. **Table 108 on page 541** summarizes the field names, data types, and usage notes for each field in the SQLDA portion of the DSECT.

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. **Table 109 on page 541** summarizes the field names, data types, and usage notes for each field in this portion of the DSECT.

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. **Table 110 on page 542** describes the valid SQLTYPE field values for each data type supported.

See **Figure 88 on page 548** for an illustration of the Assembler parameter control block relationships.
Figure 88  Major Assembler user exit pointer relationships

Figure 89 is a sample user exit written in Assembler.

Figure 89  Sample Assembler user exit (part 1 of 13)

```
ADUEUEUE TITLE 'ADUEUEUE - UNLOAD PLUS USER EXIT USER EXAMPLE - V13'
ADUEUEUE CSECT
ADUEUEUE AMODE 31
ADUEUEUE RMODE 24
*****************************************************************************
*  DISCLAIME 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
**********************************************************************

Notes
**********************************************************************

ADUQUEUE 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 ADUXITP 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 shareable 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 -
**Figure 89  Sample Assembler user exit (part 4 of 13)**

```
* 4 - A(COMMON AREA) GETMAIN AREA
* 5 - A(DCB) USED FOR OUTPUT OF SELECTED RECORDS BY THE EXIT
* 6 -
* 7 -
* 8 -
* 9 -
* 10 - A(ADUXITP 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                  R
*                  R1       EQU   1                   E
*                  R2       EQU   2                    G
*                  R3       EQU   3                     I
*                  R4       EQU   4                      S
*                  R5       EQU   5                       T
*                  R6       EQU   6                        E
*                  R7       EQU   7                         R
*                  R8       EQU   8
*                  R9       EQU   9                           E
*                  R10      EQU   10                           Q
*                  R11      EQU   11                            U
*                  R12      EQU   12                             A
*                  R13      EQU   13                              T
*                  R14      EQU   14                               E
*                  R15      EQU   15                                S
*
**********************************************************************
* 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.
**********************************************************************
*
ADUXITP 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
```
### Sample Assembler user exit (part 5 of 13)

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPREF#</td>
<td>DS</td>
<td>BLOCK REFERENCE#</td>
</tr>
<tr>
<td>XPSSID@</td>
<td>DS</td>
<td>A(SSID) 4 BYTES</td>
</tr>
<tr>
<td>XUSER@</td>
<td>DS</td>
<td>A(USERID) 8 BYTES</td>
</tr>
<tr>
<td>XPUTID@</td>
<td>DS</td>
<td>A(UTILITY ID) 16 BYTES</td>
</tr>
<tr>
<td></td>
<td>DS</td>
<td>RESERVED FOR UNLOAD PLUS</td>
</tr>
</tbody>
</table>

* **USER AREA**

* XPUSRMAG CONTAINS A SINGLE MESSAGE ENTRY. ON RETURN FROM THE EXIT, IF THIS FIELD IS NON-BLANK, IT IS PRINTED THEN BLANKED.
* XPUSRMAG@ POINTS TO A MESSAGE BUFFER THAT CAN CONTAIN MULTIPLE EXIT MESSAGES OF EQUAL LENGTH. THESE MESSAGE(S) WILL BE PRINTED AFTER XPUSRMAG (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

* XUSERW@ DS F  USER WORK AREA ADDRESS
* XUSER1 DS F  USER FIELD
* XUSER2 DS F  USER FIELD
* XUSER3 DS F  USER FIELD
* XUSER4 DS F  USER FIELD
* XUSER5 DS F  USER MESSAGE BUFFER ADDRESS
* XUSER6 DS F  USER MESSAGE BUFFER TOTAL SIZE
* XPUSRMAG DS CL100  USER SINGLE MESSAGE AREA

* XPFLAGS DS X  VARIOUS FLAGS FOR EXIT
* XPDEBUG EQU X'01'  DEBUG IS ON

* **USER WORK SPACE**

* XPUSER DS OF  USER WORK AREA

* XPUSERPAD DS (1024-(*-ADUEXITP))C  PADDING
* XPUSER$ EQU *-XUSER
* XP$ EQU *-ADUEXITP

* **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
Figure 89  Sample Assembler user exit (part 6 of 13)

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLN</td>
<td>DS H TOTAL VARS</td>
</tr>
<tr>
<td>SQLD</td>
<td>DS H PERTINENT VARS</td>
</tr>
<tr>
<td>SQLVAR</td>
<td>DS OF BEGIN VARS</td>
</tr>
<tr>
<td>SQLDSIZ</td>
<td>EQU *-SQLDA SIZE OF FIXED PART</td>
</tr>
<tr>
<td>SQLVARN</td>
<td>DSECT NTH VARIABLE</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>DS H TYPE CODE</td>
</tr>
<tr>
<td>SOLLEN</td>
<td>DS OH NAME LENGTH</td>
</tr>
<tr>
<td>SQLPRCSN</td>
<td>DS X DEC PRECISION</td>
</tr>
<tr>
<td>SQLSCALE</td>
<td>DS X DEC SCALE</td>
</tr>
<tr>
<td>SQLDATA</td>
<td>DS A ADDR OF DATA IN OUTPUT RECORD</td>
</tr>
<tr>
<td>SQLIND</td>
<td>DS A ADDR OF NULL FIELD BYTE (?? IF NULL)</td>
</tr>
<tr>
<td>SQLNAME</td>
<td>DS H,CL30 DESCRIBE NAME</td>
</tr>
<tr>
<td>SQLNAM$</td>
<td>DS H SIZE OF COL NAME</td>
</tr>
<tr>
<td>SQLNAM</td>
<td>DS CL30 NAME ALONE</td>
</tr>
<tr>
<td>SQLVSI$</td>
<td>EQU *-SQLVARN</td>
</tr>
</tbody>
</table>

**********************************************************************
* 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
TBNAME  DS CL128 TABLE NAME
SPARES  DS XL926
WRKAREA$ EQU *-WORKAREA
EJECT

**********************************************************************
* 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 ADUXITP AND IS COPIED *
* TO EACH ADUXITP 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 ARE LOCKED
*
COMNREC# DS F OUTPUT RECORD COUNT
COMNDBLW DS D DOUBLE WORD
COMMUNPK DS CL16 UNPACK AREA
COMNDATE DS CL10 DATE HOLDING AREA
*
COMNFLAG DS XL1 FLAG WORD
COMN#FND EQU X'80' ..GAMES_BEHIND COLUMN FOUND
COMN#KEP EQU X'40' ..PROCESS/DONT WRITE RECORD
COMN#DIS EQU X'20' ..DONT PROCESS/WRITE RECORD
Figure 89  Sample Assembler user exit (part 7 of 13)

```
COMN#OFF EQU X'00'  ..FLAG RESET
   DS XL3     SPARE
   DS 4F     SPARE
*
COMNMSGA DS 0F      PROCESSING MESSAGE AREA
COMNMSG1 DS CL28    MESSAGE AREA 1
COMNMS1$ EQU *COMNMSG1 MESSAGE AREA 1 LENGTH
COMNMSG2 DS CL18    MESSAGE AREA 2
COMNMS2$ EQU *COMNMSG2 MESSAGE AREA 2 LENGTH
COMNMSG3 DS CL9     MESSAGE AREA 3
COMNMS3$ EQU *COMNMSG3 MESSAGE AREA 3 LENGTH
COMNMSGP DS CL45    MESSAGE AREA PAD
COMNMSG$ EQU *COMNMSGA MESSAGE AREA TOTAL LENGTH
*
COMNRECA DS CL80    DISCARD RECORD AREA
*
COMNOPNL OPEN (0),MF=L LIST FORM OF OPEN
   DS 4F     SPARE
*
COMNCLSL CLOSE (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 ADDRESSABILITY
*
   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
```
Appendix D  UNLOAD PLUS user exits

Figure 89  Sample Assembler user exit (part 8 of 13)

<table>
<thead>
<tr>
<th>LA</th>
<th>R15, SAVE</th>
<th>MY SAVE AREA@</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST</td>
<td>R15, 8(R13)</td>
<td>SAVE IN CALLERS SAVE AREA</td>
</tr>
<tr>
<td>ST</td>
<td>R13, 4(R15)</td>
<td>SAVE CALLERS SAVE AREA@ IN MINE</td>
</tr>
<tr>
<td>LR</td>
<td>R13, R15</td>
<td>SET OURS CURRENT</td>
</tr>
<tr>
<td>EJECT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* CHECK THE FUNCTION CODE AND BRANCH TO PROCESSING ROUTINE

<table>
<thead>
<tr>
<th>CHKF0000 DS OH</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>R2, MAXFUNC</td>
</tr>
<tr>
<td>BNH</td>
<td>CHKF0010</td>
</tr>
<tr>
<td>MVC</td>
<td>XPUSRMSG(L'MBADFUNC), MBADFNC INDICATE BAD FUNCTION CODE</td>
</tr>
<tr>
<td>B</td>
<td>RETC0008</td>
</tr>
</tbody>
</table>

* CHKF0010 SLL R2, 2    CONVERT FOR BRANCHING

| B                      | **+4(R2)                  |
| B                      | PROC0000                  |
| B                      | INIT0000                  |
| B                      | CLEAN0000                 |
| EJECT                  |                             |

* INITIALIZATION ROUTINE

<table>
<thead>
<tr>
<th>INIT0000 DS OH</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>R15, XPUTID@</td>
</tr>
<tr>
<td>CLC</td>
<td>=C'DEBUG', 0(R15)</td>
</tr>
<tr>
<td>BNE</td>
<td>INIT0010</td>
</tr>
<tr>
<td>OI</td>
<td>XPFLAGS, XPFDEBUG</td>
</tr>
</tbody>
</table>

* INIT0010 DS OH         |                             |

| L                      | R3, XPTABLE@               |
| MVC                    | TBCREAT, 0(R3)             |
| MVC                    | TBNAME, 128(R3)            |
| CLC                    | TBNAME, BBALLTBL           |
| BNE                    | RETC0004                   |

* GETMAIN RC, LV= COMNAREL, GET COMMON AREA FOR IO * LOC=BETWEN BELOW 16M

| LTR                    | R15, R15                   |
| BNZ                    | INIT9900                   |
| ST                     | R1, XPUSERW@               |
| LR                     | R4, R1                     |
| USING                  | COMNAREA, R4               |

* MVI COMNAREA, X'00'  INITIALIZE GETMAIN AREA

| L                      | R1, =A(COMNAREL-1)         |
| LA                     | R0, COMNAREA+1             |
| LA                     | R14, COMNAREA              |
| LA                     | R15, 1                     |

Sample Assembler user exit
Figure 89   Sample Assembler user exit (part 9 of 13)

```
MVCL R0,R14                   PERFORM MOVE
MVC COMNODCB(COMNODC$),OUTDCB INITIALIZE COMMON OUTPUT DCB
LA R5,COMNODCB              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  XPFLOGS,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  OH             GETMAIN ERROR
  MVC  XPUSRMSG(L'MGETMERR),MGETMERR
  B  RETC0008                TERMINATE THE EXECUTION

  INIT9910 DS  OH             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(SYSSREC) BY UNLOADPLUS.
******************************************************************************

PROC0000 DS  OH
  ICM  R4,B'1111',XPUSERW@  GET A(COMMON AREA)
  BZ  PROC9900             NO COMMON AREA - ERROR

*  
PROC0010 DS  OH
  LA  R14,COMN#FRE         SERIALIZE ON COMMON AREA
  LA  R15,COMN#HLD
  CS  R14,R15,COMN#OKW     ATTEMPT LOCK
  BE  PROC0020            GOT IT, CONTINUE
  CALLDISP ,             RELEASE CPU
  B  PROC0010            TRY AGAIN

*  
```
Sample Assembler user exit

Figure 89  Sample Assembler user exit (part 10 of 13)

```
PROC0020 DS 0H
   L R8,XPSQLDA@ GET SQLDA ADDRESS
   USING SQLDA,R8
   LH R9,SQLN GET NUMBER OF COLUMNS IN RECORD
   LA R8,SQLDSIZ(R8) PASS FIXED AREA
   USING SQLVARN,R8
*
PROC0030 DS 0H COLUMN LOOP
   CLC SQLNAM,DTCLNAME DATE COLUMN?
   BNE PROC0040 ..NO, CHECK FOR GAMES BEHIND
   L R7,SQLDATA GET A(DATE FIELD)
   MVC COMNRECA(80),0(R7) SAVE ENTIRE RECORD
   MVC COMNDATE,0(R7) SAVE DATE VALUE
   B PROC0050 NOW FIND GAMES BEHIND FIELD
*
PROC0040 DS 0H 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 0H GAMES BEHIND NOT FOUND - FATAL
   LA R8,SQLVSIZ(R8) ..NO, POINT TO NEXT FIELD
   BCT R9,PROC0030 LOOP
   B PROC9900 GAMES BEHIND NOT FOUND - FATAL
*
* PROCESS RECORD IN UNLOAD - DONT WRITE RECORD IN EXIT
*
PROC0060 DS 0H
   OI COMNFLAG,COMN#KEP SET KEEP FLAG
   B PROC9000 GO FINISH UP
*
* DONT PROCESS RECORD IN UNLOAD - WRITE RECORD IN EXIT
*
PROC0070 DS 0H
   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 XFLAGS,XPDEBUG  IS DEBUG ON?
  BNO PROC9050  ..NO, SKIP MESSAGE PROCESSING
  CVD R1,COMNDBLW  CONVERT RECORD COUNT TO DECIMAL
  MVI COMNUNPK,X'40'  BLANK 1ST BYTE OF UNPK AREA
  MVC COMNUNPK+1(L'COMNUNPK-1),COMNUNPK  AND PROPAGATE
  MVI COMNMSGA,X'40'  BLANK 1ST BYTE OF MSGAREA
  MVC COMNMSGA+1(COMNMSG$-1),COMNMSGA  AND PROPAGATE
  UNPK COMNUNPK,COMNDBLW  UNPACK RECORD COUNT
  OI COMNUNPK+15,X'F0'  TURN OFF SIGN
  MVC COMNMSG1(COMNMS1$),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
  B PROC9020

* 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
*
Figure 89  Sample Assembler user exit (part 12 of 13)

```
RETC0004 DS  OH
  LA  R3,4           SAVE RETURN CODE
  B   RET9000        DON'T PROCESS/DISCARD
  *
RETC0008 DS  OH
  LA  R3,8           SAVE RETURN CODE
  *
  TERMINATE UNLOAD+ EXEC
RETC0008 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  COMNL0KW,=A(COMN#FRE) RELEASE LOCK ON COMMON AREA
  *
RETC0008 DS  OH
  L   R13,SAVE+4    FREE LOCAL WORK AREA
  FREEMAIN RC,LV=WRKAREA$,A=(R11) FREE LOCAL WORK AREA
  LR  R15,R3        RESTORE RETURN CODE
  L   R14,12(,R13)
  LM  R0,R12,20(R13)
  BSM  0,R14        RETURN
  EJECT

* CLEANUP (FUNCTION CALL = 2)

CLEAN000 DS  OH
  TM  XPFLAGS,XPFDEBUG IS DEBUG ON?
  BNO  CLEAN010 ..NO, SKIP MESSAGE
  MVC  XPUSRMSG(L'IGOT2CLN),IGOT2CLN SAY I GOT HERE
  *
CLEAN000 DS  OH
  ICM  R4,B'1111',XPUSERW@ GET A(COMMON AREA)
  BZ  CLEAN020 NO @, NO CLOSE OR FREEMAIN
  LA  R5,COMNODCB GET A(OUTPUT DCB)
  MVI  COMNCLSL,X'80' SET END OF LIST
  CLOSE ((5),REREAD),MF=(E,COMNCLSL) ISSUE CLOSE
  *
  FREEMAIN RC,LV=COMNAREL,A=(R1) FREE STORAGE
  *
CLEAN020 DS  OH    RETURN
  L   R13,SAVE+4    FREE LOCAL WORK AREA
  FREEMAIN RC,LV=WRKAREA$,A=(R11) FREE LOCAL WORK AREA
  L   R14,12(,R13)
  LM  R0,R12,20(R13)
  LA  R15,0
  BSM  0,R14
  EJECT

* CONSTANTS
```

Appendix D  UNLOAD PLUS user exits  559
This section provides a sample exit parameter structure and exit that you can use to write a user exit in C.

C exit parameter structure

Figure 90 illustrates the C exit parameter structure. The ADUEXITP structure is contained in a header file, aduxith0.h.
### C exit parameter structure

```c
# define XPFLAGS                         xpflags
# define XPFUNC                          xpfunc
# define XPREFP_                         xprefP_
# define XPRESRV                         xpresrv
# define XPSQLDAA_                       xpsqldaA_
# define XPSSIDA_                        xpssidA_

int xfunc;                          /* 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) 4 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.*/
/*                                   */
/* 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:*/
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
int xfunc;                          /* USER WORK AREA ADDRESS */
int xuserf1;                         /* USER FIELD */
int xuserf2;                         /* USER FIELD */
int xuserf3;                         /* USER FIELD */
int xuserf4;                         /* USER FIELD */
void *xusermA_;                      /* USER MESSAGE BUFFER ADDRESS */
int xusermz;                         /* USER MESSAGE BUFFER TOTAL SIZE */
CL100 xusrmmsg;                      /* USER SINGLE MESSAGE AREA */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
/*                                   */
XL2 xpresrv;                         /* RESERVED */
/*                                   */
char xusrpad(|842|);                 /* PADDING */
#define XPK 1024                        /* */

#define XPFLAGS                         xpflags
#define XPFUNC                          xpfunc
#define XPREFP_                         xprefP_
#define XPRESRV                         xpresrv
#define XPSQLDAA_                       xpsqldaA_
#define XPSSIDA_                        xpssidA_
```

---

**Figure 90** C user exit parameter structure (part 2 of 3)
#define XPTABLEA_    xptableA_
#define XPUSERA_     xpuserA_
#define XPUSERF1     xpuserf1
#define XPUSERF2     xpuserf2
#define XPUSERF3     xpuserf3
#define XPUSERF4     xpuserf4
#define xpusermz     xpusermz
#define XPUSERMA_    xpusermA_
#define XPUSERWA_    xpuserwA_
#define XPUSRMSG     xpusrmsg
#define XUSRSPAD     xpusrpad
#define XPUTIDA_     xputidA_

/*
SYMBOL   OFFSET   SIZE      TYPE   C-TYPE  C-NAME
ADUEXITP 000000   000000   X       char    xpflags
XPFLAGS   00000C   000001   F       int     xfunc
XPFUNC   000000   000004   F       int     xfunc
XPREF#   00000E   000002   H       short   xprefP_
XPRESRV   0000B4   000002   XL2     XL2     xpresrv
XPUSQLDA@ 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     xfunc
XPUSERF2  00003C  000004   F       int     xfunc
XPUSERF3  000040  000004   F       int     xfunc
XPUSERF4  000044  000004   F       int     xfunc
XPUSERM@  000048  000004   A       void *  xusermz
XPUSERM@  000048  000004   A       void *  xusermz
XPUSERW@  00003A  000004   A       void *  xpuserwA_
XPUSRMSG  000050  000064   CL100   CL100  xpusrmsg
XPUSRSPAD 0000B6  00034A   (1024-... char (|842|) xpusrpad
XPUTID@  000018  000004   A       void *  xputidA_
*/

Table 112 on page 563 explains major parameter structure elements and their uses.
C pseudo-SQLDA structure

Figure 91 illustrates the C exit pseudo-SQLDA.

**Table 112  C user exit parameter fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*xsqlldaA_</td>
<td>pointer to the pseudo-SQLDA for this table as shown in Figure 91. 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. 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. 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>*xssidA_</td>
<td>pointer to a 4-byte field containing the SSID value</td>
</tr>
<tr>
<td>*xuserA_</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>
<tr>
<td>CL100 xusrmsg</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>*xusermA_</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 XUSRMSG message.</td>
</tr>
<tr>
<td>xpuserm$</td>
<td>must be set to the size of the user message buffer pointed to by *xusermA_</td>
</tr>
</tbody>
</table>

**C pseudo-SQLDA structure**

**Figure 91  C pseudo-SQLDA record (part 1 of 2)**

```
struct SQLDA
{
  CL8 sqldaid;    /* ID */
  int sqldabc;    /* BYTE COUNT */
  short sqln;     /* TOTAL VARS */
  short sqld;     /* PERTINENT VARS */
}```
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. Table 108 on page 541 summarizes the field names, data types, and usage notes for each field in the SQLDA portion of the structure.

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. Table 109 on page 541 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 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. Table 110 on page 542 describes the valid SQLTYPE field values for each data type supported.

See Figure 92 on page 565 for an illustration of the C parameter control block relationships.
Sample C user exit

Figure 93 is a sample user exit written in C.

---

```c
/**
 * IDENTIFICATION AND DESCRIPTION
 * 
 * PGM:       ADUEXIC
 * 
 * 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. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* prototypes and #defines */

#define ADU_MVS

/* defines */

#if defined(ADU_MSC)
  #define __asm
  #define _BYREF
#else
  #define _BYREF @
#endif

/* prototypes */

__asm void aduxitr1(int,struct LVAREA*);

#if !defined(_XL2)
  typedef struct
  {
    #ifdef ADU_MVS
      unsigned BF : 16;
    #else
      unsigned BF1 : 8;
      unsigned BF2 : 8;
    #endif
    } XL2;
  #endif
#endif

/* Figure 93 Sample C user exit (part 2 of 16) */

* Sample C user exit
* Note that all non-standard C headers are included at the front of this code.
* *
* *********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* prototypes and #defines */

#define ADU_MVS

/* defines */

#if defined(ADU_MSC)
  #define __asm
  #define _BYREF
#else
  #define _BYREF @
#endif

/* prototypes */

__asm void aduxitr1(int,struct LVAREA*);

#if !defined(_XL2)
  typedef struct
  {
    #ifdef ADU_MVS
      unsigned BF : 16;
    #else
      unsigned BF1 : 8;
      unsigned BF2 : 8;
    #endif
    } XL2;
  #endif
#endif
#if !defined(_CL100)
#define _CL100
typedef char CL100(|100|);
#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) */
  /*********************************************************************/
  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 */
  short _f0; /* */
  short xprefP_; /* BLOCK REFERENCE # */
  void *xpssidA_; /* A(SSID) 4 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. */
  /*
  * 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*/
  /*
  void *xuserwa_; /* USER WORK AREA ADDRESS */
  int xuserf1; /* USER FIELD */
  int xuserf2; /* USER FIELD */
  int xuserf3; /* USER FIELD */
  int xuserf4; /* USER FIELD */
  void *xuserma_; /* USER MESSAGE BUFFER ADDRESS */
  int xusermz; /* USER MESSAGE BUFFER TOTAL SIZE */
  CL100 xusermsg; /* USER SINGLE MESSAGE AREA */
  /*
  XL2 xpresrv; /* RESERVED V12*/
  /**/
};
char xpusrpad([842]); /* PADDING */
#define XPK 1024 /* */
};

#define XPFLAGS xpflags
#define XPFUNC xpfnc
#define XPREFP_ xprefP_
#define XPRESRV xpresrv
#define XPSQLDA_ xpsqldaA_
#define XPSSID_ xpssida_
#define XPTABLEA_ xptableA_
#define XPUSERA_ xpuserA_
#define XPUSERF1 xpuserf1
#define XPUSERF2 xpuserf2
#define XPUSERF3 xpuserf3
#define XPUSERF4 xpuserf4
#define xpusermz xpusermz
#define XPUSERMA_ xpusermA_
#define XPUSERWA_ xpuserwa_
#define XPUSRMSG xpusrmsg
#define XPUSRPAD xpusrpad
#define XPUTIDA_ xputidA_

#define XPUTIDA_ xputidA_

/#include "aduxith2.h" lvarea

/*-----------------------------------------*/
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
ADUXITP 000000 000000
XPFLAGS 00000C 000001 X char xpflags
XPFUNC 000000 000004 F int xpfnc
XPREF# 00000E 000002 H short xprefP_
XPRESRV 000004 000002 XL2 XL2 xpresrv
XPSQLDA@ 000004 000004 A void * xpsqldaA_
XPSSID@ 000010 000004 A void * xpssida_
XPTABLE@ 0000E8 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 00004C 000004 F int xpusermz
XPUSERMA_ 000048 000004 A void * xpusermA_
XPUSERWA_ 000034 000004 A void * xpuserwa_
XPUSRMSG 000050 000064 CL100 CL100 xpusrmsg
XPUSRPAD 0000B6 000004 (1024-... char ([842]) xpusrpad
XPUTIDA_ 000018 000004 A void * xputidA_
*/

/*-----------------------------------------*/
```c
#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 {
    #ifdef ADU_MVS
    unsigned BF :16;
    #else
    unsigned BF1 :8;
    unsigned BF2 :8;
    #endif
} PL2;
#endif

struct LVAREA {
    /*********************************************************************/
    /* LOCAL VARIABLES WORK AREA DSECT */
    /*********************************************************************/
    int lvsave(|18|);    /* LOCAL SAVE AREA */
    PL2 lvkeepP_;        /* RECORDS KEPT */
    PL2 lvdiscP_;        /* RECORDS DISCARD */
    #define LVBLANKA 76   /* START OF BLANKING AREA */
    CL6 lvkeep;          /* UNPKED KEEPERS */
    CL6 lvdisc;          /* UNPKED DISCARDS */
    CL128 lvtbcrea;      /* TABLE CREATOR */
    CL128 lvtbodyname;   /* TABLE NAME */
    #define LVBLANKZ 38   /* SIZE OF BLANKING AREA */
    XL292 lvspares;      /* */
    #define LVAREAZ 406   /* */
```


Figure 93 Sample C user exit (part 6 of 16)

```c
#define LVDISC  lvdisc
#define LVDISCP_ lvdiscP_
#define LVKEEP  lvkeep
#define LVKEEPP_ lvkeepP_
#define LVSAVE  lvsave
#define LVSPARES lvspares
#define LVTBCREA lvtbcrea
#define LVTBNAME 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
LVTBNAME 000060  000012 CL128 CL128 lvtbname
*/

/*---------------------------------------*/
/* #include "aduxith3.h" SQLDA Header */
/*---------------------------------------*/

#if !defined(_CL8)
#define _CL8
typedef char CL8(|8|);
#endif

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
```
```c
#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  sqlda-id
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 CL30(|30|);
#endif

struct SQLVARN
{
    short sqltype;          /* TYPE CODE     */
#if defined 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;
    };
};
```

**Figure 93** Sample C user exit (part 7 of 16)
/*--------------------*/
/* end SQLVARN HEADER */
/*--------------------*/

#define SQLDATA sqldata
#define SQLIND sqind
#define SQILLEN sqllen(|0|)
#define SQLNAM _s1.sqlnam
#define sqnamz _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 * sqind
SILLEN 000002 000000 0H short (|0|) sqllen
SQLNAM 00000E 00001E CL30 CL30 _s1.sqlnam
sqlnamz 00000C 00002 H short _s1.sqlnamz
SQLNAME 00000C 00002 H short _s0.sqlname
SQLPRCSN 000002 00001 X char sqlprcsn
SQLSCALE 000003 00001 X char sqlscale
SQLTYPE 000000 00002 H short sqltype
SQLVARN 000000 00000
*/

static int BadFuncCode(struct ADUEXITP *pzAduexitp);

#define MAXFUNC 2
#define PROCESS 0
#define INITIAL 1
#define CLEANUP 2

static char
szAccept || = "ACCEPTED"
.szDiscard || = "DISCARDED"
.szMbafunc || = "BAD FUNCTION CODE RECEIVED FROM MAIN"
.szMgetmerr || = "GETMAIN ERROR GETTING USER STORAGE"
.szMgetmer2 || = "GETMAIN ERROR GETTING MSG STORAGE"
.sz1got2ini || = "COLUMNS AVAILABLE FROM TABLE"
.sz1got2pro || = "PROCESSING REGISTRATION ID:"
.sz1got2cln || = "PROCESSING IS COMPLETED"
;

int aduexic( long lFcode, struct ADUEXITP *pzAduexitp )
{
    int lRc;
}
Figure 93  Sample C user exit (part 9 of 16)

```c
/*-------------------------------*
 *  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 *pzSqlda;
struct SQLVARN *pzSqlvarn;

if ( (pstrMsg=pzAduexitp->XPUSERMA_)!=NULL ){
    *(short*)pstrMsg = 0;
}

/* Say I got here */
memcpy(pzAduexitp->XPUSRMSG ,szIgot2pro ,sizeof(szIgot2pro));

pzSqlda = pzAduexitp->XPSQLDAA_; 

sCols = pzSqlda->SQLN;
pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqlda)+SQLDSIZ);

while(sCols){
    if ( memcmp(pzSqlvarn->SQLNAM ,"REGISTRATION" ,12)==0 ){
        memcpy(pzAduexitp->XPUSRMSG+sizeof(szIgot2pro)
            ,pzSqlvarn->SQLDATA ,8);
        if ( (memcpy(pzSqlvarn->SQLDATA ,"NCC-1700" ,8)>=0)
            && (memcpy(pzSqlvarn->SQLDATA ,"NCC-1711" ,8)<=0) ){
            /* We're starting over with columns */
            sCols = pzSqlda->SQLN;
pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqlda)+SQLDSIZ);
            while(sCols){
                if ( memcmp(pzSqlvarn->SQLNAM ,"NAME ",5)==0){
                    if ( (pzSqlvarn->SQLIND == NULL)
                        || (pzSqlvarn->SQLIND != NULL)
                        && (*((char*)pzSqlvarn->SQLIND)=='?')){
                        memcpy(pzAduexitp->XPUSRMSG
                            +sizeof(pzAduexitp->XPUSERWA_)-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 ! */
                    }
                } else {
                    aduxitr1(1,pzAduexitp->XPUSERWA_); /* bump discard count */
                }
                pzSqlvarn++;
                sCols--;
            }
            memcpy(pzAduexitp->XPUSRMSG ,"NAME COLUMN NOT FOUND" ,21);
            return 8;   /* Q U I T ! */
        } else {
            aduxitr1(1,pzAduexitp->XPUSERWA_); /* bump discard count */
        }
    }
}
### Sample C user exit (part 11 of 16)

```c
memcpy(pzAduexitp->XPUSRMSG + sizeof(pzAduexitp->XPUSRMSG) - sizeof(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);
```
Sample C user exit

Figure 93  Sample C user exit (part 12 of 16)

```c
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));

pzSqlda = pzAduexitp->XPSQLDAA_;  
SMsgs = pzSqlda->SQLN * sizeof(pzAduexitp->XPUSRMSG);  
SMsgs += 4+256;  /* adjust for prefix and pad */
{  
pzAduexitp->xpusermz = (int)sMsgs;  
if ( (pzAduexitp->XPUSERMA_=calloc(1,sMsgs))==NULL){  
  return GetmainError(2,pzAduexitp);  
}

SCols = pzSqlda->SQLN;  
pzSqlvarn = (struct SQLVARN*)( ((char*)pzSqlda)+SQLDSIZ);

PstrMsg = pzAduexitp->XPUSERMA_;  
PstrMsg += 4;  
SMsgs = 0;  
{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_) = sMsgs;  
*(((short*)pzAduexitp->XPUSERMA_)+1) = sizeof(pzAduexitp->XPUSRMSG);

  return 0;  
}

/*---------------------------*/
*   Exit termination processing   *
/*---------------------------*/

static int Cleanup(struct ADUEXITP *pzAduexitp){
```

UNLOAD PLUS for DB2 Reference Manual
```c
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++; 

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);
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;
```
/*-------------------------------------*
*   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;
}

/*
 *****************************************
*        D I S C L A I M E 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

 *****************************************
*        E X I T S C E N A R I O        *
 *****************************************

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.
Sample C user exit

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. Getmain 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

*****************
*     NOTES     *
*****************

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

This section provides 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

Figure 94 illustrates the LE C exit parameter structure. The ADUXEPITP structure is contained in a header file, aduxith0.h.

Figure 94 LE C user exit parameter structure (part 1 of 3)

```c
struct ADUXEPITP
{
    /**************************************************************************
    /* ADUXEPITP 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) 4 BYTES*/
    void *xpuserA_;      /* A(USERID) 8 BYTES*/
    void *xputidA_;      /* A(UTILITY ID) 16 BYTES*/
    int _f1Y6;           /* */
    /* */
    /* 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*/
    /* */
    void *xpuweraA_;     /* USER WORK AREA ADDRESS */
    int xpuserf1;        /* USER FIELD */
};
```
LE C exit parameter structure

Figure 94  LE C user exit parameter structure (part 2 of 3)

```c
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          /* */

#define XPFLAGS                         xpflags
#define XPFUNC                          xpfunc
#define XPREFP_                         xprefP_
#define XPRESRV                         xpresrv
#define XPSQLDA_                        xpsqldaA_
#define XPSSID_                         xpssidA_
#define XPTABLEA_                       xptableA_
#define XPUSERA_                        xpuserA_
#define XPUSERF1                        xpuserf1
#define XPUSERF2                        xpuserf2
#define XPUSERF3                        xpuserf3
#define XPUSERF4                        xpuserf4
#define xpusermz                        xpusermz
#define XPUSERMA_                       xpusermA_
#define XPUSERWA_                       xpuserWA_
#define XPUTIDA_                        xputidA_

/*
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
ADUEXITP 000000 000000 X char  xpflags
XPFLAGS 000000 000001 X char  xpflags
XPFUNC 000000 000004 F int  xpfunc
XPREFP# 000000 000002 X short xprefP_
XPRESRV 000000 000002 XL2 XL2 xpresrv
XPSQLDA@ 000000 000004 A void * xpsqldaA_
XPSSID@ 000000 000004 A void * xpssidA_
XPTABLE@ 000000 000004 A void * xptableA_
XPUSER@ 000000 000004 A void * xpuserA_
XPUSERF1 000000 000004 F int  xpuserf1
XPUSERF2 000000 000004 F int  xpuserf2
XPUSERF3 000000 000004 F int  xpuserf3
XPUSERF4 000000 000004 F int  xpuserf4
XPUSERMZ 000000 000004 F int  xpusermz
XPUSERM@ 000000 000004 A void * xpusermA_
```
Table 113 describes the major parameter structure elements and their uses.

**Table 113  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 Figure 95 on page 584</td>
</tr>
<tr>
<td></td>
<td>For each table unloaded, this record describes the column names, data types,</td>
</tr>
<tr>
<td></td>
<td>and pointers to the column data within the output record being unloaded.</td>
</tr>
<tr>
<td></td>
<td>This record is identical in format to a DB2 SQLDA, and is used in much the</td>
</tr>
<tr>
<td></td>
<td>same way.</td>
</tr>
<tr>
<td>*xptableA_</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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>an SQLDA.                      The XPREF# is incremented to provide a unique</td>
</tr>
<tr>
<td></td>
<td>reference number.</td>
</tr>
<tr>
<td>*xsidA_</td>
<td>pointer to a 4-byte field containing the SSID value</td>
</tr>
<tr>
<td>*xuserA_</td>
<td>pointer to an 8-byte field containing the user ID of the user executing the</td>
</tr>
<tr>
<td></td>
<td>UNLOAD PLUS utility</td>
</tr>
<tr>
<td>*xputidA_</td>
<td>pointer to the 16-byte utility ID</td>
</tr>
<tr>
<td>CL100 xpusrmsg</td>
<td>a 100-byte message field to use when issuing a simple message</td>
</tr>
<tr>
<td></td>
<td>On return from any function call, if this field is nonblank, UNLOAD PLUS</td>
</tr>
<tr>
<td></td>
<td>issues message 51635I identifying the exit name, exit reference number, and</td>
</tr>
<tr>
<td></td>
<td>this message text. This field is blank after processing.</td>
</tr>
<tr>
<td>*xusermA_</td>
<td>pointer to a user-defined message buffer that can contain multiple messages</td>
</tr>
<tr>
<td></td>
<td>of equal length</td>
</tr>
<tr>
<td></td>
<td>Any messages pointed to by this address are printed after the XUSRMSGS</td>
</tr>
<tr>
<td>xpuserm$</td>
<td>must be set to the size of the user message buffer pointed to by *xusermA_</td>
</tr>
</tbody>
</table>
Pseudo-SQLDA structure

Figure 95 illustrates the UNLOAD PLUS LE C exit pseudo-SQLDA.

Figure 95  LE C pseudo-SQLDA record

```
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
#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. Table 108 on page 541 summarizes the field names, data types, and usage notes for each field in the SQLDA portion of the structure.

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. Table 109 on page 541 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 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. Table 110 on page 542 describes the valid SQLTYPE field values for each data type supported.
See Figure 96 for an illustration of the LE C parameter control block relationships.

**Figure 96  Major LE C user exit pointer relationships**

![Diagram showing relationships between ADUEXITP, SQLDA, and other variables such as Creator, Table Name, SSID, User ID, Utility ID, and Message Size.]

---

**Sample LE C user exit**

Figure 97 is a sample user exit written in LE C.

**Figure 97  Sample LE C user exit (part 1 of 16)**

```c
/*********************************************************************
*     IDENTIFICATION AND DESCRIPTION                              *
*   *************************************************************************
* PGM:         ADUEXIL                                              *
*   *************************************************************************
* xpsqlidaA_  *xptableA_  *xpssidA_  *xputidA_  *xpusermA_  xpusermsg  *
* xpsqldaA_  *xptableA_  *xpssidA_  *xputidA_  *xpusermA_  xpusermsg  *
* xpsqldaA_  *xptableA_  *xpssidA_  *xputidA_  *xpusermA_  xpusermsg  *
* CL100      xpuserm$                                             *
* Creator    Table Name                                         *
* SSID       128                                                                                      *
* User ID    8                                                                                          *
* Utility ID 16                                                                                         *
* #Msgs      Message Size                                      *
* = Size of User Message Area                                                                            *
* User Message 100 bytes                                     *
```
Figure 97  Sample LE C user exit (part 2 of 16)

* 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

/*@*/
#define ADU_MSC
#define __asm
#define _BYREF
#else
#define _BYREF @
#endif

/*-----------------------------------------------------*/
/* #include "aduxith1.h"      aduexitp        */
/*-----------------------------------------------------*/
#define _XL2

typedef struct
{
  #ifdef ADU_MVS
    unsigned BF : 16;
  #else
    unsigned BF1 : 8;
    unsigned BF2 : 8;
  #endif
  #endif
#endif

@if !_defined(_XL2)
#define _XL2
typedef struct
{
  #ifdef ADU_MVS
    unsigned BF : 16;
  #else
    unsigned BF1 : 8;
    unsigned BF2 : 8;
  #endif
  #endif
#else

#pragma linkage(ADUXITR1,OS)

/*@*/
#define "aduxith1.h" aduxitp
/*@*/
# Figure 97  Sample LE C user exit (part 3 of 16)

\)
XL2;
#endif

if !defined(_CL100)
define _CL100
typedef char CL100Ý100¨;
#endif

struct ADUEXITP
{

int xpfive; /* 0 = PROCESS, 1 = INIT, 2=TERMINATE */
void *xpsqlda_; /* 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) 4 BYTES*/
void *xputidA_; /* A(USERID) 8 BYTES*/
/* */
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 */

Figure 97  Sample LE C user exit (part 3 of 16)
/*
   XL2 xpresrv; /* RESERVED V12*/
/*
   char xpusrpad[842]; /* PADDING */
#define XPK 1024 /* */
}

#define XPFLAGS
#define XPFUNC
#define XPREFP_
#define XPRESRV
#define XPSQLDA_
#define XPSSID_
#define XPTABLEA_
#define XPUSERA_
#define XPUSERF1
#define XPUSERF2
#define XPUSERF3
#define XPUSERF4
#define xpusermz
#define XPUSERMA_
#define XPUSERWA_
#define XPUSRMSG
#define XPUSRPAD
#define XPUTIDA_

/
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
ADUEXITP 000000 000000 X char xpflags
XPFLAGS 00000C 000001 F int xfunc
XPFUNC 00000E 000002 H short xprefP_
XPRESRV 0000B4 000002 XL2 XL2 xpresrv
XPSQLDA@ 000004 000004 A void * xpsqldaA_
XPSSID@ 000010 000004 A void * xpsidA_
XPTABLE@ 000014 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 00004C 000004 F int xpusermz
XPUSERM@ 000048 000004 A void * xpusermA_
XPUSERW@ 000054 000004 A void * xpuserwA_
XPUSRMSG 000050 000004 CL100 CL100 xpusrmsg
XPUSR PAD 000086 000034A (1024-... char [842] xpusrpad
XPUTIDA@ 000018 000004 A void * xputidA_
*/
Figure 97  Sample LE C user exit (part 5 of 16)

/*-----------------------------------------*/
/* #include "aduxith2.h" lvarea */
/*-----------------------------------------*/

#if !defined(_XL292)
define _XL292
typedef char XL292Ỹ292¨;
#endif

#if !defined(_CL128)
define _CL128
typedef char CL128Ỹ18¨;
#endif

#if !defined(_CL8)
define _CL8
typedef char CL8Ỹ8¨;
#endif

#if !defined(_CL6)
define _CL6
typedef char CL6Ỹ6¨;
#endif

#if !defined(_PL2)
define _PL2
typedef struct
{  
  #ifdef ADU_MVS
  unsigned BF :16;
  #else
  unsigned BF1 :8;
  unsigned BF2 :8;
  #endif
} PL2;
#endif

struct LVAREA
{
  /********************************************************************
  /* LOCAL VARIABLES WORK AREA DSECT */
  /********************************************************************
  int lvsaveỸ18¨;        /* 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 */
  */
/*
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 CL30Ý30¨;
#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.sqlname
#define sqlnamz            _s1.sqlnamz
#define SQLPRCSN           sqlprcsn
#define SQLSCALE           sqlscale
```c
#define SQLTYPE sqltype

/*
SYMBOL OFFSET SIZE TYPE C-TYPE C-NAME
SQLDATA 000004 000004 A void * sqldata
SQLIND  000008 000004 A void * sqlind
SOLLLEN 000002 000000 0H short [0] slllen
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
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:",
    szIgot2cln [] = "PROCESSING IS COMPLETED"
    :

int main(int argc, char *argv[]) {

    int lRc;
    long lFcode;
    long *rc_ptr;
    long *hold_ptr;
    struct ADUEXITP *pzAduexitp;

    lFcode = *(long *)argv[1];
    hold_ptr = (long *)(*(long *)argv[2]);
    pzAduexitp = (void *)hold_ptr;
    rc_ptr = (long *)argv[3];
    *rc_ptr = 0;

    /*---------------------*
     *   Main Entry Point   *
     */
```

Figure 97  Sample LE C user exit (part 8 of 16)
lRc = 0;

switch(lFcode){
    case PROCESS:
        lRc = Process(pzAduexitp);
        break;
    case INITIAL:
        lRc = Initial(pzAduexitp);
        break;
    case CLEANUP:
        lRc = Cleanup(pzAduexitp);
        break;
    default:
        lRc = BadFuncCode(pzAduexitp);
    }

*rc_ptr = lRc;
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 *pzSqlDa;
struct SQLVARN *pzSqlVarn;

if ( (pstrMsg=pzAduexitp->XPUSERMA_)!=NULL ){
  *(short*)pstrMsg = 0;
}

/* Say I got here */
memcpy(pzAduexitp->XPUSRMSG ,szIgot2pro ,sizeof(szIgot2pro));

pzSqlDa = pzAduexitp->XPSQLDAA_;
sCols = pzSqlDa->SQLN;
pzSqlVarn = (struct SQLVARN*)( ((char*)pzSqlDa)+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 = pzSqlDa->SQLN;
pzSqlVarn = (struct SQLVARN*)( ((char*)pzSqlDa)+SQLDSIZ);
      while(sCols){
        if ( memcmp(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; /* QUIT ! */
}
}
pzSqlvarn++;
sCols--;
memcpy(pzAduexitp->XPUSRMSG ,"REGISTRATION COLUMN NOT FOUND" ,29);
return 8; /* QUIT ! */
}

/*------------------------*
* 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);  

pzSqllda = pzAduexitp->XPASSWORDA_;  
sMsgs = pzSqllda->SQLN * sizeof(pzAduexitp->XPUSRMSG);  
sMsgs += 4 + 256; /* adjust for prefix and pad */
if ( (pzAduexitp->XPUSERMA_ = calloc(1, sMsgs)) == 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_) = 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 = pzAduxitp->XPUSERWA_; 

memcpy(pzAduxitp->XPUSRMSG ,szIgot2cln ,sizeof(szIgot2cln));
sMsgs = 0;
pstrMsg = pzAduxitp->XPUSERMA_; 
pstrMsg += 4; /* bump past gunk */
aduxitr1(3,pzAduxitp->XPUSERWA_); /* unpack counts */

memset(pstrMsg ,'' ,sizeof(pzAduxitp->XPUSRMSG));
memcpy(pstrMsg ,pzLvarea->LVDISC ,sizeof(pzLvarea->LVDISC));
memcpy(pstrMsg+sizeof(pzLvarea->LVDISC)+1 ,"DID NOT QUALIFY AS HEAVY CRUISERS",33);
pstrMsg += sizeof(pzAduxitp->XPUSRMSG);

sMsgs++;

memset(pstrMsg ,'' ,sizeof(pzAduxitp->XPUSRMSG));
memcpy(pstrMsg ,pzLvarea->LVKEEP ,sizeof(pzLvarea->LVKEEP));
memcpy(pstrMsg+sizeof(pzLvarea->LVKEEP)+1 ,"QUALIFIED AS HEAVY CRUISERS",27);
pstrMsg += sizeof(pzAduxitp->XPUSRMSG);

sMsgs++;

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;
Figure 97  Sample LE C user exit (part 14 of 16)

/*-------------------------------------*
*   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;
}

/*
***************************
*   D I S C L A I M E R   *
***************************

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

*******************************
*   E X I T S C E N A R I O   *
*******************************

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, star fleet 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

***************
*      NOTES    *
***************

ADUXIL 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 ADUXEXITP 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 library and are shown here. The first one (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

Figure 98 illustrates the UNLOAD PLUS COBOL exit parameter record. (The exit parameter record is the same for both sample exits.)

**Figure 98** COBOL user exit parameter record

<table>
<thead>
<tr>
<th>01</th>
<th>EXITPARMS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>EXIT-FUNCTION-CODE  PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>88</td>
<td>PROCESS-CALL  VALUE 0.</td>
</tr>
<tr>
<td>88</td>
<td>INITIALIZE-CALL  VALUE 1.</td>
</tr>
<tr>
<td>88</td>
<td>CLEANUP-CALL  VALUE 2.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-SQlda-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-TB-NAME-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER  PIC XX.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-REFERENCE-NUM  PIC S9(4) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-SSID-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERID-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-UTILID-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER  PIC X(24).</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USER-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF1  PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF2  PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF3  PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF4  PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERMSG-POINTER  POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERMSG-SIZE  PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USER-MESSAGE  PIC X(100).</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-FLAGS  PIC X.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER  PIC X.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER  PIC X(214).</td>
</tr>
<tr>
<td>02</td>
<td>FILLER  PIC X(100).</td>
</tr>
</tbody>
</table>
Table 114 describes the major fields and their uses.

### Table 114  COBOL user exit parameter fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| EXIT-SQLDA-POINTER | pointer to the pseudo-SQLDA for this table as shown in Figure 99 on page 603 and Figure 100 on page 603  
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. |
| EXIT-TB-NAME-POINTER | 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. |
| EXIT-REFERENCE-NUM | exit reference number  
Each table being unloaded gets its own copy of a user parameter block and an SQLDA. The XREF# is incremented to provide a unique reference number. |
| EXIT-SSID-POINTER | pointer to a 4-byte field containing the SSID value |
| EXIT-USERID-POINTER | pointer to an 8-byte field containing the user ID of the user executing the UNLOAD PLUS utility |
| EXIT-UTILID-POINTER | pointer to the 16-byte utility ID |
| EXIT-USER-MESSAGE | 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. |
| EXIT-USERMSG POINTER | 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. |
| EXIT-USERMSG-SIZE | must be set to the size of the user message buffer pointed to by XPUSERM@ |
Figure 99 and Figure 100 illustrate the exit pseudo-SQLDA DSECTs for the two samples.

Figure 99   COBOL pseudo-SQLDA record: sample 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLDA</td>
<td></td>
</tr>
<tr>
<td>SQLDAX</td>
<td>PIC X(8).</td>
</tr>
<tr>
<td>SQLDABC</td>
<td>PIC S9(8) COMPUTATIONAL.</td>
</tr>
<tr>
<td>SQLN</td>
<td>PIC S9(4) COMPUTATIONAL.</td>
</tr>
<tr>
<td>SQLD</td>
<td>PIC S9(4) COMPUTATIONAL.</td>
</tr>
<tr>
<td>SQLVAR</td>
<td>OCCURS 1 TO 300 TIMES DEPENDING ON SQLN</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>PIC S9(4) COMP.</td>
</tr>
<tr>
<td>SQLLEN</td>
<td>PIC S9(4) COMP.</td>
</tr>
<tr>
<td>SQLDATA</td>
<td>POINTER.</td>
</tr>
<tr>
<td>SQLIND</td>
<td>POINTER.</td>
</tr>
<tr>
<td>SQLNAME</td>
<td></td>
</tr>
<tr>
<td>SQLNAMEL</td>
<td>PIC S9(4) COMP.</td>
</tr>
<tr>
<td>SQLNAMEC</td>
<td>PIC X(30).</td>
</tr>
</tbody>
</table>

Figure 100   COBOL pseudo-SQLDA record: sample 2

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLDA</td>
<td></td>
</tr>
<tr>
<td>SQLDAX</td>
<td>PIC X(8).</td>
</tr>
<tr>
<td>SQLDABC</td>
<td>PIC S9(8) COMPUTATIONAL.</td>
</tr>
<tr>
<td>SQLN</td>
<td>PIC S9(4) COMPUTATIONAL.</td>
</tr>
<tr>
<td>SQLD</td>
<td>PIC S9(4) COMPUTATIONAL.</td>
</tr>
<tr>
<td>SQLVAR</td>
<td>OCCURS 1 TO 300 TIMES DEPENDING ON SQLN INDEXED BY I.</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>PIC S9(4) COMP.</td>
</tr>
<tr>
<td>SQLLEN</td>
<td>PIC S9(4) COMP.</td>
</tr>
<tr>
<td>SQLDATA</td>
<td>POINTER.</td>
</tr>
<tr>
<td>SQLIND</td>
<td>POINTER.</td>
</tr>
<tr>
<td>SQLINDN</td>
<td>REDEFINES SQLIND PIC S9(9) COMP.</td>
</tr>
<tr>
<td>SQLNAME</td>
<td></td>
</tr>
<tr>
<td>SQLNAMEL</td>
<td>PIC S9(4) COMP.</td>
</tr>
<tr>
<td>SQLNAMEC</td>
<td>PIC X(30).</td>
</tr>
</tbody>
</table>

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. Table 108 on page 541 summarizes the field names, data types, and usage notes for each field in the SQLDA portion of the DSECT.

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. Table 109 on page 541 summarizes the field names, data types, and usage notes for each field in this portion of the DSECT.
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. Table 110 on page 542 describes the valid SQLTYPE field values for each data type supported.

See Figure 101 for an illustration of the COBOL parameter control block relationships.

**Figure 101  Major COBOL user exit pointer relationships**
Figure 102 is a sample user exit written in COBOL.

**Figure 102  Sample COBOL user exit: sample 1 (part 1 of 4)**

```cobol
*------------------------
IDENTIFICATION DIVISION.
*------------------------
*
* THIS IS A SAMPLE OF A COBOL EXIT FOR UNLOAD+ FOR DB2.
*
PROGRAM-ID.    ADUXITC.
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.
```
**Figure 102  Sample COBOL user exit: sample 1 (part 2 of 4)**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>FILLER PIC X(16) VALUE &quot;CLEANUP DONE. &quot;.</td>
</tr>
<tr>
<td>10</td>
<td>FILLER PIC X(21) VALUE &quot;PROCESSED ROW COUNT: &quot;.</td>
</tr>
<tr>
<td>10</td>
<td>TERM-ROW-COUNT PIC ZZ,ZZZ,ZZ9.</td>
</tr>
<tr>
<td>01</td>
<td>WORK-STUFF.</td>
</tr>
<tr>
<td>10</td>
<td>ROW-COUNT PIC S9(9) VALUE 0.</td>
</tr>
</tbody>
</table>

**LINKAGE SECTION.**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>EXITPARMS.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-FUNCTION-CODE PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>88</td>
<td>PROCESS-CALL VALUE 0.</td>
</tr>
<tr>
<td>88</td>
<td>INITIALIZE-CALL VALUE 1.</td>
</tr>
<tr>
<td>88</td>
<td>CLEANUP-CALL VALUE 2.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-SQLDA-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-TB-NAME-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER PIC XX.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-REFERENCE-NUM PIC S9(4) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-SSID-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERID-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-UTILID-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER PIC X(24).</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USER-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF1 PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF2 PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF3 PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERF4 PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERSMSG-POINTER POINTER.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USERSMSG-SIZE PIC S9(9) BINARY.</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-USER-MESSAGE PIC X(100).</td>
</tr>
<tr>
<td>02</td>
<td>EXIT-FLAGS PIC X.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER PIC X.</td>
</tr>
<tr>
<td>02</td>
<td>FILLER PIC X(214).</td>
</tr>
<tr>
<td>02</td>
<td>FILLER PIC X(100).</td>
</tr>
<tr>
<td>02</td>
<td>ROW-COUNT PIC S9(9) VALUE 0.</td>
</tr>
<tr>
<td>01</td>
<td>TABLE-INFO.</td>
</tr>
<tr>
<td>13</td>
<td>TABLE-CREATOR PIC X(128).</td>
</tr>
<tr>
<td>13</td>
<td>TABLE-NAME PIC X(128).</td>
</tr>
<tr>
<td>01</td>
<td>SSID PIC X(04).</td>
</tr>
<tr>
<td>01</td>
<td>USERID PIC X(08).</td>
</tr>
<tr>
<td>01</td>
<td>UTILITY-ID PIC X(16).</td>
</tr>
<tr>
<td>01</td>
<td>SQLDA.</td>
</tr>
<tr>
<td>02</td>
<td>SQLDAX PIC X(8).</td>
</tr>
<tr>
<td>02</td>
<td>SQLDABC PIC S9(8) COMPUTATIONAL.</td>
</tr>
<tr>
<td>02</td>
<td>SQLN PIC S9(4) COMPUTATIONAL.</td>
</tr>
<tr>
<td>02</td>
<td>SOLN PIC S9(4) COMPUTATIONAL.</td>
</tr>
<tr>
<td>02</td>
<td>SOLVAR OCCURS 1 TO 300 TIMES DEPENDING ON SQLN.</td>
</tr>
<tr>
<td>03</td>
<td>SQLENC PIC S9(4) COMP.</td>
</tr>
<tr>
<td>03</td>
<td>SQLMENC PIC S9(4) COMP.</td>
</tr>
</tbody>
</table>
Figure 102  Sample COBOL user exit: sample 1 (part 3 of 4)

    03  SQLDATA      POINTER.
    03  SQLIND      POINTER.
    03  SQLNAME.
       04  SQLNAMEL  PIC S9(4) COMP.
       04  SQLNAMEC  PIC X(30).

    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.
   *---------------------

      MOVE ZERO TO RETURN-CODE.
      SET ADDRESS OF SQLDA TO EXIT-SQLDA-POINTER.
     MOVE "PROCESS" TO EXIT-USER-MESSAGE.
      ADD 1 TO ROW-COUNT.

      PROCESS-RECORD-EXIT.
         EXIT.

   *---------------------
   INITIALIZE-THE-EXIT.
   *---------------------

      MOVE ZERO TO RETURN-CODE.
      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-UTILITY-ID-POINTER.
COBOL II and LE COBOL user exits: sample 2

Figure 102  Sample COBOL user exit: sample 1 (part 4 of 4)

```
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.
```

Figure 103 is another sample user exit written in COBOL.

COBOL II and LE COBOL user exits: sample 2

Figure 103  Sample COBOL user exit: sample 2 (part 1 of 7)

```
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.
INITIALIZE-THE-EXIT-EXIT.
EXIT.

*---------------------
CLEANUP-THE-EXIT.
*---------------------

MOVE ZERO TO RETURN-CODE.
MOVE ROW-COUNT TO TERM-ROW-COUNT.
MOVE TERM-MSG TO EXIT-USER-MESSAGE.
CLEANUP-THE-EXIT-EXIT.
EXIT.
```
* PROGRAM-ID.   ADUEXTC2.
  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='.
Figure 103  Sample COBOL user exit: sample 2 (part 3 of 7)

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 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).
  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 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-SOLDA-POINTER  POINTER.
  02 EXIT-TB-NAME-POINTER POINTER.
  02 FILLER               PIC XX.
**Figure 103 Sample COBOL user exit: sample 2 (part 4 of 7)**

```
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-USERF1          PIC S9(9) BINARY.
02 EXIT-USERF2          PIC S9(9) BINARY.
02 EXIT-USERF3          PIC S9(9) BINARY.
02 EXIT-USERF4          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.
```
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-POINTERS.
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.
Index

Symbols

$ADUOPTS macro 438
$ADUOUTP macro 438
$AUPSMAC macro 438
$C30DOPPT customization job 438
$C32SOPT customization job 438
$NO_UNIT$ keyword 192, 467
&ATTACH variable 194, 468
&DATE variable 194, 468
&DAY variable 194, 468
&DB variable
description 194, 468
resolving when using DIRECT NO 196
&HOUR variable 194, 468
&JDATE variable 194, 468
&JDAY variable 194, 468
&JOBNAME variable 194, 468
&MIN variable 194, 468
&MINUTE variable 194, 468
&MOUNT variable 194, 468
&OBID variable
description 194, 468
resolving when using DIRECT NO 196
&PART variable
description 195, 469
examples 300, 387
resolving when using DIRECT NO 196
&POSNUM variable 195, 469
&SEC variable 195, 469
&SECOND variable 195, 469
&SELNUM variable
description 195, 469
example 299
&SSID variable 114, 195, 469
&STEPNAME variable 195, 469
&TIME variable 195, 469
&TS variable
description 195, 469
resolving when using DIRECT NO 196
&TYPE variable 195, 469
&USERID variable 195, 469
&UTIL variable 195, 470
&YEAR variable 195, 470

A

abends
JES3 47, 307
recovery from 178
using STOP UTILITY for 180
using TERMINATE UTILITY for 178, 181
above-the-bar storage 53
access, shared 492
accessibility of objects 36, 110
ACF2. See CA-ACF2
ACFORTSS option 26
ACHKP (auxiliary CHECK pending) status 57
ACS (Automatic Class Selection) routines 304
activating dynamic allocation 130, 460
ACTIVE option 130
ADU$OPTS options module 437
ADUUMAIN module 272
algorithms for calculating space for data sets 305
aliases, specifying in FROM option 223
ALL keyword
of FORCE option 187, 449
of ON FAILURE option 178
allocating data sets
copy 279
DDLIN 278
dynamically 293
See also dynamic allocation
estimating size with ANALYZE option 131, 441
output 286
sort (SORTWK) 281
SORTWK 281
SYSCNTL 284
SYSIN 285
SYSPRIN2 285
SYSPRINT 285
SYSREC 286
SYSRED 286
using DD statements 276
VSAM 280
allocating directory, for referenced file data sets 199
ANALYZE option
command 131
installation 441
using to calculate data set size 301
AND option of FORMAT CSV option 80
APF authorization requirements 276, 437, 538
application defaults module 273
application-period temporal tables
  CNTLCARDS BMICLOAD 141
  CNTLCARDS DB2 139
  CNTLCARDS DB2DDL 142
  CNTLCARDS DB2LOAD 138
ASC keyword of ORDER BY option 232
ASCII
data translation 266
exit routines 84
option 165
ASM keyword of UNLOADEXIT option 176
assembler user exit routine
  example 544
  specifying 176
&ATTACH variable 194, 468
authorization mechanisms 51
authorizations
  APF authorization 276, 538
  DISPLAY 52
  for canceling threads 52
  for data sets 52
  mechanisms, description 51
  OPNDB2ID option 454
  RACF 52
  READ privileges 52
  required to run UNLOAD PLUS 51
  system 52
AUTO keyword
  of DIRECT option 108
  of FORMAT XML option 160
  of RECFM option 136, 455
AUTOTAG option 152
auxiliary CHECK pending (ACHKP) status 57
availability, data 36, 110

B
backing up BMC tables 484
BASIC keyword of DSNTYPE option 198, 471
batch jobs, building and executing 271
batch pipe data 71
batch threads, number used by UNLOAD PLUS 54
benefits of UNLOAD PLUS 32
BIGINT data type 246
binary data 248, 251
BINARY data type 248
binary files, specifying 241, 243
BINARY VARYING data type 248
binary XML data 241, 243
BINARYXML keyword
  of CHAR data type 241
  of VARCHAR data type 243
bitemporal tables
  CNTLCARDS BMICLOAD 141
  CNTLCARDS DB2 139
  CNTLCARDS DB2DDL 142
  CNTLCARDS DB2LOAD 138
  FORMAT BMICLOAD 74
BLOB data type 251
BLOBF keyword of character data types 241, 243
block size, of copy data sets 424, 451
BMC internal format, output in 157
BMC Software, contacting 2
BMC utilities
  displaying status 486
  running concurrently 55
  terminating 486
BMC utility tables 483
BMCDICT table
  considerations 487
  contents 487
  maintaining 488
BMCFORCE data set 277
BMCHIST table
  backing up 484
  contents 488
  maintenance 490
  querying 486
  updating using HISTORY 451
BMCLGRNX table 490
BMCMLOAD keyword
  of CNTLCARDS option 140, 294, 407
  of FORMAT option 157
BMCSORT
  controlling 425
  description 48
  ENUMROWS to pass number of rows to 148
  installation option 480
  memory usage 425, 457
  performance tuning 425
  SMCORE option 457
  sort work data sets, allocating 281
  SORTDEVT option 149
  SORTNUM option 149, 458
  version requirement 50
BMCSYNC table
  backing up 484
  considerations 493
  contents 491
  LOB data considerations 494
  maintaining 494
  purpose 55
  running utilities concurrently 55
  updating 42, 452
  XML data considerations 493
BMCTTRANS table 495
BMCULOAD option (in LOADPLUS) 407
BMCUTIL table
   backing up 484
   contents 496
   ending an UNLOAD PLUS job 308
   maintaining 498
   updating 42, 452
BMCXCOPY table
   backing up 484
   cabinet copies 120
   contents 499
   Instant Snapshot copies 125
   maintaining 503
   online consistent copies 125
   querying 486
buffer usage
   controlling 423
   IBUFFS option 424, 451
   specifying for INFILE option data sets 424
   UBUFFS option 424, 459
BYTASK keyword of ORDER YES option 147

C
C language
   sample user exit 560
   UNLOADEXIT option 176
$C30DOPT customization job 437
$C32SOPT customization job 437
CA-ACF2 security product 51
cabinet copies, unloading from 118
cache, sufficient for SUF 62
caching, DASD 430
calculating
   data set size during dynamic allocation 301
   estimated number of rows for a table space 133
   nonpartitioned sort work space 283
   SORTWK data sets size 282
   space for SMS-managed data sets 303
canceling a job 308
canceling threads
   authorizations required 52
   FORCE option 187, 449
   FORCE_AT option 449
   FORCE_RPT option 450
   reporting 188, 277, 450
   specifying when to cancel 187, 449
carriage return characters, in FORMAT CSV options 80
catalog of DB2, unloading from 70
CA-Top Secret security product 51
CCSID option
   description 166
   performance considerations 422
   relation to FORMAT BMCLOAD option 75
CCSIDs (code character set identifiers) 165, 166
CENTURY option
   command 183
   installation 442
   of DATE data type 249
   of TIMESTAMP data type 250
CHANGE_CONSISTENT option 442
CHANGE_QUIESCE option 443
changes to product 22
CHAR data type 240
character constants
   specifying with CONSTRULES 444
   string rules 228
   translation 268
character data
   collating sequence 146, 231
   conversions (translation) supported 267
   data types 240, 242
   translation 266
   characters, multiple-byte 80
   characters, special 80
classes, SMS 203, 304, 473
cleaning up unused data sets 181
CLOB data type 251
CLOBF keyword of character data types 241, 243, 379
CLONE option 105
close tables 29, 105, 223
clustering key, ordering by 146, 320, 427
CMAX option 155, 443
CMRATIO option
   calculating HURBA with formula 132
   installation 444
   with ANALYZE option 133
CNTLCARDS options
   BMCLOAD keyword 140, 294, 513
   DB2 keyword
      description 138, 509
      examples 325, 346, 370
   DB2DDL keyword 141, 514
   DB2LOAD keyword 137, 379, 508
   description 137
   EASYTRIEVE keyword 142, 525
   examples
      DB2 325, 346, 370
      DB2LOAD 379
      default 397
      SAS 334
   FOCUS keyword 142, 515
   inserting additional command options using the
      string parameter 138, 506
   NOMAD keyword 143, 535
   performance considerations 422
   SAS keyword 143, 334, 518
   specifying SYSCTNL data set 284
   SQL/DS keyword 143, 522
   SQL/DS-DDL keyword 143, 524
CNTLCARDS options (continued)
   SQL/DS-LOAD keyword 144, 520
   TERADATA keyword 144, 534
   TERADATA-BULK keyword 144, 528
   TERADATA-FAST keyword 145, 531
   TERADATA-MULT keyword 145, 533

CNTLDDN option
   description 145
   examples 346
   specifying SYSCNTL data set 284

COBOL II
   sample user exit 601
   UNLOADEXIT COBOL2 option 176

codes
   function 540
   return 177, 179, 540
   SQL 458

column-level security 41, 218

columns
   comparing value to a constant 226
   definitions when using FORMAT BMLOAD 75
   identity 77
   IMPLICITLY HIDDEN 218
   timestamp, with precision 24, 250
   timestamp, with time zone 24, 170
   unloading selected 320

columns
   comma as decimal point 228
   comma as separator 228
   comma-delimited data. See CSV keyword of FORMAT option and CSV output

command options
   descriptions 103
   diagrams 92
   examples 309
   performance 421
   rules for syntax 87

column-separator-value data. See CSV keyword of FORMAT option and CSV output

comparison operators 225, 226

components that UNLOAD PLUS uses 48

conversion
   BMCDICT table 487
   defining for tape data sets 214
   dictionary, objects containing 27, 120, 123
   ratio, specifying 444

concatenated copy data sets 124

concurrency
   considerations 55
   SHRLEVEL option 57, 110

concurrent partitions, number of 453

consistent data image, unloading 61

constants
   AUTOTAG option 152
   binary 229
   character string 228
   command 268

compression
   comparison 268
   CONSTRULES option 444
   date/time string 229
   decimal 228
   double-byte character set (DBCS) 218
   floating point 239
   graphic string 229
   hexadecimal 229
   IN keyword 268
   integer 228
   length in string 78
   LIKE keyword 268
   NULL keyword 229
   NULLCHAR option 162
   output-record lagging 37
   predicate block 227, 268
   RECORDID option 222
   rules for 228
   supported types 239
   user-specified 38
   with WHERE clause 227
   CONSTRULES option 444
   CONTINUE UTILITY keyword of ON MESSAGE option 177

control statements
   generating 137, 505
   inserting additional command options into 506
   output data set specification 284
   product type specification 137

conventions
   documentation 20
   rules for specifying command options 87

conversions
   data type, allowable 255
   errors 152
   examples 392
   EXIT data type for user-defined 253
   floating-point and graphic constants 253
   formats supported 254, 255, 263
   order of 269
   role of DISCARDS option 152
   rounding of values 245, 246
   supported types 237
   TRIM and TRUNCATE options 242, 248
   user-defined exit routines 253

copies, cabinet 118

copies, encrypted
   authorization requirements 53
   key data set for 452
   specifying 120

copies, unloading from. See image copies, unloading from copy data sets
   authorization for 52
   description 46
   specifying 279
   unloading from 118
COPY PLUS
  encrypted copies from 120
  Instant Snapshot copies 125
  key data set 452
copy registration
  BMCXCOPY table 499
  cabinet copies 120
  online consistent copies 125
  snapshot copies 125
CSV keyword of FORMAT option
  description 158
  with LRECL option 201
  with USELRECL option 135
CSV output
  considerations 79
  example 84, 366
  specifying 158
  CURRENT DATE option 218, 229, 239
  current date variables 194, 468
  CURRENT RID option 219
  CURRENT TIME option 239
  current time variables 194, 468
  CURRENT TIMESTAMP option 230, 239
  CURRENTDEGREE option 109, 370, 444
  customer support 3

D

D2U (DB2 Utilities Common Code) 48, 50, 276
DASD, group 304
data
  availability 36, 110
  compression, specifying 214, 479
  consistent image 61
  conversion 253, 255
  high-speed migration 73, 407, 422
  migration, using UNLOAD PLUS and LOADPLUS
    73, 407, 422
  sampling 151
  sharing 273
  translation 266, 422
data classes, SMS 203, 473
data conversions
  allowable 255
  numeric considerations 264
  with respect to data translation 269
data sets
  See also individual data set types (for example, SYSREC
data sets)
  allocating 277
  allocating dynamically 130, 460
    See also dynamic allocation
  authorization 53
  BMCFORCE 277
  calculating size with ANALYZE option 301
  cleaning up unused 181
  compression for tape data sets 214
  concatenated 124
  copy
    description 46
    specifying 279
    unloading 118
  DB2, controlling buffer usage 423
  DDLIN 278
  deleting 181
  description 46, 276
  disk 206
  dynamic allocation 130, 460
    See also dynamic allocation
  EAV 29
  expiration date, setting 215
  extended sequential 198, 471
  key, for encrypted copies 452
  large format 198, 471
  LBI 424, 451
  logical record length, specifying 201
  multiple 289, 429
  options for disk and tape 191
  output 286
  overriding default ddnames 126, 128, 129
  overview 46
  record format for output, specifying 136, 455
  referenced files for LOB and XML data
    specifying 287
    specifying directory size 199
    unloading to 63
  retention period, setting 215
  setting maximum number in a data group 202
  sizing using FILESZPCT option 211
  SORTWK 281
  space, adjusting when using automatic sizing 211
  specifying DD statements 277
  SYSCNTL 140, 284
  SYSIN 285
  SYSPRIN2 285
  SYSPRINT 285
  SYSREC 286
  SYSRED 286
  tape 214
  unload, multiple 429
  UTPRINT 307
data translation
  process 266
  specifying options 165, 422
data types
  BIGINT 246
  BIGINT EXTERNAL 246
  BINARY 248
  BINARY VARYING 248
  BLOB 251
  BLOBF 241, 243
  CHAR 240, 263
  CLOB 251

Index  619
data types (continued)
   CLOB 241, 243
   conversions 255, 269
   DATE 249
   DATE EXTERNAL 249
   date/time considerations 265
   date/time formats 254
   DATE-format EXTERNAL 249
   DBCLOB 251
   DBCLOBF 241, 243
   DECFLOAT 172, 252, 264
   DECIMAL 247
   DECIMAL EXTERNAL 247
   DECIMAL PACKED 247
   DECIMAL ZONED 247
   default formats 237
   default lengths 255
   description of each 240
   EXTERNAL 263, 265
   FLOAT 248
   FLOAT EXTERNAL 248
   GRAPHIC 244
   INTEGER 245
   INTEGER EXTERNAL 245
   keywords 240
   numeric considerations 264
   numeric EXTERNAL considerations 263
   option (with SELECT) 237
   &DB variable 194, 468
   &DAY variable 194, 468
   &DATE variable 194, 468
   &DATEFMT option 167
   DATE-format EXTERNAL data type 183, 249
   DATE data type 249
   DATE EXTERNAL data type 249
   DATEFMT option 167
   DATE-format EXTERNAL data type 183, 249
   &DB variable 194, 468
   &DB variable 194, 468
   DB2
   authorization needed to execute UNLOAD PLUS 51
   catalog, unloading from 70
   control statements 137, 508
   CREATE TABLE DDL generation 138, 514
   data sets 423
   features, unsupported 85
   internal format, output in
   FORMAT INTERNAL 73, 157
   FORMAT STANDARD 72, 157
   parallelism 369
   platforms, unloading data to non-DB2 422
   subsystem identifier (SSID) parameter 273
   system setup 50
   user exits 76, 84, 427
   versions supported 50
   DB2 Solution Common Code (SCC) 48
   DB2 Utilities Common Code (D2U) 48, 50, 276
   DB2STATS keyword of ANALYZE option 131
   DBCLOB data type 251
   DBCLOBF keyword of ANALYZE option 131
   DBCS (double-byte character set), support for 218
   DD statements
   BMCFORCE 277
   DSN1COPY 279
   image copy 279
   SORTWK 281
   subsystem identifier (SSID) parameter 273
   system setup 50
   UTPRINT 307
   SYSREC 286
   SYSRED 286
   VSAMDD 280
   DDL, using for DB2 object definitions with INFILE
   ddname 278
   DDLDDN option 128
   DDLIN
   data set 46, 278
   data sets for DIRECT NO 47
   data sets for DIRECT YES 46
   data set 46, 278
   data sets for DIRECT NO 47
   data sets for DIRECT YES 46
ddnames (continued)
  descriptions of data sets 276
  specifying for the control statement data set 145
  specifying for the DDL input data set 128
  specifying for the output data set 129, 460
  with INFILE option 118, 122
DECFLOAT data type 172, 252, 264
DECIMAL data type 247
DECIMAL data type 247
  decimal data, zoned 171, 247
  decimal floating-point data type 172
DECIMAL data type 247
DECIMAL data type 247
default options. See installation options
DEFINE NO, objects created with 27
DELETEFILES option 181
deleting work files
  examples 374, 408
  specifying 181, 445
DELFILES option 445
delimited data. See CSV option
delimited tokens 87
delimiting characters
  for CSV format 158
  for FORMAT CSV 158
  for FORMAT XML 161
DESC keyword of ORDER BY option 232
dictionaries, compression
  BMCDICT table 487
  objects containing 27, 120, 123
DIR option 199
DIRECT option
  AUTO 108, 147
  description 39, 106
  NO
  data sets 47
  description 107
  examples 370, 375, 378
  number of DB2 threads 54
  processing phases 44
  relation to ANALYZE processing 131
  resolving symbolic variables 196
  serialization and concurrency issues 59
  YES
  data sets 46
  description 106
  example 378
  number of DB2 threads 54
  processing phases 42
  relation to AUTO option 108
  serialization and concurrency issues 57
  view restrictions 223
DISABLED keyword of ZIIP option 114, 463
disabling zIIP processing 114, 463
discarded records 152
DISCARDS option 152
disk data sets
  command options for dynamic allocation 206
  expiration date, specifying 213, 478
  installation defaults for dynamic allocation 474
  retention period, specifying 213, 477
disk space, specifying 474, 476
disk volumes, specifying 477
DISKEXPD option 213, 478
DISKRETN option 213, 477
DISPLAY privileges 52
displaying status of BMC utilities 486
disposition
  and deleting files 182, 408, 445
  SYSREC data sets 288
tape files 459
DOPTS. See installation options
double-byte character set. See DBCS
drain
  canceling threads for 187, 449
displaying locks 186, 447
  number of retry attempts 185, 446
  processing description 59
  read claimers, canceling 187, 449
time between retry attempts 185, 446
timeout value 184, 446
  write claimers, canceling 187, 449
DRAIN_WAIT option 184
DRNDELAY option 446
DRNRETRY option 446
DRNWAIT option 446
DSECT, assembler exit parameter block 544
DSN1COPY
  buffer specification for reading 424
  compression dictionary limitation 123
data sets 37
  source of unload 118
table space status 57
DSNAME option
  command 193
  example 378
  installation 467
  substitution variables in 295
DSNDB04 database name 105
DSNHDECP
  copy loaded from STEPLIB 452
date and time format 261
default values at installation 165
  printing values 275
settings for changing decimal points to comma 228
SSID from 273
time zone value 170
DSNTIAUL format, output in 78, 158
DSNTIAUL keyword of FORMAT option
  description 78, 158
  example of unloaded data 84
DSNTIAUL keyword of FORMAT option (continued)
relation to FIXEDVARCHAR option 174
relation to NULLCHAR option 162
DSNTYPE option 197, 378, 470
DSNZPARMS, utility timeout value 185
DSPLOCKS option 186, 447
dump, system 448, 456
DYNALOC installation option 149, 480
dynamic allocation
  activating and configuring 130, 460
  benefits 60
  BMCSORT allocation 281
  calculating data set size 301
  command options 190
  common defaults 466
  data set type 197, 470
  disk defaults 474
  examples 296, 386
  extended sequential data sets 198, 471
  installation options 464
  large format data sets 198, 471
  sizing unload data sets 441
  SMS 61
  syntax options 190
  tape defaults 478
  UNLOADDN_ACTIVE option 293, 421, 464
  using to avoid complex JCL 36
dynamic SQL
  DIRECT option 107
  example 369
  ISOLATION levels 59
  maximum number of tasks 155, 443
  overview 39
  specifying 106
dumped copies
  authorization requirements 53
  key data set for 452
  specifying 120
dumped data 85
ENUMROWS option
  description 148
  examples 346, 353
  performance considerations 422
error byte 269
error handling
  IF ERROR option 238
  using ON FAILURE to recover from abnormal termination 178, 181
ERROR keyword of IF option 238
ERRORBYTE fields 269
ESCAPE keyword of predicate 227
ESTROWS option
  description 232
  determining data set size during dynamic allocation 301
  determining number of records to allocate 304
example UNLOAD PLUS jobs
  clustering order, unloading in 320
  DELETEFILES processing 374
dynamic allocation of data sets, unloading 386
dynamic SQL, unloading 369
element table, unloading 313
elementary image copy, unloading 338
incremental image copy, unloading 357
LOADPLUS for DB2, unloading data to 396, 407
LOB data, unloading 378
multiple SELECT statements 346
multiple tables, unloading 325
nonstandard null indicators, unloading 378
partitioned table space, unloading 313
prior image copy, unloading 357
selected columns, unloading 320
table space, unloading 396
termination, handling an abnormal 374
XML data, unloading 378
example user exits 537
EXCLDUMP option 448
EXEC statement 272
executing UNLOAD PLUS jobs 307
execution phases of UNLOAD PLUS 42, 44
EXIT data type 253
exit parameters
  assembler DSECT 544
  C structure 560
  LE C structure 581
  record for COBOL II AND LE COBOL 601
exit routines
  assembler, sample 548
  authorization 85
exit routines (continued)
C, sample 565
calling in supervisor or problem state 462
COBOL II, sample 601
data conversion specification 253
data type conversion 253
discarded records, cause of 152
invoking DB2 user exits 84, 427
LE C, sample 585
LE COBOL, sample 601
parameter block DSECT 544
performance considerations 427
programming language 176
requirements 538
specifying 176
with FORMAT BMCLOAD option 76
EXPT option 215, 479
expiration date
disk data sets 213
relation to DISKRETN option 213
tape data sets 215, 479
extended address volume (EAV) data sets 29
EXTENDED BUFFER MANAGER. See XBM
extended partitioned data sets, for referenced files 197, 287, 471
extended sequential data sets 198, 471
Extensible Markup Language. See XML keyword of
FORMAT option or XML data
extents, specifying for secondary space using NBRSECD
option 210
external format, output in 73, 157
EXTERNAL keyword
numeric field considerations 263
of BIGINT data type 246
of DECIMAL data type 252
of DECIMAL data type 247
of FLOAT data type 248
of FORMAT option 84, 157
of GRAPHIC data type 244
of INTEGER data type 245
of SMALLINT data type 245
of TIME data type 251
of TIMESTAMP data type 250
of TIMESTAMP WITH TIME ZONE data type 250
with DATE data type 249
externalizing pages for table spaces 111, 443
EXTPREF keyword of DSNTYPE option 198, 471
EXTREQ keyword of DSNTYPE option 198, 471
field specification options 236
FIELDPROCs
how UNLOAD PLUS invokes 84
with FORMAT BMCLOAD option 76
fields
adding 508, 510, 513
assigning names for new constant type 508
binary 248, 251
century specification 442
caller 240
constant values 239
converting from columns 313
creating new during unload phase 310
data type specification 237
definitions of the output unload phase 217
delimiting 158
DSECT 545
fixed-length 174
FOCUS files 515
IF ERROR option 238
IF option 237
length variations 78
LOB data 251
mixed double-byte character set (DBCS) 240
nonnumeric 158
null indicator 78, 454
nullable 78
output record identification 237
output with NULL values 38
pseudo-SQLDA 539
separating character 158
specification option, with SELECT 236
SQLDA 541
SQLTYPE 541, 542
SQLVAR 541
TRIM and TRUNCATE options 78, 242, 248
VARBINARY 248
VARCHAR 242
variable-length 78
with FORMAT BMCLOAD 236
XML data 252
file referencing. See referenced files
FILEREFDN option 448
FILESZPCT option 211, 476
FILL option
command 164
installation 448
of BIGINT data type 246
of CHAR data type 240
of DECIMAL data type 247
of INTEGER data type 246
of SMALLINT data type 245
of VARCHAR data type 242
filtering partitions 116, 422

F

features of UNLOAD PLUS 35
FETCH, specifying multiple rows for one request 108, 455
FILTERPART option
   description 116
   limits opening of data sets 429
   performance considerations 422
fixed-length output records 78, 174
FIXEDVARCHAR option
   description 174
   example 320
   performance considerations 422
fixes applied, generating a report of 275
FlashCopy image copy 118
FLOAT data type 248
floating-point data types 248, 252
FOCUS keyword of CNTLCARDS option 142
FORCE option 187, 449
FORCE_AT option
   command, of FORCE option 187, 189
   installation 449
FORCE_DELAY option 188, 189
FORCE_RPT option 450
FORCE_RPT option of FORCE option 188
FORMAT options
   BMCLOAD keyword 73, 157, 422
   BMCUNLOAD keyword (in LOADPLUS) 407
   CSV keyword 81, 158
   description 156
   DSNTIAUL keyword 78, 158, 422
   examples 83, 366, 407
   EXTERNAL keyword 73, 157, 422
   INTERNAL keyword 73, 157, 422
   STANDARD keyword 72, 157, 422
   XML keyword 81, 159
formats, floating point 172
formats, syntax diagrams 21
FROM option 223, 313
full image copies, unloading from
   FULL option 121, 124
   table space status 57
FULL keyword
   of INFILE ddname option 124
   of INFILE IMAGECOPY option 121
G
GDEMPTY option 472
GDLIMIT option 202, 472
GDGs (generation data groups)
   for unload data sets 294, 295
   limiting number of versions 202, 472
   removing from the VTOC 472
   uncataloging 472
GDSCRATCH option 472
GENERATED ALWAYS columns 107, 266
   generating control statements
     DB2 LOAD 137, 138, 508
     Easytrieve 142, 525
   FOCUS 142
   LOADPLUS 140, 508, 512
   NOMAD 143, 535
   SAS 143, 518
   SQL/DS tables 143, 520
   Teradata tables 144, 527
generating DDL statements for DB2 tables 141, 514, 524
   generation data groups. See GDGs
   global trim option, with FORMAT CSV 159
   graphic characters 218
   GRAPHIC data type
      allowable data type conversions 255
      description 244
      relation to LIKE operator 227
   graphic strings rules for constants 229
   GRECP (group RECOVER pending) status 57
   group attachment name 273
   group RECOVER pending (GRECP) status 57
   guaranteed space 305
H
   hardware compression 487
   hardware copies, unloading from 125
   hash-organized tables 41
   hexadecimal strings rules for constants 229
   HFS files, for LOB and XML data
      allocating 287
      specifying 198, 471
      unloading to 63
   HFS keyword of DSNTYPE option 198, 471
   hierarchical file system. See HFS
   hierarchical file system. See HFS files
   high-speed data migration 157, 407
   high-transaction environment 431
   high-used RBA 301
   HISTORY installation option
      BMCHIST table 488
      description 451
   history table. See BMCHIST table
   &HOUR variable 194, 468
   HURBA keyword of ANALYZE option 132, 301
I
   I/O tuning 429
   IBM Resource Access Control Facility. See RACF
   IBUFFS option 424, 451
   ICTYPE column value 125
   identity columns, with FORMAT BMCLOAD 77
   IDs
      RACF 53, 454
      subsystem 114, 273, 462
      user 195, 275
INDEX 625

A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z

IDs (continued)
  utility 195, 274
  XBM 114, 462
IF NULL VALUE option 320
IF option with SELECT 237
image copies, unloading from
  base tables 66
  buffers for 424
  cabinet copies 118
  compression dictionary limitation 120, 123
  copy before most recent 121
  different subsystems 124
  encrypted 120
  examples 338, 357
  FlashCopy 118
  full 121, 124
  incremental 121, 125
  INFILE option 118
  inline 124, 451
  input data set, specifying the number of buffers 451
  Instant Snapshot copy 118
  online consistent copy 125
  SHRLEVEL CHANGE 111
  SHRLEVEL REFERENCE 110
  table space status 57
IMAGECOPY option 120
  implicit time zone 170
  IMPLICIT_TZ option 170
  IMPLICITLY HIDDEN columns 218
  IN option, with WHERE clause 227
  incompatible options, when unloading LOB or XML data 65, 68
incremental image copies
  example 357
  specifying 121, 125
INCREMENTAL keyword
  of INFILE ddname option 125
  of INFILE IMAGECOPY option 121
indexes
  ordering by 146
  versioned 75
INDSN option 407
INFILE option
  DDL specification 278
  ddname specification 279, 280
  description 118
  examples
    IMAGECOPY 338
    IMAGECOPY INCREMENTAL 358
    sequential data set 334, 361
    relation to ANALYZE processing 131
    SNAPCOPY option 125
inline image copies, unloading from 124, 451
inline LOB data 41
INLINE option 124, 451
installation options
  definitions and default values 438
  dynamic allocation 464
  multiple modules 438
  output descriptor 466
  setting for optimal performance 419
  specifying a module 276
Installation System 437
Instant Snapshot copy
  authorization requirement 53
  specifying 118
integer constants, rules for 228
INTEGER data type 245
internal BMC format, output in 73, 157
internal DB2 format, output in
  FORMAT INTERNAL 73, 157
  FORMAT STANDARD 72, 157
INTERNAL keyword of FORMAT option
  description 73, 157
  example 84
INTERVAL option 151, 330
INTO option
  description 220
  example 313
  inserting additional command options using the
  string parameter 506
invoking UNLOAD PLUS 307
IRLMWAIT, used with drain timeout 185
IS NOT NULL operator, with WHERE clause 227
IS NULL operator, with WHERE clause 227
ISOLATION levels for DB2 dynamic SQL 59

J

JCL
  building and executing UNLOAD PLUS jobs 271
  DD statements 276
  disposition for SYSREC data sets 288
  examples 309
  EXEC statement 272
  JOB statement 272
  REGION parameter 272, 430
  SYSIN data set 285
&JDATE variable 194, 468
&JDAY variable 194, 468
JOB statement 272
&JOBNAME variable 194, 468
jobs
  building 271
  canceling 308
  examples 309
  executing 307
  restarting 308
  terminating 308
Julian date variables 194, 468
K
key data set, for encrypted copies 452
KEYDSNAM option 452
keywords
$NO_UNITS option 192, 467
ACTIVE option 130
AGE keyword of ANALYZE DB2STATS option 131
ALL keyword
   of FORCE option 187, 449
   of ON FAILURE option 178
ANALYZE option 131, 441
ANY keyword of CURRENTDEGREE option 109
ASC keyword of ORDER BY option 232
ASCI1 option 165
ASM keyword of UNLOADEXIT option 176
AUTO keyword
   of DIRECT option 108
   of FORMAT XML SELECT_ELEMENT option 160
   of NBRSECD option 210
   of PCTPRIM option 208
   of RECFM option 136, 455
AUTOTAG option 152
BASIC keyword of DSNTYPE option 198, 471
BIGINT data type 246
BINARY data type 248
BINARY VARYING data type 248
BINARYXML keyword
   of CHAR data type 241
   of VARCHAR data type 243
BLOB data type 251
BLOBF keyword of character data types 241, 243
BMCLOAD keyword
   of CNTLCARDS option 140
   of FORMAT option 157
BYTASK keyword of ORDER YES option 147
C keyword of UNLOADEXIT option 176
CCSID option 166
CENTURY option
command 183
installation 442
   of DATE data type 249
   of TIMESTAMP data type 250
CHANGE keyword of SHRLEVEL CHANGE option 111
CONSTRULES option 444
CONTINUE UTILITY keyword of ON MESSAGE option 177
CSV keyword of FORMAT option 158
CURRENT DATE option 218, 229, 239
CURRENT RID option 219
CURRENT TIME option 239
CURRENT TIMESTAMP option 230, 239
CURRENTDEGREE option 109, 444
DATACLAS option 203
DATE data type 249
DATEFMT option 167
DB2 keyword of CNTLCARDS option 138
DB2DDL keyword of CNTLCARDS option 141
DB2LOAD keyword of CNTLCARDS option 137
DB2STATS keyword of ANALYZE option 131
DBCLOB data type 251
DBCLOBF keyword of character data types 241, 243
DDLDDN option 128
DECFLOAT data type 252
DECFLOAT_ROUNDING option 172
DECIMAL data type 247
DELETEFILES option 152
DELTFILES option 445
DESC keyword of ORDER BY option 232
DIR option 199
DIRECT option 106
DISABLED keyword of ZIIP option 114, 463
DISCARDS option 152
DISKEXP option 213, 478
DISKRETN option 213, 477
DRAIN_WAIT option 184
DRNDELAY option 446
DRNFAIL keyword of DSPLOCKS option 186
DRNRETRY option 446
DRNWAIT option 446
DSNAME option 193, 467
DSNTIAUL keyword of FORMAT option 158
DSNTYPE option 197, 470
DSPLCKS option 186, 447
EASYTRIEVE keyword of CNTLCARDS option 142
EBCDIC option 165
ENABLED keyword of ZIIP option 113, 463
ENCLOSEDBY option of FORMAT option 158, 161
ENUMROWS option 148
ERROR keyword of IF option 238
ESCAPE keyword of predicate 227
ESTROWS option 232
EXCLUDUMP option 448
EXIT data type 253
EXPDT option 215
EXTERNAL keyword
   of BIGINT data type 246
   of DATE data type 249
   of DECFLOAT data type 252
keywords (continued)

EXTERNAL keyword (continued)
  of DECIMAL data type 247
  of FLOAD data type 248
  of FORMAT option 157
  of GRAPHIC data type 244
  of INTEGER data type 245
  of SMALLINT data type 245
  of TIME data type 251
  of TIMESTAMP data type 250
  of TIMESTAMP WITH TIME ZONE data type 250

EXTPREF keyword of DSNTYPE option 198, 471
EXTREQ keyword of DSNTYPE option 198, 471
FILEREFD N option 448
FILESZPCT option 211
FILL option
  command 164
  installation 448
  of BIGINT data type 246
  of CHAR data type 240
  of DECIMAL data type 247
  of INTEGER data type 246
  of SMALLINT data type 245
  of VARCHAR data type 242
FILTERPART option 116
FIXEDVCHAR option 174
FLOAT data type 248
FOCUS keyword of CNTLCARDS option 142
FORCE option 187, 449
FORCE_AT option
  command, of FORCE option 187, 189
  installation 449
FORCE_DELAY option of FORCE option 188, 189
FORCE_RPT option
  command, of FORCE option 188
  installation 450
FORMAT option 156
FROM option 223
FULL keyword
  of INFILE ddname option 124
  of INFILE IMAGECOPY option 121
GDGEMPTY option 472
GDGLIMIT option 202, 472
GDGSCRATCH option 472
GRAPHIC data type 244
HFS keyword of DSNTYPE option 198, 471
HISTORY option 451
HURBA keyword of ANALYZE option 132
IBUFFS option 451
IF option 237
IMAGECOPY option 120
IMPLIED_TZ option 170
INCREMENTAL keyword
  of INFILE ddname option 125
  of INFILE IMAGECOPY option 121
INFFILE option 118
INLINE option 124, 451
INTEGER data type 245
INTERNAL keyword of FORMAT option 157
INTERVAL option 151
INTO option 220
KEYDSNAM option 452
LARGE keyword of DSNTYPE option 198, 471
LASTRETRY keyword of FORCE_AT option 188, 449
LE_C keyword of UNLOADEXIT option 176
LE_COBOL keyword of UNLOADEXIT option 176
LIBRARY keyword of DSNTYPE option 197, 471
LIKE operator 227
LIMIT option 151
LOADDECP option 452
LOCKROW option 452
LOGICAL option 127, 233
LRECl option 201
MAINT parameter of EXEC statement 275
MAXBLKSIZE option 135
MAXCONNECT option 155
MAXP option 453
MAXPRIM option 207, 475
MAXSECD option 209, 475
MAXSORTS option 154
MGMTCLAS option 204, 473
MIXED keyword
  of CHAR data type 240
  of VARCHAR data type 242
MSGLEVEL option 453
NAME option 220
NBRSECD option 210, 476
NEW parameter of EXEC statement 274
NOCOMP keyword of TRTCH option 214
NOLIMIT keyword of ANALYZE DB2STATS AGE option 132
NOMAD keyword of CNTLCARDS option 143
NONE keyword
  of CURRENTDEGREE option 109
  of DRAIN_WAIT option 184
  of DSNTYPE option 198, 471
  of DSPLOCKS option 186
  of FORCE option 187, 449
  of FORMAT XML SELECT_ELEMENT option 160
  of TRTCH option 214
NOSUBS option 166
NULL keyword
  of IF option 238
  of predicate 229
  of VALUE option 239
NULLCHAR option 162, 453
NULLSTRING option of FORMAT option 158, 161
NULLTYPE option 163, 454
Obid option 224
ON FAILURE option 178
ON MESSAGE option 177
OPNDB2ID option 454
keywords (continued)
  OPTIONS option 233
  ORDER BY option 231
  ORDER option 146
  OUTPUT command 190
  PACKED keyword of DECIMAL data type 247
  PARM keyword of EXIT data type 253
  PART option 127, 233
  PCTPRIM option 207, 474
  PDS keyword of DSNTYPE option 197, 471
  PLAN option 455
  QUIESCE option 111
  RECFM option 136, 455
  RECORDID option 222
  REFERENCE keyword of SHRLEVEL option 110
  REPORTONLY keyword of FORCE option 189
  RETCODE keyword
    of ON FAILURE TERMINATE UTILITY option 179
    of ON MESSAGE CONTINUE UTILITY option 177
  RETPD option 215, 479
  RETRY keyword
    of DSPLOCKS option 186
    of FORCE_AT option 188, 449
  RETRY option 185
  RETRY_DELAY option 185
  ROUND keyword
    of BIGINT data type 246
    of DECIMAL data type 247
    of FLOAT data type 248
    of INTEGER data type 246
    of SMALLINT data type 245
  ROUND_CEILING keyword of
    DECFLOAT_ROUNDMODE option 172
  ROUND_DOWN keyword of
    DECFLOAT_ROUNDMODE option 172
  ROUND_FLOOR keyword of
    DECFLOAT_ROUNDMODE option 172
  ROUND_HALF_DOWN keyword of
    DECFLOAT_ROUNDMODE option 172
  ROUND_HALF_EVEN keyword of
    DECFLOAT_ROUNDMODE option 172
  ROUND_HALF_UP keyword of
    DECFLOAT_ROUNDMODE option 173
  ROUND_UP keyword of
    DECFLOAT_ROUNDMODE option 173
  ROWSETSZ option 108, 455
  RTRIM option 159
  SAS keyword of CNTLCARDS option 143
  SDUMP option 456
  SELECT statement 217
  SELECT_ELEMENT option 159
  semicolon 234
  SHRLEVEL option 110, 456
  SMALLINT data type 245
  SMAX option 456
  SMCORE option 457
  SNAPCOPY option 125
  SORTDEVT option 149
  SORTNUM option 149, 458
  SPACE option 207, 474
  SQL keyword of DRAIN_WAIT option 185
  SQL/DS keyword of CNTLCARDS option 143
  SQL/DS-DDL keyword of CNTLCARDS option 143
  SQL/DS-LOAD keyword of CNTLCARDS option 144
  SQLDELAY option 458
  SQLRETRY option 458
  STANDARD keyword of FORMAT option 157
  START keyword of FORCE_AT option 188, 449
  STOP UTILITY keyword
    of ON FAILURE option 180
    of ON MESSAGE option 177
  STORCLAS option 203, 473
  SUBSETS option 199
  SUPSTART keyword of WТОMSG option 115
  SYNC option 155
  TABLESPACE option 104
  TAPEDISP option 459
  TAPES option 459
  TASKMAX option 459
  TERADATA keyword of CNTLCARDS option 144
  TERADATA-BULK keyword of CNTLCARDS option 144
  TERADATA-FAST keyword of CNTLCARDS option 144
  TERADATA-MULT keyword of CNTLCARDS option 145
  TERM parameter of EXEC statement 275
  TERMINATE UTILITY keyword of ON FAILURE option 178
  TERMINATEDBY option of FORMAT CSV option 158
  TIME data type 251
  TIMEFMT option 167
  TIMESTAMP data type 250
  TIMESTAMP WITH TIME ZONE data type 250
  TRIM option
    of BINARY data type 248
    of CHAR data type 241
    of VARGRAPHIC data type 244
    of VARGRAPHIC data type 244
    of VARCHAR data type 244
    of VARGRAPHIC data type 244
    of VARGRAPHIC data type 244
  TSFMT option 168
  UBUFFS option 459
  UNICODE option 165
  UNIT option 192, 467
  UNITCNT option 205, 474
keywords (continued)
UNLDMAX option 459
UNLOAD command 104
UNLOAD keyword of ON FAILURE option 178
UNLOADDN option 129, 460
UNLOADDN_ACTIVE option 460
UNLOADEXIT option 176
USELRECL option 134, 461
UTIL keyword of DRAIN_WAIT option 185
UTILINIT keyword of ON FAILURE option 178
UTILTERM keyword of ON FAILURE option 178
UXSTATE option 462
VALUE option 239
VARBINARY data type 248
VARCHAR data type 242
VARGRAPHIC data type 244
VB keyword of RECFM option 136, 455
VOLCNT option 201, 471
VOLUMES option 212, 477
VSAMDD option 125
VSAMDDPREFIX option 126
WHERE option 225
WORKUNIT option 462
WTOMSG option 115
XBMID option 114, 462
XML data type 252
XML keyword of FORMAT option 159
ZIIP option 113, 463
ZONED keyword of DECIMAL data type 247
ZONEDDECOVP option 171, 464

L
language for user exits 176
large format sequential data sets 198, 471
LARGE keyword of DSNTYPE option 198, 471
LASTRETRY keyword of FORCE_AT option 188, 449
LBI data sets 424, 451
LE_C keyword of UNLOADEXIT option 176
LE_COBOL keyword of UNLOADEXIT option 176
length option
with BINARY data type 248
with CHAR data type 241
with DATE EXTERNAL data type 249
with referenced file subtypes 241, 243
with TIME EXTERNAL data type 251
with TIMESTAMP EXTERNAL data type 250
with VARBINARY data type 249
with VARCHAR data type 243
with VARGRAPHIC data type 244
LIBRARY keyword of DSNTYPE option 197, 471
LIKE operator 227
limit key specifications in DDL 122
LIMIT option
calculating data set size 301
description 151
determining number of records to allocate 304
examples 320, 330
limiting retry attempts 185, 446, 458
line feed characters, in FORMAT CSV options 80
load utilities, control statements 137, 221
LOADDECP option 452
LOADPLUS
control cards, from UNLOAD PLUS 294, 422
example of unloading data using UNLOAD PLUS 396, 407
high-speed migration using UNLOAD PLUS 157
inserting additional command options into LOAD control cards 506
loading CSV data 81
LOB data
BMCSYNC table considerations 494
data types 251
data types for referenced files 241, 243
discards due to length, limiting 152
example 378
field specification 63, 64
inline 41
performance considerations 428
unloading to referenced files 63
unloading to unload data sets 62
LOCKROW option 452
locks, table 59, 458
log range table 490
LOGICAL option 127, 233
logical page list (LPL) status 57
logical part REBUILD pending (RBDP*) status 57
logical partitions 127, 233, 469
long object names 88
LPL (logical page list) status 57

M
MAINT parameter 275
maintaining common utility tables 484
management classes, SMS 204, 473
management set, XBM 62
MAXBLKSIZE option 135, 422
MAXCONNECT option 155, 443
maximum disk allocation
primary 208, 475
secondary 209, 475
MAXP option 426, 453
MAXPRIM option
command 208
installation 475
relation to specifying guaranteed space algorithm 306
relation to standard SMS algorithm 305
specifying when calculating space for SMS-managed data sets 304
using to override the PCTPRIM calculations 207
MAXSECD option
command 209
installation 475
relation to FILESZPCT option 211
relation to SPACE option 209
MAXSORTS option
concurrent sort reduction 426
contention reduction 218
description 154
relation to SMAX installation option 456
sort processing 434
MEMBER CLUSTER, table spaces defined with 23, 27
MEMLIMIT system parameter 53, 272, 430
memory allocation for sort processing 282, 425
message level
EXEC statement parameter 275
installation option 453
messages
changing output displayed 275, 453
MSGLEVEL 275, 453
performance-related 431
viewing in worklist environment 286
MGMTCLAS option 204, 473
migration, high-speed DB2 to DB2 157, 407, 422
&MIN variable 194, 468
&MINUTE variable 194, 468
MIXED keyword
of CHAR data type 240
of VARCHAR data type 242
modes of execution 39
MODESET SVC, impact on performance 427
&MONTH variable 194, 468
MSGLEVEL
EXEC statement parameter 275
installation option 453
multiple ON FAILURE statements 179
multiple SELECT statements 123
multiple BYTE characters, in FORMAT CSV options 80
multi-row fetch 108, 455
multi-table table spaces 123
multitasking
description 427
effect on ENUMROWS option 148
example 320
parameters information 459
MVS cancel command 308
MVS system log, writing to using WTOMSG 115

null indicator
character, position, and length 38
field 454
NULLCHAR option 162, 453
NULLSTRING option 158, 161
NULLTYPE option 163, 454
when to use if column is nullable 237
null fields 314, 508, 513
NULLCHAR option 162, 453
NULLSTRING option of FORMAT option 158, 161
NULLTYPE option 163, 454
numeric data types, converting to external 164, 448

OBID option 224, 361
&OBID variable 194, 468
OBID, specifying
for INFILE ddname 123
in DDLIN 278
with OBID option 224
object availability, SHRLEVEL option 110, 456
object names, long 88
object status 57
ON FAILURE option 178, 374
ON MESSAGE option 177
online consistent copy, specifying 125
operating system requirements 50
operators 225, 226
OPNDB2ID option 53, 454
optimizing
CMAX option 443
memory amount 425
optimizing (continued)
  memory use 434
  performance 419
  primary and secondary extent sizes 205
  SMAX option 457
  tables in segmented table spaces 428
  tasks during sort processing 432
options, command
  See also keywords
  dynamic allocation 190
  field specification 236
  OUTPUT command 190
  setting for optimal performance 421
  unload processing 89
options, incompatible, when unloading LOB or XML data
  65, 68
options, installation
  definitions and default values 437
  multiple modules 276, 437
  setting for optimal performance 419
  specifying module 276
ORDER BY option 231
ORDER option
  description 146
  examples 320, 325
output
  BMC internal format 73, 157
  CSV format 79, 158
  data formats 71
  DB2 internal format
    FORMAT INTERNAL 73, 157
    FORMAT STANDARD 72, 157
  device default 467
  DSNTIAUL format 78, 158
  examples of format 83
  external date/time data 72, 157
  external format 73, 157
  for LOADPLUS 73
  format 71, 156
  internal date/time data 73, 157
  records, tagging 152
  specifying format 156
  XML format 81, 159
OUTPUT command 190
output data sets
  default ddname, specifying 460
  multiple 287, 289
  record format, specifying 136, 455
  referenced files 287
  SYSREC option 286
output descriptors
  &PART variable 296
  &SELNUM variable 296
  adjusting the total calculated space 211, 476
  allocation units 207, 474
  default volume list 212, 477
  descriptor name 190, 466
expiration date, disk data sets 213, 478
expiration date, tape data sets 215, 479
file type 197, 470
installation options 466
length 241, 243
matching to field specification 67
matching to FILEREFDN 448
matching to multiple SELECT statements 296
number of volumes 201, 471
retention period, disk data sets 213, 477
retention period, tape data sets 215, 479
secondary allocation size 210, 476
SMS classes 203, 473
type of file 197, 470
volume serial numbers 305
output records
  identifying 220
  tagging 152, 222
overpunch values, assigning 39, 171, 464

P
packages 51, 455
PACKED keyword of DECIMAL data type 247
padding, order of 269
page set REBUILD pending (PSRBD) status 57
parameters, EXEC statement 273
PARM keyword of EXIT data type 253
PART option
  description 127, 233
  examples 334, 338, 358
  performance considerations 423
&PART variable
  description 195, 469
  examples 300, 387
partition-by-growth table spaces 117, 147, 231
partitioned data sets, for referenced files 197, 287, 471
partitioned table spaces
  clustering order unload example 320
  ENUMROWS option 148
  INFILE IMAGECOPY option 120
  MAXP option 453
  MAXSORTS option 154
  multiple data sets 429
  multitasking performance 427
  partition-by-growth table spaces
    with FILTERPART 117
    with INFILE ddname 124
    with ORDER BY 231
    with ORDER YES 147
  performance considerations 423, 427
range-partitioned 124
sorting 148
specifying the partitions to be unloaded 127
unloading to separate data sets 289
with INFILE ddname 124
partitioning key, ordering by 146, 407
partitions
   logical
      specifying 127, 233
      with symbolic variables 195, 469
      rebalanced 122
      rotated 122
      specifying 127, 233
PCTPRIM option
   command 207
   installation 474
   relation to FILESZPCT option 211
   relation to the guaranteed space algorithm 306
   relation to the standard SMS algorithm 305
PDS keyword of DSNTYPE option 197, 471
PDS or PDSE, for referenced files 197, 287, 471
performance considerations
   buffer usage, controlling 423
   calculating SORTWK data set size 282
   command options, specifying 421
   DB2 user exits 427
   I/O processing 429
   installation options, specifying 419
   LOB data 428
   MAXSORTS option 154
   multiple table spaces 428
   multiple unload data sets 36, 429
   multitasking 427
   obtaining maximum 39
   related messages 431
   virtual storage 430
   XML data 428
phases of UNLOAD PLUS 42, 44
PLAN option 455
porting data to other databases
   FORMAT CSV option 158
   FORMAT XML option 159
position number variable 195, 469
&POSNUM variable 195, 469
precision option
   of DECFLOAT data type 252
   of DECIMAL data type 247
precision, in timestamp columns 24
predicate
   of IF condition 239
   of WHERE condition 226
PRELOAD LOAD option in LOADPLUS, specifying for performance 422
PRELOAD WHITESPACE string 138
primary data sets, dynamic allocation 460
primary table space
   allocating using PCTPRIM option 474
   calculating using the guaranteed space algorithm 306
   calculating using the standard SMS algorithm 305
   overriding the calculated value with MAXPRIM option 207
   using PCTPRIM option to set a value for the percentage 207
privileges
   DISPLAY 52
   READ 52
processing phases 42, 44
processor requirements 50
product changes 22
product support 3
programming language, of user exit routines 176
pseudo-SQLDA
   assembler user exit 546
   C user exit 563, 584
   COBOL user exit 603
PSRBD (page set REBUILD pending) status 57
publications, related 19
Q
QUIESCE option 58, 111
R
RACF (IBM Resource Access Control Facility)
   authorization needed to execute UNLOAD PLUS 53
   ID 454
   OPNDB2ID option 454
   security exit 51
RBA, high-used (HURBA) 132
RBDP (REBUILD pending) status 57
RBDP* (logical part REBUILD pending) status 57
read claimers, canceling 187, 449
READ privileges 52
read/write (RW) status, SHRLEVEL CHANGE option 110, 456
read-only (RO) status, SHRLEVEL REFERENCE option 110, 456
read-or-replication-only (RREPL) status 57
REBUILD pending (RBDP) status 57
RECFM option 136, 455
record format, specifying 136, 455
record identifiers (RIDs)
   selecting to include in output 219
   specifying a value for 222, 347
RECORDID option 222, 347
RECOVER pending (RECP) status 57
recovery from abnormal termination 178
Recovery Management for DB2 solution, cabinet copies from 118
RECP (RECOVER pending) status 57
REFERENCE keyword of SHRLEVEL option 110
referenced files
deleting 69
how to unload to 63
multiple 287, 428
performance considerations 428
specifying 67
specifying name 193, 468
specifying number to allocate 199
specifying OUTPUT statements for 63
specifying type 197, 470
using 63, 287
REFP (refresh pending) status 57
refresh pending (REFP) status 57
REGION parameter 272, 430
related publications 19
REORG pending (REORP) status 57
REORP (REORG pending) status 57
REPORTONLY keyword of FORCE option 189
requirements
authorization 51
software 50
system 50
reserved words, delimiting with quotation marks 88
restart parameters 274
restart pending (RESTP) status 57
restarting UNLOAD PLUS 308
RESTP (restart pending) status 57
RETCODE keyword
of ON FAILURE TERMINATE UTILITY option 179
of ON MESSAGE CONTINUE UTILITY option 177
retention period
of disk data sets 213, 477
of tape data sets 215, 479
RETPD option 215, 479
retry attempts, limiting 185, 446, 458
RETRY keyword
of DSPLOCKS option 186
of FORCE_AT option 188, 449
RETRY option 185
RETRY parameter of DYNALOC option 482
RETRY_DELAY option 185
retrying dynamic allocation of SORTWK 482
return codes
-911, specifying wait time before retry 458
specifying 177, 179
RIDs. See record identifiers
RO (read-only) status, SHRLEVEL REFERENCE option 110, 456
ROUND option
of BIGINT data type 246
of DECIMAL data type 247
of FLOAT data type 248
of INTEGER data type 246
of SMALLINT data type 245
ROUND_CEILING keyword of DECFLOAT_ROUNDMODE option 172
ROUND_DOWN keyword of DECFLOAT_ROUNDMODE option 172
ROUND_FLOOR keyword of DECFLOAT_ROUNDMODE option 172
ROUND_HALF_DOWN keyword of DECFLOAT_ROUNDMODE option 172
ROUND_HALF_EVEN keyword of DECFLOAT_ROUNDMODE option 173
ROUND_HALF_UP keyword of DECFLOAT_ROUNDMODE option 173
ROUND_UP keyword of DECFLOAT_ROUNDMODE option 173
rounding 172, 265
routines, user exit
assembler, sample 548
authorization 85, 538
C, sample 565
calling in supervisor or problem state 462
COBOL II, sample 601
data conversion specification 253
data type conversion 253
discarded records, cause of 152
invoking DB2 user exits 84, 427
LE C, sample 585
LE COBOL, sample 601
parameter block DSECT 544
performance considerations 427
programming language 176
requirements 538
specifying 176
unload exit specification 176
with FORMAT BMXLOAD option 76
ROWID data type 257, 266
row-level security 41, 218
rowset, specifying 108, 455
rowset-positioning cursors 108, 455
ROWSETSZ option 108, 455
RREPL (read-or-replication-only) status 57
RTRIM option 159, 237
rules for command syntax 87
running BMC utilities concurrently 55
running UNLOAD PLUS 307
RW (read/write) status, SHRLEVEL CHANGE option 110, 456
S
sample UNLOAD PLUS jobs 309
sample user exits
assembler 548
C 565
sample user exits (continued)
  COBOL II 605
  LE C 585
  LE COBOL 605
  where to find 539
sampling data, unloading 151
SAS control statements
  example 334
  specifying 143, 518
SAS keyword of CNTLCARDS option 143
scale, specifying on numeric fields 264
SCC (DB2 Solution Common Code) 48, 50, 276
SDUMP option 456
&SEC variable 195, 469
&SECOND variable 195, 469
secondary allocation, specifying 209, 210
secondary data sets
  description 47
  dynamic allocation 190, 460
security
  mechanisms 51
  required 51
  row- and column-level 41, 218
  using packages other than RACF 53
  using RACF 53
segmented table spaces 36, 428
SELECT statements
  description 217
  examples
    columns in clustering order 320
    data from DSN1COPY 334
    full image copy data 339
    incremental image copy 358
    multiple table spaces 353
    OBID and DDLIN data set use 361
    partitions to separate data sets 313
    prior image copy use 358
    sample of catalog rows 330
    multiple 65, 123, 140
    multiple output data sets 291
    multiple SYSCNTL data sets 284
    performance considerations 423
&SELNUM variable 195, 299, 469
semicolon specification when using DIRECT NO 234
sequential data sets, extended 198, 471
serialization 55
SHRLEVEL column of BMCSYNC table 492
SHRLEVEL option
  CHANGE keyword 58, 110, 330
  command 110
  installation 456
  REFERENCE keyword 58, 110
SMALLINT data type 245
SMAX option 434, 456
SMCORE option
  description 457
  performance considerations 426
  recommendation with large SORTNUM values 150
SMS (Storage Management Subsystem)
DATACLAS 203, 305, 473
extended sequential data sets 198, 471
managed data sets 303
MGMTCLAS 204, 473
not passing a unit value 192, 467
standard algorithm for space allocation of data sets 305
STORCLAS 203, 473
SNAPCOPY option 125
SnapshotUnload. See SNAPSHOTTUpgrade FEATURE
SNAPSHOT UPGRADE FEATURE (SUF)
  description 61
  example 330
  initialization, writing a message to the MVS system log 115
  licensing 61, 113, 463
  SHRLEVEL CHANGE CONSISTENT YES 112
  XBM subsystem ID 114, 462
  zIIP processing 113, 463
software requirements 50
solution integration 33
sort processing
  BMCSORT 148, 149, 281
  installation options 438
  memory allocation 425, 457
  multitasking limitations 427
  single task conditions 147
sort routine. See BMCSORT
sort work data sets (SORTWK)
  allocating dynamically 149, 458
  calculating size 282
  DD statements 281
  deleting 181, 445
  description 46
  unsupported data set types 283
SORTDEVT option 149, 325
sorting
  installation options for controlling 426, 456
  memory usage 434
  ORDER BY option 231
  ORDER option 146
  partitioned table spaces 148
  restricting concurrent tasks 154
  SMAX option 456
  work space, calculating 282
SORTNUM option
  command 149
  installation 458
  performance considerations 422, 434
SORTWK data sets
   allocating dynamically 149, 458
   calculating size 282
   DD statements 281
   deleting 181, 445
   description 46
   unsupported data set types 283

SPACE option
   calculating data set size during dynamic allocation 301
   command 207
   installation 474
   relation to MAXSECD option 209

spanned variable block (VBS) file
   preventing with MAXBLKSIZE 135
   when specifying RECFM option 136

special characters 80
specifying DD statements 276
specifying options modules 276
specifying referenced files
   in OUTPUT statement 63
   matching to field specification 67
   specifying name 193, 468

SQL statements
   deleting rows from the BMCDICT table 488
   deleting rows from the BMCHIST table 490
   deleting rows from the BMCSYNC table 494, 498
   deleting rows from the BMXCOPY table 503
   displaying BMC utilities 486
   querying BMCHIST table 486
   querying BMXCOPY table 486
   specifying the number of times to retry statement 458
   terminating BMC utilities 486

SQL, dynamic, DB2 107

SQL/DS
   control statements, generating 522
   CREATE TABLE DDL, generating 524
   keyword of CNTLCARDS option 143
   SQL/DS-DDL keyword of CNTLCARDS option 143, 524
   SQL/DS-LOAD keyword of CNTLCARDS option 144, 520
   SQLDELAY option 458
   SQUIRETRY option 458
   SSID (DB2 subsystem identifier) parameter of EXEC
   statement 273
   &SSID variable 114, 195, 469

STANDARD keyword of FORMAT option 72, 83, 157
START keyword of FORCE_AT option 188, 449

statistics
   estimating using ANALYZE option 131
   obtaining from the DB2 catalog using DB2STATS 131
   setting a limit on the age used to calculate data set
   sizes 132
   using HURBA to estimate the number of rows for a
   table or partition 132
   using the catalog information instead of the age using
   NOLIMIT option 131

status of table space, initial 57
status, BMC utilities 486

STEPLIB DD statement 276
&STEPNAME variable 195, 469

STOP UTILITY option 177

storage groups 304
storage, virtual 53, 272, 430

STORCLAS option 203, 473

string constants 78
striped data sets 85

SYMBYTE fields 269
SUBSETS option 199, 378

subsystem ID specification
   EXEC statement parameter 273
   for XBM subsystem 114, 462

SUF. See SNAPSHOT UPGRADE FEATURE
summary of changes 22

support, customer 3

supported DB2 versions 50

symbolic variables, for constructing DSNAMEs 193, 294,
   467

SYNC option 155

synonyms, specifying in FROM option 223

syntax
   description of options 103
   diagrams 92
   examples 309
   rules for specifying 87

syntax, format for diagrams 21

SYSCNTL data sets
   DCB attributes 284
   DD statements 284
   deleting 181, 445
   description 46, 284
   multiple, specifying 284
   single, when specifying CNTLCARDS BMCLOAD 140
   type of control statements written to 137

SYSDISC option of DELETEFILES option (in LOADPLUS) 407

SYSIBM.SYSCOPY table 111, 120, 125

SYSIBM.SYSTRINGS table, translation processing 266

SYSIN data set
   description 46, 47
   output statements 190

SYSOUT specification
   for SYSPRIN2 286
   for UTPRINT 307

SYSPRIN2 data sets
   changing message levels 275, 453
   description 46, 47
   specification and usage 285
   worklist environment considerations 286
SYSPRINT data set
   changing message levels 275, 453
   description 46, 47
   specification and usage 285
SYSREC data sets
   block size, overriding 135
   buffer usage, controlling 424
   calculating size 288
   changing default name 129, 286
   DD statements 286
   deleting 181, 445
   description 46, 47
   DISP parameter 288
   dynamic allocation 36, 38
   LOB data 62
   multiple 289
   overriding default name 286
   performance considerations 429
   record format, specifying 136, 455
   UBUFFS parameter recommendations 424
   XML data 62
SYSREC keyword of UNLOADDN option 129
SYSRED data sets 286
   See also SYSREC data sets
SYSRED keyword of UNLOADDN option 129
system authorization to execute UNLOAD PLUS 52
system dump
   excluding abend codes from 448
   generating with SDUMP option 456
system requirements 50
system setup 50
system sort routine DD statements 307
System z Integrated Information Processor (zIIP). See zIIP processing
SYSTEMPAGES NO, copies defined with 120, 123
system-period temporal tables
   CNTLCARDS BMCLOAD 141
   CNTLCARDS DB2 139
   CNTLCARDS DB2DDL 142
   CNTLCARDS DB2LOAD 138
   FORMAT BMCLOAD 74
T
   table locks 59, 458
   table names, determining 485
   table spaces
      calculating estimated number of rows 133
      externalizing pages for using CHANGE_QUIESCE option 443
      multi-table 123
      partition-by-growth
         with FILTERPART 117
         with INFILE ddname 124
   with ORDER BY 231
   with ORDER YES 147
partitioned
   examples 313, 320, 342
   MAXSORTS option 154
   multiple data sets 429
   multitasking performance 427
   separate data sets unload 289
   sorting 148
   range-partitioned 124
   segmented 148, 428
   status and concurrency 57
   unloading entire 104, 396
   versioned 75
   table structure for FORMAT BMCLOAD option 75
tables
   application-period temporal
      CNTLCARDS BMCLOAD 141
      CNTLCARDS DB2 139
      CNTLCARDS DB2DDL 142
      CNTLCARDS DB2LOAD 138
   bitemporal
      CNTLCARDS BMCLOAD 141
      CNTLCARDS DB2 139
      CNTLCARDS DB2DDL 142
      CNTLCARDS DB2LOAD 138
      FORMAT BMCLOAD 74
   clone 29, 105, 223
   organized by hash 41
   specifying 220, 223
   system-period temporal
      CNTLCARDS BMCLOAD 141
      CNTLCARDS DB2 139
      CNTLCARDS DB2DDL 142
      CNTLCARDS DB2LOAD 138
      FORMAT BMCLOAD 74
temporal
      CNTLCARDS BMCLOAD 141
      CNTLCARDS DB2 139
      CNTLCARDS DB2DDL 142
      CNTLCARDS DB2LOAD 138
      description of support 23
      FORMAT BMCLOAD 74
   tables, BMC
      backing up 484
      BMCDICT 487
      BMCHIST 488
      BMCLGRNX 490
      BMCSYNC 491
      BMCTRANS 495
      BMCUUTIL 496
      BMCCOPY 495, 499
      considerations 484
      determining names 485
tables, BMC (continued)
   querying 486
   warnings 484
TABLESPACE option 104, 396
tagging records
   AUTOTAG option 152
   RECORDID option 222
tape data sets
   device identification 459
dynamic allocation options 191, 478
   expiration date 479
   options, dynamic allocation 214
   retention period 215, 479
   specifying 465
   using data compression 214, 479
tape files, final disposition 459
   TAPEDISP option 459
   TAPES option 459
tapes. See tape data sets or tape files
   TASKMAX option 459
tasks that UNLOAD PLUS performs 36
technical support 3
temporal tables
   CNTLCARDS BMCLOAD 141
   CNTLCARDS DB2 139
   CNTLCARDS DB2DDL 142
   CNTLCARDS DB2LOAD 138
   description of support 23
   FORMAT BMCLOAD 74
TERADATA keyword of CNTLCARDS option 144, 534
Teradata, generating control statements for 144, 527
TERADATA-BULK keyword of CNTLCARDS option 145, 528
TERADATA-FAST keyword of CNTLCARDS option 145, 531
TERADATA-MULT keyword of CNTLCARDS option 145, 533
TERM parameter of EXEC statement 275
TERMINATEDBY option of FORMAT CSV option 80, 158
terminating a job 308
terminating BMC utilities 486
termination, recovering from abnormal 178
threads, canceling
   authorizations required 52
   FORCE option 187, 449
   FORCE_AT option 449
   FORCE_RPT option 450
   reporting 188, 277, 450
   specifying when to cancel 187, 449
threads, number used by UNLOAD PLUS 54
   CURRENT TIME option 239
   exit routines 84
   field 251
   formats 255
   overriding the default DB2 external format 167
rules for constants 229
   variables 194, 468
TIME data type 251
   &TIME variable 195, 469
time zone
   data type 250
   implicit 170
   support 24
TIMEFMT option 167
timeout
   displaying locks 186, 447
   number of retry attempts 185, 446
   specifying 184, 446
   time between retry attempts 185, 446
timestamp
   CURRENT TIMESTAMP option 230, 239
   default DB2 external format, overriding 168
   field 250
   formats 254
   with precision 24
   with time zone 24, 170, 250
TIMESTAMP data type 250
TIMESTAMP WITH TIME ZONE data type 250
tokens in syntax, rules for 88
trailing blanks, removing
   RTRIM option 159
   TRIM option 241, 244, 248
translation
   command constants 266
   data 266, 422
   order of 269
   processing 266
   restrictions 267, 268
   specifying 165
   supported 267
   SYSIBM.SYSSTRINGS table 266
   turning off substitution characters during translation
   between CCSIDs 166
   TRANSTAB field 267
TRIM option
   of BINARY data type 248
   of CHAR data type 241
   of VARCHAR data type 249
   of VARCHAR data type 244
trimming options
   RTRIM option 159
   TRIM option
      of BINARY data type 248
      of CHAR data type 241
      of VARCHAR data type 249
      of VARCHAR data type 244
TRTCH option 214, 479
TRUNCATE option
   of BINARY data type 248
   of CHAR data type 242
   of GRAPHIC data type 244
UNLOAD PLUS for DB2 Reference Manual

TRUNCATE option (continued)
of VARBINARY data type 249
   of VARCHAR data type 244
   of VARGRAPHIC data type 244

   truncating
      binary values 248
      character values 242
      DECIMAL values 172
   graphic values 244

   &TS variable 195, 469
   TSFMT option 168
   TSO cancel command 308
   tuning UNLOAD PLUS
      I/O 429
      memory use 434
      performance 419
   &TYPE variable 195, 469

U

UBUFFS option 459

Unicode
   examples 396
   option to support 165
   restrictions 85
   supported translation 267
   translation process 267
   UNLOAD TABLESPACE option 104
   unloading catalog data 70

   UNICODE option 165
   union in view 223
   unit count default for dynamic allocation 474

   UNIT option 192, 467

   UNITCNT option 205, 474

   universal table spaces 117, 147, 231

   UNLDMAX option 459

   UNLOAD command 104

   UNLOAD phase 42, 44

   UNLOADADDN option
      command 129
      example 346
      installation 460

   UNLOADADDN_ACTIVE option 421, 460

   UNLOADEXIT option 176

   unloading
      catalog data 70
      data for LOADPLUS 73, 407
      entire table space 33
      LOB data 62
      XML data 62

   unsupported features of DB2 85

   USELRECL option
      command 134
      installation 461
      performance considerations 423

   user exit routines
      assembler, sample 548
      authorization 85, 538
      C, sample 565
      calling in supervisor or problem state 427, 462
      COBOL II, sample 601
      data conversion specification 253
      data type conversion 253
      discarded records, cause of 152
      invoking DB2 user exits 84, 427
      LE C, sample 585
      LE COBOL, sample 601
      parameter block DSECT 544
      performance considerations 427
      programming language 176
      requirements 538
      specifying 176
      unload exit specification 176
      with FORMAT BMLOAD option 76

   user ID parameter of EXEC statement 275

   user-defined data types 41

   &USERID variable 195, 469

   &UTIL variable 195, 470

   UTILINIT keyword of ON FAILURE option 178

   UTILINIT phase 42, 44

   utility identifier parameter of EXEC statement 274

   utility parameters of EXEC statement 273

   utility restrictive state, utility exclusive control (UTUT) 57

   UTILTERM keyword of ON FAILURE option 178

   UTILTERM phase 42, 44

   UTPRINT data set 47, 307

   UTUT (utility restrictive state, utility exclusive control) 57

   UXSTATE option 462

V

   VALIDPROCs with FORMAT BMLOAD option 76

   VALUE option 239, 263, 320

   VARBINARY data type 248

   VARCHAR data type 242, 265

   VARGRAPHIC data type 244, 265

   variable block (VB) record format, specifying 136, 455

   variable-block spanned (VBS) file
      when specifying RECFM option 136
      when UNLOAD PLUS calculates record size 288

   variables
      construction of data set names 194, 294, 468
      differences when using with DIRECT YES or DIRECT
         NO 196
      in output descriptors 196
      resolving when using DIRECT NO 196

   user-defined 176

   VB keyword of RECFM option 136, 455

   versioning, index and table space 75

   versions of DB2 50
versions, XML 27
views, specifying 223
virtual storage 53, 272, 430
VOLCNT option
  command 201
  installation 471
  relation to VOLUMES option 212
volumes
  serial numbers 305
  specifying a list of 212, 477
  specifying number of 201, 471
VOLUMES option 212, 477
VSAM data sets, unloading from 47, 125, 126
VSAMDD DD statements 280
VSAMDD keyword of INFILE option 125
VSAMDDPREFIX keyword of INFILE option 126

W
WEPR (write error page range) status 57
WHERE clause
  condition of SELECT option 225, 334
  performance impact 36
work data set, specifying with WORKUNIT option 462
work files
  deleting 181, 445
  dynamically allocating 130
worklist environment
  DELETEFILES option 181
  DELFILES option 445
  NULLCHAR option 453
  NULLTYPE option 454
  viewing utility output 286
WORKUNIT option 462
write claimers, canceling 187, 449
write error page range (WEPR) status 57
WTOMSG option 115

X
XBM (EXTENDED BUFFER MANAGER)
  example 330
  licensing 61, 113, 463
  management set 62
  purpose and use 61
  SHRLEVEL CHANGE CONSISTENT YES 112
  XBM subsystem ID 114, 462
  zIIP processing 113, 463
XBMGROUP name 114, 462
XBMID option
  command 114
  example 330
  installation 462
XML data
  binary 241, 243
  BMCSYNC table considerations 493
  data type 252
  data types for referenced files 241, 243
  discards due to length, limiting 152
  example 378
  field specification 63, 64
  output format 81
  performance considerations 428
  PRESERVE WHITESPACE string 138
  unloading to referenced files 63
  unloading to unload data sets 62
  versions 27
XML data type 252
XML keyword of FORMAT option
  description 81, 159
  example of unloaded data when using 84
  specifying a string 161
XML output
  considerations and restrictions 81
  specifying 159
XML type modifier 85

Y
&YEAR variable 195, 470

Z
z/OS Unicode Conversion Services 267
zeros
  with FILL option 164, 240
  with FIXEDVARCHAR option 174
  with TRIM option 248
  with TRUNCATE option 248
ZIIP option 113, 463
zIIP processing
  enabling and disabling 113, 463
  requirements 51
  specifying an XBM subsystem 114, 462
zoned decimal data
  assigning overpunch values using ZONEDDECOVP option 171, 464
  data type 247
  example 392
ZONEDDECOVP option
  command 171
  example 392
  installation 464