Log Master for DB2®
Reference Manual

Supporting

Version 11.1 of Log Master for DB2
Version 11.1 of Recovery Management for DB2

June 2013
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

Address  
BMC SOFTWARE INC  
2101 CITYWEST BLVD  
HOUSTON TX 77042-2827  
USA

Telephone  
1 713 918 8800 or 1 800 841 2031

Fax  
1 713 918 8000

Outside United States and Canada

Telephone  
+01 713 918 8800

Fax  
+01 713 918 8000

© Copyright 1996-2013 BMC Software, Inc.

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.

CICS, DB2, DFSMS, IBM, IMS, MVS, OS/390, RACF, REXX, z/OS, and zSeries are the trademark(s) or registered trademark(s) of International Business Machines Corporation in the United States, other countries, or both. DB2 is a registered trademark of International Business Machines Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

UNIX is the registered trademark of The Open Group in the US and other countries.

All other trademarks belong to their respective companies.

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 UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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 15
  Related publications .................................................. 15
  Conventions ......................................................... 16
  Syntax diagrams ..................................................... 17
  Summary of changes .................................................. 19

Chapter 1 Operational considerations and installation information 25
  Operating environment ............................................. 26
    DB2 support .......................................................... 26
    System requirements .............................................. 26
    Setting the MEMLIMIT parameter ................................ 26
  Authorization needed to use Log Master .......................... 27
    Authorization verification mechanisms ......................... 27
    DB2 authority ........................................................ 28
    RACF authority ..................................................... 29
    APF authorizations for batch programs ......................... 30
    APF authorization for the online interface ..................... 30
  Data set requirements ................................................ 30
  Installation information .............................................. 31
    Installation of the High-speed Apply Engine ..................... 31
    Installation and the Repository .................................. 32
    Dynamic bind ....................................................... 33
    Log Master CLIST .................................................. 33
    Installation options ............................................... 34
    Installation options quick reference ......................... 35
    Installation option descriptions ................................ 42
    DYNALOC installation option ..................................... 85
  Sample files and other information ............................... 87
  DB2 catalog migration considerations ............................ 89
  Improving Log Master performance ................................ 90
  Solution integration .................................................. 91

Chapter 2 Building and running Log Master jobs 93
  Building Log Master jobs ........................................... 94
  Specifying the JOB statement ..................................... 96
  Specifying the EXEC statement ................................... 97
  Specifying the STEPLIB DD or JOBLIB DD statements .......... 100
  Specifying the DD statements ..................................... 100
  Defining sort work data sets ...................................... 102
Chapter 3 Log Master for DB2 syntax

Using the syntax diagrams .......................... 114
Log Master high-level syntax .......................... 115
OPTION statement .................................... 119
 Overtime mode ..................................... 135
LOGOPTS statement .................................. 137
STOREOPTS statement ................................ 139
LOBOPTS statement .................................. 144
XMLOPTS statement .................................. 147
SORTOPTS statement ................................ 150
Sort file size parameters ............................ 154
WORKID statement ................................... 156
IMAGECOPY statement ............................... 157
 Image copy parameters ............................... 159
INPUT statement ........................................ 161
 DB2 log input definition ............................... 162
LOGSCAN statement .................................. 164
 Ongoing log scans ................................... 179
LOGSCAN report definition ......................... 181
 ORDER BY definition ................................. 194
SYSOUT definition .................................... 197
LOGSCAN logical log output definition .............. 200
LOGSCAN SQL file definition ....................... 208
 SQL type/output definition ......................... 214
 Considerations for output files and SQL .......... 225
 Considerations for output files and Unicode ..... 229
LOGSCAN load file definition ....................... 231
 Layout of records in load files ...................... 249
 Additional Informational Columns in Load Files ... 250
LOGSCAN DDL file definition ....................... 255
LOGSCAN output definition ......................... 260
 Allocation parameters ............................... 262
LOGSCAN column include/exclude definition ....... 267
LOGSCAN scan range definition ..................... 269
 Range definition ...................................... 272
LOGSCAN search condition definition .............. 277
 Filter predicate ....................................... 279
Selectable field definition ........................... 285
<table>
<thead>
<tr>
<th>Catalog activity definition</th>
<th>292</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog object definition</td>
<td>295</td>
</tr>
<tr>
<td>Synonyms</td>
<td>298</td>
</tr>
<tr>
<td>DB2 command definition</td>
<td>299</td>
</tr>
<tr>
<td>LOGMARK statement</td>
<td>305</td>
</tr>
<tr>
<td>EXECSQL statement</td>
<td>306</td>
</tr>
<tr>
<td>DROPRECOVERY statement</td>
<td>308</td>
</tr>
<tr>
<td>Recover parameters</td>
<td>317</td>
</tr>
<tr>
<td>Temporary objects definition</td>
<td>320</td>
</tr>
<tr>
<td>Report definition</td>
<td>322</td>
</tr>
<tr>
<td>Symbolic substitutions</td>
<td>323</td>
</tr>
<tr>
<td>Log Master nonSYSIN syntax</td>
<td>326</td>
</tr>
<tr>
<td>Old objects data set syntax</td>
<td>326</td>
</tr>
<tr>
<td>SQLCODES data set syntax</td>
<td>328</td>
</tr>
<tr>
<td>SQLXLAT data set syntax</td>
<td>328</td>
</tr>
</tbody>
</table>

**Chapter 4 Logical log files**

<table>
<thead>
<tr>
<th>Logical log components</th>
<th>331</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical log control file</td>
<td>332</td>
</tr>
<tr>
<td>Type record (XTYP)</td>
<td>333</td>
</tr>
<tr>
<td>Control file ID record (DCNT)</td>
<td>335</td>
</tr>
<tr>
<td>Data sharing member record (DDSM)</td>
<td>336</td>
</tr>
<tr>
<td>Work ID record (DWRK)</td>
<td>336</td>
</tr>
<tr>
<td>Work ID description record (DWKD)</td>
<td>337</td>
</tr>
<tr>
<td>Logical log data set record (DLDS)</td>
<td>338</td>
</tr>
<tr>
<td>Logical log description record (DLUS)</td>
<td>339</td>
</tr>
<tr>
<td>Log information record (DLGI)</td>
<td>340</td>
</tr>
<tr>
<td>Log information end record (DLGE)</td>
<td>341</td>
</tr>
<tr>
<td>Open transaction record (DOTR)</td>
<td>342</td>
</tr>
<tr>
<td>Current path record (XPTH)</td>
<td>343</td>
</tr>
<tr>
<td>SysStrings base record (DSS1)</td>
<td>343</td>
</tr>
<tr>
<td>SysStrings transtable record (DSS2)</td>
<td>344</td>
</tr>
<tr>
<td>Data change header description record (XHDF)</td>
<td>345</td>
</tr>
<tr>
<td>Table space information record (DTSI)</td>
<td>346</td>
</tr>
<tr>
<td>Table information record (DTBI)</td>
<td>347</td>
</tr>
<tr>
<td>Table byte count record (DTBB)</td>
<td>349</td>
</tr>
<tr>
<td>Table record count record (DTBC)</td>
<td>350</td>
</tr>
<tr>
<td>Table synonym and alias record (DTOB)</td>
<td>351</td>
</tr>
<tr>
<td>DB2 column information record (DCLI)</td>
<td>352</td>
</tr>
<tr>
<td>DB2 column description record part two (DCL2)</td>
<td>354</td>
</tr>
<tr>
<td>Logical log column information record (DLCI)</td>
<td>355</td>
</tr>
<tr>
<td>LOB column description record (DLOB)</td>
<td>357</td>
</tr>
<tr>
<td>XML column description record (DXML)</td>
<td>359</td>
</tr>
<tr>
<td>XMLSTRING data set name record (DXSF)</td>
<td>361</td>
</tr>
<tr>
<td>LOB/XML cluster data set record (CLUS)</td>
<td>361</td>
</tr>
<tr>
<td>Index information record (DIXI)</td>
<td>362</td>
</tr>
<tr>
<td>Index column description record (DICL)</td>
<td>364</td>
</tr>
<tr>
<td>Clustering index information record (DCXI)</td>
<td>364</td>
</tr>
<tr>
<td>Clustering index column description record (DCCL)</td>
<td>365</td>
</tr>
</tbody>
</table>
Figures

Sample POF with supported default values ............................................ 76
JCL for a simple Log Master for DB2 job .............................................. 95
JCL with SYSIN syntax in a separate data set .......................................... 107
Log Master high-level syntax diagram ..................................................... 116
OPTION statement syntax diagram (part 1 of 2) ..................................... 119
LOGOPTS statement syntax diagram ....................................................... 137
STOREOPTS statement syntax diagram .................................................. 140
LOBOPTS statement syntax diagram ...................................................... 145
XMLOPTS statement syntax diagram ...................................................... 148
SORTOPTS statement syntax diagram .................................................... 151
Sort file size parameters syntax diagram ................................................ 154
WORKID statement syntax diagram ....................................................... 156
IMAGECOPY statement syntax diagram .................................................. 158
Image copy parameters syntax diagram ................................................ 159
INPUT statement syntax diagram ........................................................... 161
DB2 log input definition syntax diagram .................................................. 162
LOGSCAN statement syntax diagram (part 1 of 2) ................................... 165
Report definition syntax diagram (part 1 of 2) ........................................ 181
ORDER BY definition syntax diagram ..................................................... 194
SYSOUT definition syntax diagram ....................................................... 198
Logical log definition syntax diagram ..................................................... 200
SQL file definition syntax diagram ........................................................ 208
SQL type/output definition syntax diagram .......................................... 214
Load file definition syntax diagram (part 1 of 2) ..................................... 231
Load file record layout for different UPDATES values ............................ 250
DDL file definition syntax diagram ........................................................ 255
Output definition syntax diagram ........................................................... 260
Allocation parameters syntax diagram .................................................... 262
Column include/exclude definition syntax diagram .................................. 267
Scan range definition syntax diagram ..................................................... 269
Range definition syntax diagram ........................................................... 273
Search condition definition syntax diagram ......................................... 278
Filter predicate syntax diagram .............................................................. 279
Selectable field definition syntax diagram ............................................. 286
Catalog activity definition syntax diagram .......................................... 293
Catalog object definition syntax diagram ............................................. 295
DB2 Command definition syntax diagram .............................................. 299
LOGMARK statement syntax diagram .................................................. 305
EXECSQL statement syntax diagram ..................................................... 306
DROPRECOVERY statement syntax diagram ........................................ 308
Recover parameters syntax diagram .................................................. 318
Temporary objects definition syntax diagram ................................. 320
DROPRECOVERY report syntax diagram ........................................... 322
Old objects definition syntax diagram ........................................... 327
SQLCODES syntax diagram ............................................................. 328
SQLXLAT syntax diagram ............................................................... 328
Logical log data record: basic layout ................................................. 377
Segmented logical log records ......................................................... 377
Dependent RBA value for ongoing log scans ..................................... 379
Repository tables and their relationships ......................................... 384
Tables

Log Master for DB2 installation options quick reference ........................................... 35
Installation options for data set allocation ................................................................. 52
DB2 catalog table spaces and recommended data capture settings for tables ............. 65
DYNALOC parameters ................................................................................................. 86
Report template codes ................................................................................................. 88
Data sets that are allocated in JCL .............................................................................. 100
SORTOPTS keywords for sort work data sets .......................................................... 102
Computing the required number of sorts for a job ..................................................... 105
Requirements for running Log Master jobs ............................................................... 109
Log Master output handling at termination ............................................................... 111
Criteria for either sort or merge of DB2 log files ....................................................... 163
Product action determined by REPOS UPDATE value ............................................... 169
Example of possible errors with USE RUNSEQ or RERUN RUNSEQ ..................... 178
INCLUDE ROLLBACK values by report type ............................................................. 193
Additional actions for untranslatable Unicode characters ........................................ 230
Valid filter predicates, GENERATE EMPTY FILES keyword .................................... 247
Additional informational columns: URID FIELDS ALL ........................................... 251
Additional informational columns: URID FIELDS URID .......................................... 252
Additional informational columns: URID FIELDS NONE .......................................... 253
Additional table format: URID TABLE YES ............................................................... 253
Required and default LRECL / RECFM values by output type ................................ 263
Numeric operators in filters ....................................................................................... 280
Wildcard characters for comparisons ........................................................................ 282
Synonyms .................................................................................................................. 298
Data set type and description of RECOVER2 data set name ..................................... 313
Examples for RECOVER and RECOVER2 output file names ................................... 314
Symbolic substitutions ............................................................................................... 323
Examples of how Log Master generates and truncates data set names ...................... 325
Type record (XTYP) .................................................................................................... 333
Control file id record (DCNT) .................................................................................... 335
Data sharing member record (DDSM) ....................................................................... 336
Work ID record (DWRK) ........................................................................................... 336
Work ID description record (DWKD) ........................................................................ 337
Logical log data set record (DLDS) .......................................................................... 338
Logical log description record (DLUS) ..................................................................... 339
Log information record (DLGI) .................................................................................. 341
Log information end record (DLGE) ....................................................................... 341
Open transaction record (DOTR) .............................................................................. 342
Current path record (XPTH) ..................................................................................... 343
SysStrings base record (DSS1) ................................................................................ 343
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINEDATA field: Catalog activity line type</td>
<td>344</td>
</tr>
<tr>
<td>SysStrings base record (DSS2)</td>
<td>345</td>
</tr>
<tr>
<td>Data change header description record (XHDF)</td>
<td>346</td>
</tr>
<tr>
<td>Table space description record (DTSI)</td>
<td>347</td>
</tr>
<tr>
<td>Table information record (DTBI)</td>
<td>349</td>
</tr>
<tr>
<td>Table byte count record (DTBB)</td>
<td>350</td>
</tr>
<tr>
<td>Table record count record (DTBC)</td>
<td>351</td>
</tr>
<tr>
<td>Table synonym and alias record (DTOB)</td>
<td>354</td>
</tr>
<tr>
<td>DB2 column information record (DCILI)</td>
<td>355</td>
</tr>
<tr>
<td>DB2 column description record part two (DCL2)</td>
<td>356</td>
</tr>
<tr>
<td>Logical log column information record (DLCI)</td>
<td>357</td>
</tr>
<tr>
<td>LOB column description record (DLOB)</td>
<td>358</td>
</tr>
<tr>
<td>XML column description record (DXML)</td>
<td>359</td>
</tr>
<tr>
<td>XMLSTRING data set name record (DXSF)</td>
<td>360</td>
</tr>
<tr>
<td>LOB/XML cluster data set record (CLUS)</td>
<td>361</td>
</tr>
<tr>
<td>Index description record (DIXI)</td>
<td>362</td>
</tr>
<tr>
<td>Index column description record (DICL)</td>
<td>363</td>
</tr>
<tr>
<td>Clustering index information record (DCXI)</td>
<td>364</td>
</tr>
<tr>
<td>Clustering index column description record (DCCL)</td>
<td>365</td>
</tr>
<tr>
<td>Filter ID record (DFID)</td>
<td>366</td>
</tr>
<tr>
<td>Filter anchor record (DFIL)</td>
<td>367</td>
</tr>
<tr>
<td>Filter description record (DFDS)</td>
<td>368</td>
</tr>
<tr>
<td>Filter line record (DFLN)</td>
<td>369</td>
</tr>
<tr>
<td>Filter line data record (DLIN)</td>
<td>371</td>
</tr>
<tr>
<td>Data change record (LLDF)</td>
<td>372</td>
</tr>
<tr>
<td>Repository table spaces and tables</td>
<td>373</td>
</tr>
<tr>
<td>Automated drop recovery file table (ALPADRF)</td>
<td>374</td>
</tr>
<tr>
<td>Column list table (ALPCOLS)</td>
<td>375</td>
</tr>
<tr>
<td>DDL file pointer table (ALPDDLFL)</td>
<td>376</td>
</tr>
<tr>
<td>Dropped objects table (ALPDRFLN)</td>
<td>377</td>
</tr>
<tr>
<td>Filter header table (ALPFLTH)</td>
<td>378</td>
</tr>
<tr>
<td>Filter line table (ALPFLIN)</td>
<td>379</td>
</tr>
<tr>
<td>LINEDATA field: Input logical log file type</td>
<td>380</td>
</tr>
<tr>
<td>Filter pointer table (ALPFLTP)</td>
<td>381</td>
</tr>
<tr>
<td>Load file pointer table (ALPLODF)</td>
<td>382</td>
</tr>
<tr>
<td>Logical log file pointer table (ALPLOGF)</td>
<td>383</td>
</tr>
<tr>
<td>Mark table (ALPMARK)</td>
<td>384</td>
</tr>
<tr>
<td>Old objects table (ALPOLDO)</td>
<td>385</td>
</tr>
<tr>
<td>Open unit of recovery table (ALPURID)</td>
<td>386</td>
</tr>
<tr>
<td>Report file pointer table (ALPRPTFL)</td>
<td>387</td>
</tr>
<tr>
<td>Report template table (ALPRTMPL)</td>
<td>388</td>
</tr>
<tr>
<td>Security exception history table (ALPSECH)</td>
<td>389</td>
</tr>
<tr>
<td>SQL code handler table (ALPSQLC)</td>
<td>390</td>
</tr>
<tr>
<td>SQL file pointer table (ALPSQLF)</td>
<td>391</td>
</tr>
<tr>
<td>SQL index table (ALPIXSQ)</td>
<td>392</td>
</tr>
<tr>
<td>SQL name translation table (ALPSQLX)</td>
<td>393</td>
</tr>
<tr>
<td>Work ID header table (ALPWHDR)</td>
<td>394</td>
</tr>
<tr>
<td>Work ID history table (ALPWHS)</td>
<td>395</td>
</tr>
<tr>
<td>Work ID input file definitions table (ALPIFIL)</td>
<td>396</td>
</tr>
<tr>
<td>LINEDATA field: Input logical log file type</td>
<td>397</td>
</tr>
</tbody>
</table>
LINEDATA field: Input DB2 log file type ........................................... 415
LINEDATA field: Input (uncataloged) image copy file type .................... 416
Work ID log scan history table (ALPWLSH) ...................................... 416
Work ID output file definitions table (ALPOFIL) ................................. 418
LINEDATA2 field: Logical log file type ........................................... 419
LINEDATA2 field: SQL file type ..................................................... 420
LINEDATA2 field: Report file type .................................................. 422
LINEDATA2 field: Load file type .................................................... 424
LINEDATA2 field: DDL file type .................................................... 426
LINEDATA2 field: Drop recovery outcopy file type .............................. 426
LINEDATA2 field: Drop recovery control parameters type ..................... 427
LINEDATA2 field: LOB VSAM file type ........................................... 428
LINEDATA2 field: XML VSAM file type .......................................... 429
OUTPUTDATA field: Data set type ................................................. 430
OUTPUTDATA field: SYSOUT type ................................................. 431
Work ID step table (ALPWSTP) ..................................................... 432
STEPDATA field: Log mark step type ............................................. 433
STEPDATA field: Log scan step type .............................................. 433
STEPDATA field: Execute SQL step type ....................................... 435
STEPDATA field: High-speed Apply JCL generation step type ............... 436
STEPDATA field: Repository maintenance step type ........................... 438
About this book

This book contains detailed information about the Log Master for DB2® product from BMC Software. The book is intended for database administrators (DBAs), system administrators, and systems programmers.

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

- IBM® DB2® Universal Database for mainframe targets
- the IBM operating system that supports your version of DB2 (such as z/OS® and its successors)
- environmental software and utilities related to the operating system, including job control language (JCL), and the Interactive System Productivity Facility (ISPF)

For example, you should know how to create and submit JCL for a batch job and how to respond to ISPF panels.

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.

The software also offers online Help. To access Help, press F1 within any product or click the Help button in graphical user interfaces (GUls).

Related publications

From the BMC Support Central website (http://www.bmc.com/support), you can use either of the following methods to access related publications that support your product or solution:
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:

  \texttt{testsys/instancelfileName}

- The symbol => connects items in a menu sequence. For example, \texttt{Actions => Create Test} instructs you to choose the \texttt{Create Test} command from the \texttt{Actions} menu.
Syntax diagrams

The following figure shows the standard format for syntax diagrams:

```
statement begins  command  statement continued on next line
```

```
statement continues  required item  optional item
```

```
command
```

```
required choice  optional choice
```

```
variableValue  variableValue
```

```
multiple choices
```

```
statement ends
```

The following example illustrates the syntax for a 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.)

```
DELETE(tableName) FROM alias
```

```
WHERE CURRENT OF cursorName;
```

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 the default value when a choice is available.

- If a diagram shows punctuation marks, parentheses, or similar symbols, you must enter them as part of the syntax. Asterisks are exceptions. An asterisk in a diagram indicates a reference note.

- In general, operating system commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum portion of the operating system 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, the first letter of a word after the first word is capitalized, for example fileName, or dataSetName).

- The text descriptions that follow a syntax diagram describe any restrictions on the number of times you can repeat any syntax within a recursive (left-pointing) arrow.

- A syntax keyword enclosed in a bold rectangular box indicates that a corresponding diagram appears either on another page or later within the same diagram. The page number above and to the right of the bold rectangular box lists the page number of the referenced diagram.
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 June 2013

This release of Log Master for DB2 includes the following product enhancements and changes:

**DB2 support**

Log Master supports the following versions of DB2:

- Version 10
- Version 9

DB2 Version 8 is no longer supported. Most references to DB2 Version 8 have been removed from this book.

**Logical log changes**

Several tables in Chapter 4, “Logical log files,” show changes that have been made to the logical log to support extended RBAs and LRSNs, as well as extended timestamps.

However, Log Master allows for processing of earlier versions (back to version 7.3.00) of the logical log with normalization performed as needed to work with the new format.

The following records have been removed from this book because they are no longer used:

- Table space description record (DTSP)
- Table space description record part two (DTS2)
- Table description record (DTBL)
- Table RBA record (DTBR)
- DB2 column description record (DCOL)
- Logical log column description record (DLCO)
- Index description record (DIDX)
- Clustering index description record (DCDX)
Summary of changes

Reuse of Log Master repository tables

Log Master now supports the ability to point to existing Log Master tables when installing a new version of the product in order to avoid unload or load of the tables. ALPVERSION has been added to the following tables:

- ALPMRK (page 397)
- ALPWHDR (page 409)
- ALPURID (page 399)

This field is empty if the records were created prior to Log Master version 11.1.

Also, see the updated information in “Installation and the Repository” on page 32

Repository notes and changes

Note the following information and changes relating to the Log Master repository (Chapter 5, “Log Master for DB2 Repository”):

- If you are reusing a version 10.1 or older repository, you should complete any OVERTIME LOGSCANS for Table Objects that have been dropped before upgrading to version 11.1. After you upgrade to version 11.1, you must refresh the Table Object (not Dictionary Objects) rows in the Old Object Table. Version 11.1 OVERTIME LOGSCANS will ignore version 10.1 or older Table Objects rows in the Old Object Table.

- In the following repository tables, the specified fields have changed to format VARCHAR(10) to support the extended RBAs/LRSNs:

<table>
<thead>
<tr>
<th>Table name</th>
<th>Column name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPLOGF</td>
<td>STARTRBA</td>
</tr>
<tr>
<td></td>
<td>ENDRBA</td>
</tr>
<tr>
<td>ALPMARK</td>
<td>MARKRBA</td>
</tr>
<tr>
<td>ALPOLDO</td>
<td>BEGINRBA</td>
</tr>
<tr>
<td></td>
<td>ENDRBA</td>
</tr>
<tr>
<td>ALPURID</td>
<td>URID</td>
</tr>
<tr>
<td></td>
<td>URIDLRSN</td>
</tr>
<tr>
<td>ALPSQLF</td>
<td>REDORBA</td>
</tr>
<tr>
<td>ALPWHIS</td>
<td>START_RBA</td>
</tr>
<tr>
<td></td>
<td>END_RBA</td>
</tr>
<tr>
<td>ALPWLSH</td>
<td>ACTUAL_FROM_RBA</td>
</tr>
<tr>
<td></td>
<td>ACTUAL_TO_RBA</td>
</tr>
<tr>
<td></td>
<td>REDORBA</td>
</tr>
</tbody>
</table>
- References to record type H have been removed from the ALPOLDO table description (page 398).

- References to DB2 Version 8.1 have been removed from the tables.

**LOGTAPES for DASD log files and new TAPES installation option**

LOGTAPES is ignored if DB2 log files to be read are DASD files, or the number of actual tape units is not greater than the LOGTAPES value in a subsystem with mixed tapes/DASD log files. For this new behavior, the TAPES installation option (page 74) is added in order to find what devices are tapes.

**Support for automated drop recovery of clone tables**

If a table space or table to be recovered has a clone table, Log Master now generates syntax that instructs RECOVER PLUS to use image copies to recover the clone table (page 313, page 314, and page 318).

**Data set names, symbolic variables, and the new DSNSYMPRE installation option**

The DSNSYMPRE installation option (page 51) allows you to control if Log Master forces the prefix for &TABNAME and &WORKID in input and output data set names. With DSNSYMPRE=YES, the default, Log Master always adds the prefix. With DSNSYMPRE=NO, Log Master adds the prefix only if the first character of the node is not an alphabetic or a national character.

This enhancement

- removes the requirement for &TABOWN for SEPARATE DATASETS (page 247)
- updates the actions that Log Master performs to create a valid data set name (page 324)

**Support user-selection of image copy resources**

Log Master adds the following installation options:

- LOCCPSEL=(FC,LP,LB) (page 85)
- REMCPSEL=(RP,RB,FC) (page 85)

These installation options allow you to select the order in which Log Master uses image copy resources for both completion processing and compression dictionary access for data decompression at local and remote sites.

To override the value of these installation options at runtime, Log Master adds the OPTION RESOURCE SELECTION COPIES syntax (page 120, page 134).
Provide a USELOGS override

Log Master adds LOGS to the OPTION command, RESOURCE SELECTION syntax to enable you to override the USELOGS installation option (page 62, page 120, and page 134).

Support for graphic data in predicate values

Log Master supports UX for Unicode graphic data and GX for EBCDIC graphic data in predicate values, as well as a string for Unicode graphic data. See Table 23 on page 282.

Support for versioned rows with inline LOBs

Log Master now supports versioned rows with inline LOBs. Log Master no longer issues error message BMC397064E (removed).

Support for REUSABLE striped data sets

Log Master now allows use of REUSABLE striped data sets. Restrictions no longer apply. Log Master no longer issues error message BMC097197 (removed).

BMCXCOPY analysis for automated drop recovery

Log Master now supports an automated drop recovery of a table space using the following copies registered in BMCXCOPY:

- encrypted copies
- Snapshot copies
- standard copies

Cabinet copies are not supported.

Mass delete and exchange records in load outputs

Log Master now includes the mass delete and exchange records in load outputs. For more information, see “Layout of records in load files” on page 249.

Quiesce records for COPY YES indexes

When you specify QUIESCE with the LOGSCAN statement (page 173) or in the LOGSCAN report definition (page 185), Log Master now inserts index quiesce records into SYSIBM.SYSCOPY for COPY YES indexes along with the table space. Log Master records the correct quiet point registration so all recoverable objects have the proper QUIESCE entry in SYSCOPY.
Enhanced messages for DB2 log processing

Log Master adds messages that provide the information necessary to determine the progress of log processing. See messages BMC397070 and BMC397071 in the BMC Doc Center.

Support for optional column names in SQL INSERT

Log Master now provides an option to omit the column name list clause for INSERT statements in SQL output. INCLUDE or EXCLUDE INSERT COLUMN NAMES syntax has been added to the LOGSCAN SQL file definition (page 208, page 212). Additional columns have also been added to the table in “LINEDATA2 field: SQL file type” on page 420.

Enhanced report templates

Log Master now provides sample report templates for Summary, Audit, and Detail reports with control breaks on Table Name and Create Timestamp (page 87, page 183).

XML installation options

Because the XML installation options are duplicates of the LOB installation options, Log Master now supports only the LOB installation options for use with both LOB and XML VSAM files. Log Master no longer supports the following installation options, and they have been removed from this book:

- XMLALLOC
- XMLDATA
- XMLDUPD
- XMLLIMIT
- XLMGMT
- XMLSPACE
- XMLSTOR
- XMLVOLS

**NOTE**

Continue to use the following syntax:

- Use the LOBOPTS syntax to override installation option settings for LOB data.
- Use the XMLOPTS syntax to override installation option settings for XML data.

Documentation changes

This release includes the following documentation changes:
Summary of changes

- p Secret from Computer Associates from the product documentation.

- Changes the LUWINSTANCENO definition from CHAR(6) to BIN(6) in the Data change record (LLDF) table (Table 64 on page 371).

- Adds XMLINCLUDED to the Table information record (DTBI) table (Table 44 on page 347) and updates the Total Length.

- Corrects information about executing Log Master from a panel on page 33.

- 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 now located in the following books:
  - *Installation System User Guide*
  - *BMC Products and Solutions for DB2 Configuration Guide*
Chapter 1 Operational considerations and installation information

This chapter contains the following topics:

Operating environment ......................................................... 26
  DB2 support ................................................................. 26
  System requirements ...................................................... 26
  Setting the MEMLIMIT parameter ...................................... 26
Authorization needed to use Log Master .................................. 27
  Authorization verification mechanisms ............................... 27
  DB2 authority .............................................................. 28
  RACF authority ............................................................ 29
  APF authorizations for batch programs .............................. 30
  APF authorization for the online interface ......................... 30
Data set requirements .......................................................... 30
Installation information ....................................................... 31
  Installation of the High-speed Apply Engine ...................... 31
  Installation and the Repository ...................................... 32
  Dynamic bind .............................................................. 33
  Log Master CLIST ......................................................... 33
  Installation options ....................................................... 34
  Installation options quick reference ................................. 35
  Installation option descriptions ..................................... 42
  DYNALOC installation option ......................................... 85
  Sample files and other information ................................ 87
DB2 catalog migration considerations .................................... 89
Improving Log Master performance ....................................... 90
Solution integration ........................................................... 91
Operating environment

Log Master requires the operating environment described in this section.

DB2 support

This version of Log Master supports IBM-supported versions of DB2.

System requirements

This version of Log Master supports IBM-supported versions of z/OS that are active on all systems in the sysplex.

For data migration, Log Master includes versions of the High-speed Apply Engine that run on distributed systems (UNIX and Microsoft Windows) against target databases on either IBM’s DB2 Universal Database (UDB) or Oracle.

To use the Log Master online interface, you need ISPF version 3.3 or later.

Setting the MEMLIMIT parameter

The following products and components require above-the-bar memory and might abend if sufficient memory is not available:

- ALTER
- BMCSORT
- CATALOG MANAGER
- CHANGE MANAGER
- CHECK PLUS
- COPY PLUS
- DASD MANAGER PLUS
- High-speed Apply Engine
- LOADPLUS
- Log Master
- RECOVER PLUS
Authorization needed to use Log Master

- RECOVERY MANAGER
- REORG PLUS
- UNLOAD PLUS

In z/OS versions before 1.10, the default value for the System Management Facility (SMF) MEMLIMIT parameter is 0; a value of 0 means that no address space can use virtual storage above the bar. In z/OS version 1.10 and later, the default value is 2 GB.

For most jobs, BMC recommends a value of at least 1 GB for the MEMLIMIT parameter. However, if you are operating on LOB or XML data, BMC recommends a value of at least 32 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.

Authorization needed to use Log Master

To use Log Master, you need authorization within DB2 and through your system security package, such as the IBM product Resource Access Control Facility (RACF). The authorization must be sufficient to access DB2 resources and perform the tasks accomplished during processing. The following sections provide more information about the required authorizations.

Authorization verification mechanisms

If the DB2 DSNX@XAC authorization exit is available for your system, Log Master uses this exit to verify authorization for external access. The exit is available from the following sources:

- IBM provides a sample exit with DB2 for the IBM Resource Access Control Facility (RACF®) component.
- CA Technologies provides the DSNX@XAC exit with 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, Log Master uses the standard DB2 method to check security.

**DB2 authority**

To ensure that Log Master runs correctly in your environment, you must have the following DB2 authorizations:

- EXECUTE privilege on the Log Master batch and online plans
- DISPLAY system privilege
- authority to perform quiesce at log mark

Before a Log Master job can use this feature of the product, the user ID of the job must also have one of the following DB2 authorities or privileges:

- DBADM, DBCTL, or DBMAINT authority for the databases
- SYSCTRL or SYSADM authority
- IMAGCOPY privilege for the databases
- authorizations to execute SQL

Log Master uses the High-speed Apply Engine to execute generated SQL statements. The required authorizations are listed in this section. For more information about controlling access, see the installation section in the *High-speed Apply Engine Reference Manual*.

Before a Log Master job can execute SQL, the user ID of the job (or the user ID specified in either the EXECSQL statement or the BINDOWN installation option) must have the following DB2 privileges:

- EXECUTE privilege for the plan that the High-speed Apply Engine uses to access its own restart tables and the catalog (normally provided during installation)
- EXECUTE privilege for the High-speed Apply restart package (normally provided during installation)
- INSERT, UPDATE and DELETE privileges on the target tables
- appropriate privileges to bind or administer plans, packages, and collections
The High-speed Apply Engine provides several ways to grant these privileges. Some techniques avoid granting bind privileges to the user ID that runs Log Master. For more information, see the *High-speed Apply Engine Reference Manual*.

**RACF authority**

Log Master does not run as part of the DB2 subsystem To use the product, you must have system authority similar to that of DB2.

Log Master reads data from certain underlying DB2 data sets such as table spaces, active and archive logs, or the bootstrap data set (BSDS). If the underlying data sets are protected by RACF or a similar system security package, the user ID of the Log Master batch job must have authority to access all of the underlying data sets that the job requires.

To avoid granting authority for each required data set to the user ID of each Log Master batch job, use the OPNDB2ID installation option. This option works when all of the following conditions exist:

- your environment uses RACF

  The OPNDB2ID installation option does not operate in other security environments.

- you install the product with the OPNDB2ID installation option set to YES

  When OPNDB2ID is set to YES, Log Master uses the RACF ID of DB2 to open the DB2 data sets.

- you explicitly associate a user ID with the DB2 address space

  — For OPNDB2ID to work correctly, you must explicitly associate a user ID with DB2 regardless of whether you specify DB2 as a privileged or trusted task in the RACF started procedures table (ICHRIN03).

  — To ensure that OPNDB2ID works correctly in a data sharing environment, the RACF IDs of the DBM1 address spaces within all DB2 subsystems within the data sharing group must be the same. The authorizations for the bootstrap and log data sets must also be the same.
APF authorizations for batch programs

Log Master batch programs use operating system services that require APF authorization. Accordingly, Log Master must reside in APF-authorized libraries. Any libraries that you reference in the STEPLIB DD statements must also be APF authorized.

APF authorization for the online interface

You can run the Log Master online interface with or without APF authorization. The APFONLIN installation option determines whether Log Master expects to have proper APF authorization (for more information, see “APFONLIN=YES” on page 76).

- Without authorization, an online user must enter the name and location of the BSDS on the Product Options panel. The online interface does not run as an authorized TSO program.

- With proper authorization, Log Master can obtain the name of the BSDS dynamically from DB2. The online interface runs as an authorized TSO program.

The TSO program name for the product is SCCAUTH. You must place this name in the operating system’s SYS1.PARMLIB data set in the authorized command table. The command table is a member of SYS1.PARMLIB named IKJTSOxx. The suffix xx is assigned during installation. The TSO command table contains several different lists. Place SCCAUTH in the authorized program list (which is specified as AUTHPGM NAMES).

**NOTE**

Perform this procedure on all operating system images where you expect the product to run as an authorized TSO program.

Data set requirements

The following data set requirements apply to Log Master:

- All output data sets created by Log Master must be cataloged.

- Input logical log and SQL data sets must be cataloged.
BMC recommends that you avoid using a VSAM buffering tool to perform I/O caching on the following data sets. If any of these data sets are buffered, Log Master can terminate abnormally, encounter errors, or generate inaccurate data.

- active log file data sets
- archive log file data sets
- bootstrap data set (BSDS)

### Installation information

To install Log Master, use the Installation System from BMC Software. The process does not require any modifications to DB2, nor does it require an initial program load (IPL) of your system. However, it does require an APF-authorized library. For more information, see “APF authorizations for batch programs” on page 30.

This section summarizes considerations that you should be aware of when you install Log Master. For more information, see the *Installation System User Guide* or the *BMC Products and Solutions for DB2 Configuration Guide*.

### Installation of the High-speed Apply Engine

High-speed Apply Engine is automatically installed when you install Log Master and the Recovery Management for DB2 solution.

You can also install High-speed Apply separately by selecting it from the list in the Installation System. The High-speed Apply Engine honors passwords for Log Master, the Recovery Management for DB2 solution, or itself.

**NOTE**

If installed separately, High-speed Apply cannot process logical log input that contains data definition language information called *DDL objects*. To activate this capability, contact BMC Customer Support.

For data migration, Log Master for DB2 includes versions of the High-speed Apply Engine that run on distributed systems (UNIX or Windows) against target databases under either IBM’s DB2 Universal Database (UDB) or Oracle. To install these versions, use the CD-ROM that is distributed with Log Master. For more information, see the *High-speed Apply Engine Reference Manual*.
Installation and the Repository

Log Master uses a Repository to store information as you run the online interface or run batch jobs. The Repository contains several DB2 tables. When you install a new version of Log Master, the Installation System prompts with the following choices:

- whether to create new Repository tables, or continue using existing tables from the version that is already installed

The Installation System displays this prompt only when

- the format of the Repository tables has not changed between the previous version and the current version of Log Master
- the previous version of Log Master is version 10.1 or later

If Log Master version 11.1 is installed with repository reuse and the previous version is version 10.1, in order to continue using the online menu in version 10.1, you must apply a PTF with support repository reuse in version 10.1.

- whether to migrate information from the previous version’s Repository, or insert only new information into the Repository after installation is complete

The Installation System always displays this prompt.

- If you choose to migrate information from the previous version’s Repository, the Installation System generates the $C68ALP job. The Installation system runs this job to migrate information after it creates the new Repository. If you need to migrate your Repository information after Log Master installation is complete, run the Installation System again and select the choice to migrate information.

- If you choose to insert only new information, the Installation System generates an empty Repository.

The Installation System automatically generates the repository tables to accommodate the maximum object name length that your version of DB2 supports.

The Repository

The Installation System automatically generates the Repository tables in the correct format for the current DB2 subsystem. After you install Log Master, run a DB2 REORG utility to remove the advisory reorg (AREO) status from several repository table spaces.

If you migrate a DB2 subsystem from compatibility mode of a DB2 version to new-function mode without reinstalling Log Master, complete the following steps after the DB2 catalog is in new-function mode:
- Run the job stored in the ALPURPTB member of the data set where you stored the JCL jobs that were generated during installation. (For example, HLQ.JCL, where HLQ is the high-level qualifier that you assign during installation.)
- Run a DB2 REORG utility to remove the advisory reorg (AREO) status from several Repository table spaces.

**Dynamic bind**

Log Master uses a dynamic bind process, which is a proprietary technology of BMC Software. This capability uses packages instead of plans to optimize the DBRM bind process, and allows SQL preparation to complete during execution.

BMC recommends executing the installation verification procedure (IVP) that the Installation System generates. The IVP executes the dynamic bind process, which helps to avoid potential bind problems, including authorization problems, during later executions of the product. For more information, see the *Installation System User Guide* or the *BMC Products and Solutions for DB2 Configuration Guide*.

**Log Master CLIST**

As you install Log Master, one of the installation jobs places a CLIST named ALPISPF in the HLQ.DBCLIB library (or to a different library specified during installation).

To run the Log Master online interface, execute ALPISPF in one of the ways described in this section. In all examples, *clistLibName* is the name of the data set where ALPISPF exists and *ssid* is the identifier of the DB2 subsystem where Log Master will run.

To execute from ISPF Option 6, enter one of the following TSO commands, depending on whether you want to pass the DB2 subsystem ID (SSID):

```plaintext
EX 'clistLibName(ALPISPF)'
EX 'clistLibName(ALPISPF) 'SSID (ssid)'
```

To execute from an ISPF panel, add an option to the panel that executes ALPISPF. For example, if you use LM as the option name, enter one of the following lines in the panel definition, depending on whether you want to pass the SSID:

```plaintext
LM, 'CMD(EX ''clistLibName(ALPISPF)'' )'
LM, 'CMD(EX ''clistLibName(ALPISPF)'' 'SSID(ssid)'' )'
```
Installation options

When you install Log Master, the Installation System generates a customized installation data set that contains all of the jobs that install Log Master into your DB2 environment. One of these jobs establishes the installation option values (also known as DOPTs) for Log Master.

The $C30DOPT member of the installation data set contains a job that runs an assembler program. The program uses an options macro to establish the installation processing values for Log Master. You can tailor the installation options of the product, including changing plan names, by overriding the installation options.

When you submit the generated job that contains the installation options, it assembles and links this option macro call to create the ALP$OPTS load module. This module resides in the APF-authorized library that you designated during installation.

If you change any installation option values after Log Master has been installed, you must rerun the installation job for these changes to take effect.

If you have previously installed Log Master and want to keep the same installation option values, you can take one of the following actions:

- gather your existing option values before you start to install, and then enter the same values as you run the Installation System

- use the options migration feature of the Installation System. For more information about options migration, see the Installation System User Guide or the BMC Products and Solutions for DB2 Configuration Guide.

Log Master can also read a product options file (POF) to obtain certain default values for the Log Master online interface. For more information about this optional file, see “ALPOFDSN=” on page 75.
Installation options quick reference

Table 1 lists the installation options (also known as DOPTs) in alphabetical order, summarizes them, provides their default values, provides a brief description, and includes any SYSIN syntax that can override the option’s value.

Table 1  Log Master for DB2 installation options quick reference (part 1 of 7)

<table>
<thead>
<tr>
<th>Installation option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPOFDSN</td>
<td>none</td>
<td>specifies the location of a data set that contains the Log Master POF. Log Master can read the POF to obtain certain default values for the online interface.</td>
<td>page 75</td>
</tr>
<tr>
<td>AMSGLEV</td>
<td>0</td>
<td>determines the type of messages that Log Master issues when a job encounters a DB2 security exception as it processes either DB2 log or logical log input</td>
<td>page 46</td>
</tr>
<tr>
<td>ANPCT</td>
<td>20</td>
<td>specifies the percentage of key store memory that is allocated to the anomaly (AN) key store</td>
<td>page 54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MEMPERCENT ANOMALY</td>
<td></td>
</tr>
<tr>
<td>APFONLIN</td>
<td>YES</td>
<td>determines whether the online interface runs as an authorized TSO command</td>
<td>page 76</td>
</tr>
<tr>
<td>BCPCT</td>
<td>10</td>
<td>specifies the percentage of key store memory that is allocated to the backward completion (BC) key store</td>
<td>page 54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MEMPERCENT BACKWARDCOMPLETION</td>
<td></td>
</tr>
<tr>
<td>BINDOWN</td>
<td>none</td>
<td>specifies an authorization ID that Log Master uses to bind application plans when it executes SQL statements</td>
<td>page 43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: EXECSQL, BINDOWN</td>
<td></td>
</tr>
<tr>
<td>BINDQUALIFIER</td>
<td>ALP&lt;QUALIFIER&gt;</td>
<td>specifies a qualifier value that Log Master uses during execution to dynamically bind internal packages that are distributed with Log Master</td>
<td>page 43</td>
</tr>
<tr>
<td>BYPASSLL</td>
<td>NO</td>
<td>determines whether Log Master generates Summary reports that include or exclude the INSERT log records that result when a DB2 Load utility is run with the LOG option set to YES</td>
<td>page 68</td>
</tr>
<tr>
<td>CMPLGRNG</td>
<td>YES</td>
<td>indicates whether Log Master can use the SYSIBM.SYSLGRNX tables to reduce the amount of log that it reads during row completion processing</td>
<td>page 59</td>
</tr>
<tr>
<td>CSTFILE</td>
<td>2000</td>
<td>defines the relative cost of processing an additional data set, expressed in terms of the cost to read one page from the DB2 log</td>
<td>page 57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, FILECOST</td>
<td></td>
</tr>
<tr>
<td>CSTMOUNT</td>
<td>25000</td>
<td>defines the relative cost of a single tape mount, expressed in terms of the cost to read one page from the DB2 log</td>
<td>page 58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, FILECOST</td>
<td></td>
</tr>
<tr>
<td>CSTSEG</td>
<td>2000000000</td>
<td>defines the relative cost of obtaining information from a segmented table space, expressed in terms of the cost to read one page from the DB2 log</td>
<td>page 58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, FILECOST</td>
<td></td>
</tr>
<tr>
<td>Installation option</td>
<td>Shipped value</td>
<td>Brief description</td>
<td>Reference</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>DASDDSNS</td>
<td>10</td>
<td>determines the maximum number of recalled data sets that Log Master maintains on DASD at one time</td>
<td>page 81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, DASD DATASETS</td>
<td></td>
</tr>
<tr>
<td>DB2AUTH</td>
<td>NO</td>
<td>determines whether Log Master honors DB2 security at the table level whenever a job accesses DB2 log input</td>
<td>page 44</td>
</tr>
<tr>
<td>DB2CAT</td>
<td>NO</td>
<td>determines whether Log Master processes log records that were created as a result of updates to DB2 catalog tables</td>
<td>page 65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOGSCAN, DB2CATALOG</td>
<td></td>
</tr>
<tr>
<td>DBCS</td>
<td>NO</td>
<td>directs Log Master to use only uppercase characters in the online interface and Log Master-generated portions of reports. This installation option is provided for environments that display double-byte characters online.</td>
<td>page 77</td>
</tr>
<tr>
<td>DDLOBJ</td>
<td>NO</td>
<td>determines whether Log Master includes DDL information in all output logical log files</td>
<td>page 69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOGSCAN, INCLUDE DDL OBJECTS</td>
<td></td>
</tr>
<tr>
<td>DDPRMBR</td>
<td>YES</td>
<td>specifies the scope of the DASDDSNS value</td>
<td>page 81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, PER MEMBER</td>
<td></td>
</tr>
<tr>
<td>DICTSPC</td>
<td>50M</td>
<td>specifies a limit on the amount of memory that Log Master uses to store compression dictionaries during processing</td>
<td>page 55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, DICTIONARYSPACE</td>
<td></td>
</tr>
<tr>
<td>DSNSYMPE</td>
<td>YES</td>
<td>forces the prefix for &amp;TABNAME and &amp;WORKID in input and output data set names</td>
<td>page 51</td>
</tr>
<tr>
<td>DYNALOC</td>
<td>n/a</td>
<td>provides information for dynamically allocating SORTWK data sets</td>
<td>page 85</td>
</tr>
<tr>
<td>ERLYRCL</td>
<td>YES</td>
<td>directs Log Master to issue requests to recall archived data sets before it needs the data sets for log processing</td>
<td>page 78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, EARLY RECALL</td>
<td></td>
</tr>
<tr>
<td>EXEMODE</td>
<td>CURRENT</td>
<td>determines whether Log Master processes log records for DB2 objects that have been dropped and possibly re-created</td>
<td>page 66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, EXECUTION MODE</td>
<td></td>
</tr>
<tr>
<td>FCPCT</td>
<td>10</td>
<td>specifies the percentage of key store memory that is allocated to the forward completion (FC) key store</td>
<td>page 54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MEMPERCENT FORWARDCOMPLETION</td>
<td></td>
</tr>
<tr>
<td>FCUSE</td>
<td>YES</td>
<td>determines whether Log Master uses the forward completion (FC) key store to perform a special type of row completion processing</td>
<td>page 60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, HEURISTIC FORWARDCOMPLETION</td>
<td></td>
</tr>
<tr>
<td>FLTRMTHD</td>
<td>STATIC</td>
<td>determines how Log Master obtains DB2 catalog information (DBIDs, OBIDs, or PSIDs) for the DB2 objects that the filter names</td>
<td>page 66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, FILTER METHOD</td>
<td></td>
</tr>
<tr>
<td>GENMDEL</td>
<td>YES</td>
<td>determines the output that Log Master generates when it encounters the DB2 log records that reflect a LOAD REPLACE action</td>
<td>page 68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, GENERATE MASSDELETE</td>
<td></td>
</tr>
<tr>
<td>Installation option</td>
<td>Shipped value</td>
<td>Brief description</td>
<td>Reference</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>IMAGESRC</td>
<td>ANY</td>
<td>specifies the source that Log Master uses to perform row completion or decompression processing</td>
<td>page 56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, IMAGESOURCE</td>
<td></td>
</tr>
<tr>
<td>KEYDSNAM</td>
<td>none</td>
<td>specifies the name of the key data set that is used for encrypted image copies</td>
<td>page 46</td>
</tr>
<tr>
<td>KSALLOCU</td>
<td>CYLS</td>
<td>specifies the storage unit that Log Master uses when it allocates VSAM data sets as overflow storage for key store work areas</td>
<td>page 50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, CYLINDERS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRACKS</td>
</tr>
<tr>
<td>KSCLUST</td>
<td>&amp;&amp;SYSUID..CLUSTER</td>
<td>specifies a prefix name that Log Master uses to derive the cluster name for a VSAM data set</td>
<td>page 50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this data set as overflow storage for key store work areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, CLUSTER</td>
<td></td>
</tr>
<tr>
<td>KSDACLS</td>
<td>none</td>
<td>specifies a valid data class within the IBM product Data Facility Storage Management System (DFSMS)</td>
<td>page 52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this data class to allocate VSAM data sets for use as overflow storage for key store work areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, DATACLAS</td>
<td></td>
</tr>
<tr>
<td>KSDATA</td>
<td>&amp;&amp;SYSUID..DATA</td>
<td>specifies a prefix name that Log Master uses to derive the data set name for a VSAM data set</td>
<td>page 51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this data set as overflow storage for key store work areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, DATA</td>
<td></td>
</tr>
<tr>
<td>KSMEMORY</td>
<td>100M, 10M</td>
<td>specifies the amount of memory that is used to allocate a temporary working storage area called a key store that Log Master uses for processing DB2 log records</td>
<td>page 49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MEMORY</td>
<td></td>
</tr>
<tr>
<td>KSMGMT</td>
<td>none</td>
<td>specifies a valid management class within the IBM product DFSMS</td>
<td>page 52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this management class to allocate VSAM data sets for use as overflow storage for key store work areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MGMTCLAS</td>
<td></td>
</tr>
<tr>
<td>KSSPACE</td>
<td>100, 100</td>
<td>specifies the primary and secondary amounts of space that Log Master uses to allocate overflow data sets for key stores (temporary working storage areas)</td>
<td>page 50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, SPACE</td>
<td></td>
</tr>
<tr>
<td>KSSTOR</td>
<td>none</td>
<td>specifies a valid storage class within the IBM product DFSMS</td>
<td>page 51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this storage class to allocate VSAM data sets for use as overflow storage for key store work areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, STORCLAS</td>
<td></td>
</tr>
<tr>
<td>Installation option</td>
<td>Shipped value</td>
<td>Brief description</td>
<td>Reference</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>KSVOLS</td>
<td>none</td>
<td>specifies the volumes that Log Master uses to allocate VSAM data sets to use as overflow storage for key store work areas</td>
<td>page 53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, VOLUME</td>
<td></td>
</tr>
<tr>
<td>LLOGAUTH</td>
<td>NO</td>
<td>determines whether Log Master honors DB2 security at the table level when a job accesses logical log input</td>
<td>page 45</td>
</tr>
<tr>
<td>LOADFMT</td>
<td>LOGMAST</td>
<td>specifies the overall format for the load data files and load control files</td>
<td>page 69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOGSCAN, LOAD FORMAT</td>
<td></td>
</tr>
<tr>
<td>LOBALLOC</td>
<td>CYLINDERS</td>
<td>specifies the units that Log Master uses to allocate LOB or XML VSAM files</td>
<td>page 69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses LOB or XML VSAM files to store data from LOB or XML columns in selected log records.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, CYLINDERS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRACKS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XMLOPTS, CYLINDERS</td>
<td></td>
</tr>
<tr>
<td>LOBDATA</td>
<td>none</td>
<td>specifies a valid data class within the IBM product DFSMS</td>
<td>page 69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this data class to allocate LOB or XML VSAM files.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, DATACLAS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMLOPTS, DATACLAS</td>
</tr>
<tr>
<td>LOBDUPS</td>
<td>ERROR</td>
<td>determines how Log Master responds to a duplicate data set condition as it attempts to create a LOB or XML VSAM file</td>
<td>page 70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, DUPLICATE DATASET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMLOPTS, DUPLICATE DATASET</td>
</tr>
<tr>
<td>LOBLIMIT</td>
<td>10</td>
<td>determines the maximum number of LOB or XML VSAM files that Log Master can create to store the LOB or XML data of one LOB or XML column</td>
<td>page 70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, LOBLIMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMLOPTS, XMLLIMIT</td>
</tr>
<tr>
<td>LOBMGMT</td>
<td>none</td>
<td>specifies a valid management class within the IBM product DFSMS</td>
<td>page 71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this management class to allocate LOB or XML VSAM files.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, MGMTCLAS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMLOPTS, MGMTCLAS</td>
</tr>
<tr>
<td>LOBPREF</td>
<td>&amp;&amp;SYSUID..LOB</td>
<td>specifies a prefix name that Log Master uses to derive the name for a LOB VSAM file</td>
<td>page 71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, PREFIX</td>
<td></td>
</tr>
<tr>
<td>LOBSpace</td>
<td>10, 10</td>
<td>specifies the primary and secondary amounts of space that Log Master uses to allocate LOB or XML VSAM files</td>
<td>page 72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, SPACE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMLOPTS, SPACE</td>
</tr>
<tr>
<td>LOBSTOR</td>
<td>none</td>
<td>specifies a valid storage class within the IBM product DFSMS</td>
<td>page 72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this storage class to allocate LOB or XML VSAM files.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, STORCLAS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMLOPTS, STORCLAS</td>
</tr>
</tbody>
</table>
Table 1  Log Master for DB2 installation options quick reference (part 5 of 7)

<table>
<thead>
<tr>
<th>Installation option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOBVOLS</td>
<td>none</td>
<td>specifies the volumes that Log Master uses to allocate LOB or XML VSAM files</td>
<td>page 72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOBOPTS, VOLUME XMLLOPTS, VOLUME</td>
<td></td>
</tr>
<tr>
<td>LOCCPSEL</td>
<td>(FC,LP,LB)</td>
<td>specifies resource selection for local site copies</td>
<td>page 84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, RESOURCE SELECTION, COPIES</td>
<td></td>
</tr>
<tr>
<td>LOGSORT</td>
<td>WHENEVER</td>
<td>determines how Log Master processes DB2 log data sets from multiple members of a DB2 data sharing group</td>
<td>page 63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOGOPTS, LOGSORT</td>
<td></td>
</tr>
<tr>
<td>LOGTAPES</td>
<td>0</td>
<td>specifies the maximum number of tape units or DASD data sets that Log Master allocates for reading log files</td>
<td>page 64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: LOGOPTS, LOGTAPES</td>
<td></td>
</tr>
<tr>
<td>LRPCT</td>
<td>40</td>
<td>specifies the percentage of key store memory that is allocated to the log record (LR) key store</td>
<td>page 53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MEMPERCENT LOGRECORD</td>
<td></td>
</tr>
<tr>
<td>MIGRATE</td>
<td>YES</td>
<td>directs Log Master to request storage management software to migrate any recalled data sets to their original status</td>
<td>page 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, MIGRATE</td>
<td></td>
</tr>
<tr>
<td>MIGRWAIT</td>
<td>NO</td>
<td>determines whether Log Master terminates at the end of processing or waits for data set migration requests to complete</td>
<td>page 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, WAIT</td>
<td></td>
</tr>
<tr>
<td>MINLOGPT</td>
<td>NO</td>
<td>specifies how Log Master determines the log scan end point</td>
<td>page 60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, MINLOGPT</td>
<td></td>
</tr>
<tr>
<td>MTPRMBR</td>
<td>NO</td>
<td>specifies the scope of the SMSTASKS value</td>
<td>page 79</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, PER MEMBER</td>
<td></td>
</tr>
<tr>
<td>OPNDB2ID</td>
<td>YES</td>
<td>determines how Log Master opens DB2 data sets, such as table spaces or logs</td>
<td>page 44</td>
</tr>
<tr>
<td>OSRCHCNT</td>
<td>3</td>
<td>determines how Log Master responds if the user running the online interface does not enter a log file name when specifying DB2 log data sets</td>
<td>page 77</td>
</tr>
<tr>
<td>PARMSTOR</td>
<td>none</td>
<td>specifies a valid storage class within the IBM product DFSMS Log Master uses this storage class for dynamic allocation of data sets other than sort work data sets.</td>
<td>page 48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: SOROPTS, PARMOPTS STORCLAS</td>
<td></td>
</tr>
<tr>
<td>PARMUNIT</td>
<td>none</td>
<td>specifies a unit name that Log Master uses for dynamic allocation of data sets other than sort work data sets</td>
<td>page 48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: SOROPTS, PARMOPTS UNIT</td>
<td></td>
</tr>
<tr>
<td>PLANALPB</td>
<td>ALPover</td>
<td>specifies the batch execution DB2 plan for Log Master</td>
<td>page 42</td>
</tr>
<tr>
<td>PLANALPO</td>
<td>ALPOver</td>
<td>specifies the online interface DB2 execution plan for Log Master</td>
<td>page 42</td>
</tr>
<tr>
<td>Installation option</td>
<td>Shipped value</td>
<td>Brief description</td>
<td>Reference</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PLANESQL</td>
<td>ALP</td>
<td>specifies either a prefix or an eight-character plan name that Log Master uses to execute generated SQL Syntax override: EXECSQL, PLAN NAME</td>
<td>page 42</td>
</tr>
<tr>
<td>PUBLICPLAN</td>
<td>YES</td>
<td>determines whether Log Master grants the authority to run Log Master (execute against the Log Master base plan) to PUBLIC</td>
<td>page 43</td>
</tr>
<tr>
<td>QBLRBUF</td>
<td>48M</td>
<td>determines the maximum amount of memory that Log Master uses for the log record buffer Syntax override: OPTION, LOGRECORD BUFSIZE</td>
<td>page 55</td>
</tr>
<tr>
<td>REBUILD</td>
<td>NO</td>
<td>determines the keyword that Log Master includes in the recovery syntax that it generates for an automated drop recovery job step</td>
<td>page 74</td>
</tr>
<tr>
<td>REMCPSEL</td>
<td>(RP,RB,FC)</td>
<td>specifies resource selection for remote site copies Syntax override: OPTION, RESOURCE SELECTION, COPIES</td>
<td>page 85</td>
</tr>
<tr>
<td>RESINV</td>
<td>0K</td>
<td>specifies the amount of memory (below the 16-MB line) that Log Master instructs the sort routine to reserve for processing other than sort processing Syntax override: SOROPTS, RESINV</td>
<td>page 47</td>
</tr>
<tr>
<td>SMCORE</td>
<td>0M, 0K</td>
<td>specifies the maximum amount of memory that the sort routine uses Syntax override: SOROPTS, SMCORE</td>
<td>page 46</td>
</tr>
<tr>
<td>SMSTASKS</td>
<td>0</td>
<td>limits the number of early recall subtasks that Log Master creates Syntax override: OPTION, SMSTASKS</td>
<td>page 79</td>
</tr>
<tr>
<td>SORTDYN</td>
<td>USE</td>
<td>indicates whether Log Master uses the dynamic allocation rules for the sort routine, or allows the user to specify allocation characteristics Syntax override: SOROPTS, SORTDYN</td>
<td>page 49</td>
</tr>
<tr>
<td>TAPES</td>
<td>CART, TAPE</td>
<td>specifies tape esoteric names of datasets that Log Master allocates for reading log files. Syntax override: NONE</td>
<td>page 74</td>
</tr>
<tr>
<td>TRCHARS</td>
<td>&gt;&gt;</td>
<td>specifies the character string that Log Master uses to indicate truncated DB2 object names within the online interface</td>
<td>page 78</td>
</tr>
<tr>
<td>TRPOS</td>
<td>END</td>
<td>defines which part of DB2 object names Log Master truncates for display within the online interface</td>
<td>page 78</td>
</tr>
<tr>
<td>TZRULE</td>
<td>NONE</td>
<td>specifies the rules that determine when Daylight Saving Time (DST) begins and ends Use this installation option to enable DST adjustments.</td>
<td>page 81</td>
</tr>
<tr>
<td>ULOGORD</td>
<td>ASCENDING</td>
<td>specifies the order in which Log Master reads DB2 log files that you explicitly define (either through the online interface or the INPUT DB2LOG keyword of the LOGSCAN statement)</td>
<td>page 62</td>
</tr>
<tr>
<td>UNITWAIT</td>
<td>YES</td>
<td>determines whether dynamic file allocations should wait for unavailable resources</td>
<td>page 64</td>
</tr>
<tr>
<td>URIDTHR</td>
<td>0</td>
<td>specifies the maximum number of data changes (insert, update, or delete actions) that Log Master processes within one unit of work, as represented by one unit of recovery identifier (URID)</td>
<td>page 67</td>
</tr>
</tbody>
</table>
### Table 1  Log Master for DB2 installation options quick reference (part 7 of 7)

<table>
<thead>
<tr>
<th>Installation option</th>
<th>Shipped value</th>
<th>Brief description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>URPCT</td>
<td>20</td>
<td>specifies the percentage of key store memory that is allocated to the URID (UR) key store</td>
<td>page 53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: STOREOPTS, MEMPERCENT URID</td>
<td></td>
</tr>
<tr>
<td>USELGRNG</td>
<td>NO</td>
<td>indicates whether Log Master uses the SYSIBM.SYSLGRNX tables when determining the ranges for a log scan&lt;sup&gt;a&lt;/sup&gt;</td>
<td>page 61</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION, USELGRNG</td>
<td></td>
</tr>
<tr>
<td>USELOGS</td>
<td>ACT1, ACT2, ARCH1, ARCH2</td>
<td>specifies the order in which Log Master reads active and archive log files</td>
<td>page 62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: OPTION RESOURCE SELECTION LOGS</td>
<td></td>
</tr>
<tr>
<td>UTILPLNS</td>
<td>none</td>
<td>specifies the plan names of any products that can perform LOAD LOG YES, LOAD REPLACE, or REORG LOG YES actions</td>
<td>page 44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: XMLOPTS PREFIX</td>
<td></td>
</tr>
<tr>
<td>WKSTOR</td>
<td>none</td>
<td>specifies a valid storage class within the IBM product DFSMS</td>
<td>page 48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Log Master uses this storage class for dynamic allocation of large temporary files and sort work data sets.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: SORTOPTS, WORKOPTS STORCLAS</td>
<td></td>
</tr>
<tr>
<td>WKUNIT</td>
<td>none</td>
<td>specifies the unit name that Log Master uses for dynamic allocation of sort work data sets</td>
<td>page 48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: SORTOPTS, WORKOPTS UNIT</td>
<td></td>
</tr>
<tr>
<td>WORKNUM</td>
<td>0</td>
<td>specifies the number of sort work data sets that Log Master uses</td>
<td>page 48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: SORTOPTS, WORKOPTS NUM</td>
<td></td>
</tr>
<tr>
<td>XMLPREF</td>
<td>&amp;SYSUID..XML</td>
<td>specifies a prefix name that Log Master uses to derive the name for an XML VSAM file</td>
<td>page 73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: XMLOPTS PREFIX</td>
<td></td>
</tr>
<tr>
<td>ZIIP</td>
<td>ENABLED</td>
<td>determines whether zIIP processing is enabled</td>
<td>page 84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax override: ZIIP</td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup> This option applies only in a data sharing environment.
Installation option descriptions

This section contains more details about each installation option. If an installation option has a default value, the default value is shown following the option name. Also noted is SYSIN syntax, if available, to override the setting of the installation option at runtime, with a reference to more information.

**PLANALPB=ALPB\textit{vvr}**

Specifies the batch execution DB2 plan for Log Master. The default value is \textit{ALPB\textit{vvr}}, where \textit{vvr} represents the version and release level of Log Master. You can specify any valid DB2 plan name.

**PLANALPO=ALPO\textit{vvr}**

Specifies the online interface DB2 execution plan for Log Master. The default value is \textit{ALPO\textit{vvr}}, where \textit{vvr} represents the version and release level of Log Master. You can specify any valid DB2 plan name.

**PLANESQL=ALP**

Specifies either a prefix or an eight-character plan name that Log Master uses to execute generated SQL. Log Master invokes the DB2 BIND command to create the actual plan dynamically before executing any generated SQL. The default value is ALP. Values for PLANESQL are as follows:

- **plan name prefix (1-7 characters)**
  If you choose to specify a prefix, Log Master dynamically binds and frees the plan when the job starts and ends. Log Master interprets any character string from one to seven characters long as a plan prefix.

- **plan name (8 characters)**
  If you use a plan name that is eight characters long, Log Master binds the plan each time a job is run, but does not free the plan. This enables a user with BINDADD authority to create the selected plan, and allows other users to execute under this plan if they have BIND REPLACE authority for the plan.

You can override the installation option value at runtime by specifying the EXECSQL statement, PLAN NAME syntax option. For more information, see page 307.
**BINDOWN=**

Specifies an authorization ID that Log Master uses to bind an application plan when Log Master executes generated SQL statements. This authorization ID must have BINDADD authority for the DB2 subsystem where the SQL statements are executed.

If you do not define a value for BINDOWN, Log Master uses the primary authorization ID of the job.

You can override the installation option value at runtime by specifying the EXECSQL statement, BINDOWNER syntax option. For more information, see page 306.

**BINDQUALIFIER=ALPvvv**

Specifies a qualifier value that Log Master uses during execution to dynamically bind internal packages that are distributed with Log Master. This value also determines which set of synonyms Log Master uses. Log Master does not use this installation option when it binds packages to execute input SQL statements or when it generates SQL statements.

Specify any value that is valid for the QUALIFIER option of the DB2 BIND command, up to 16 bytes in length. The default value is ALPvvv, where vv represents a two-character version number and v represents a one-character release number.

**PUBLICPLAN=YES**

Determines whether Log Master grants the authority to run Log Master (execute against the Log Master base plan) to PUBLIC. Use this option to avoid specific grants of authority to individual users. Valid values include:

- **YES**
  Log Master grants EXECUTE authority to PUBLIC. All users have authority to run Log Master. The default value is YES, and BMC Software recommends that you use this value.

- **NO**
  Log Master avoids granting EXECUTE authority to PUBLIC. You must grant EXECUTE authority to specific users as required.
**UTILPLNS=**

Specifies the plan names of any third-party DB2 load or reorganization utilities. If Log Master knows the plan names of these utilities, it can recognize and process log records that they create.

If your environment uses third-party load or reorganization utilities, enter the plan names for the utilities. For the BMC Software LOADPLUS® for DB2 product and the REORG PLUS for DB2 product, use the value of the PLAN installation option. The default value of PLAN is $\textit{PPP}_{vvrm}$, where $\textit{PPP}$ represents the product code (AMU or ARU), $vv$ represents the version level, $r$ represents the release level, and $m$ represents the maintenance level of the product.

This option has no default value. If you do not enter the plan names for these utility products, Log Master does not interpret their log records.

**OPNDB2ID=YES**

Determines how Log Master opens the data sets that DB2 addresses directly (such as the VSAM files that contain table spaces and the archive log files). Values for OPNDB2ID are as follows:

- **YES**: Log Master uses the ID of the DB2 DBM1 address space as defined in Resource Access Control Facility (RACF) instead of using the RACF ID of the user who is running the product. For more information, see “RACF authority” on page 29.

- **NO**: Log Master uses the ID of the user who is running the product to access DB2 data sets (such as table space files and log files). The ID must have the proper access to all required DB2 objects.

**DB2AUTH=NO**

Determines whether Log Master honors DB2 security at the table level when a job accesses DB2 log input. If you enable this type of security and a Log Master job does not have the authorization that is required to access a given table, Log Master suppresses log records relating to the table in the job’s output. Depending on the message level set by the AMSGLEV installation option, the job might not receive any notification that records have been suppressed. Values for DB2AUTH are as follows:

- **YES**: Indicates that Log Master honors DB2 security as it processes DB2 log input.

- **NO**: Indicates that Log Master does not honor DB2 security. The default value is NO.
Installation option descriptions

### Log Master honors either native DB2 security or external security (the DSNX@XAC exit provided by RACF, ACF2 for DB2, Top Secret for DB2, or other security packages).

#### LLOGAUTH=NO

Determines whether Log Master honors DB2 security at the table level when a job accesses logical log input. Log Master can read input from logical log files at a time after the files were originally created, or on a different DB2 subsystem, or in a different security environment. If you enable this type of security and a Log Master job does not have the authorization needed to access a given table, Log Master suppresses log records that relate to the table in the job’s output. Depending on the message level set by the AMSGLEV installation option, the job might not receive any notification that records have been suppressed.

Values for LLOGAUTH are as follows:

**YES**
Indicates that Log Master honors DB2 security when it processes logical log input.

**NO**
Indicates that Log Master does not honor DB2 security. The default value is NO.

#### NOTE
When you enable DB2 security, ensure that your jobs have the proper security attributes. With this type of security, Log Master jobs that access DB2 log records that relate to a given database object must run under an authorization ID that has one of the following authorities:

- SYSADM or SYSCTRL authority (checked once per job)
- DBADM or DBCTRL authority for the database that contains the object (checked once per database, if required)
- SELECT authority for the object (declared by specific GRANT statements stored in the SYSIBM.SYSTABAUTH table, checked once per table if required)
Log Master honors either native DB2 security or external security (the DSNX@XAC exit provided by RACF, ACF2 for DB2, Top Secret for DB2, or other security packages).

**AMSGLEV=0**

Determines the type of messages that Log Master issues when a job encounters a DB2 security exception as it processes either DB2 log or logical log input. This installation option is related to the DB2AUTH and LLOGAUTH options. Values for AMSGLEV are as follows:

0  
Log Master issues minimal messages. The messages do not indicate that log records are suppressed in the output. The default value is 0.

1  
Log Master issues summary messages. The messages indicate that log records are suppressed, but do not specify the tables that are involved or how many records are suppressed.

2  
Log Master issues detailed messages. The messages indicate that log records are suppressed, which tables contain the suppressed records, and how many records are suppressed.

**KEYDSNAM=**

Specifies the name of the key data set that is used for encrypted image copies. The BMC Software COPY PLUS for DB2 product can create encrypted image copies. When the name of the key data set is provided through this option, Log Master can use encrypted image copies for row completion processing, or to obtain compression dictionaries. (Both Log Master and COPY PLUS must use the same instance of the BMC Software table BMC_BMCXCOPY.)

This installation option does not have a default value. For more information about encrypted image copies, including the format of the key data set and suggestions for managing them, see the sections on encrypted image copies in the *COPY PLUS for DB2 Reference Manual*.

**SMCORE=0M,0K**

Specifies the maximum amount of memory that the sort routine can use. The SMCORE installation option requires you to specify two values:

- The first value specifies the total amount of memory that is used both below and above the 16-MB line.
■ The second value specifies only the amount of memory that is used below the 16-MB line.

You can enter either value as a number of bytes, kilobytes (using the suffix K), or megabytes (using the suffix M).

BMC Software recommends that you specify zero for both SMCORE values (as 0M or 0K). This action specifies the Log Master default values (currently 6M, 384K if a FILSZ value is not specified) for the amount of storage to allocate above and below the 16-MB line.

If FILSZ estimate syntax is coded under SORTOPTS, Log Master determines the default SMCORE values by using the minimum storage required by BMCSORT when all of the following conditions exist:

■ You accept the default values of 0 for the SMCORE installation option.
■ You specify a value for the FILSZ keyword in the SORTOPTS statement.
■ You do not specify SMCORE values in the SORTOPTS statement.

The value of the SMCORE option affects the performance of Log Master sort actions. To further affect the performance of sort actions, you can also

■ control how the sort routine allocates sort work data sets, as described in “Defining sort work data sets” on page 102
■ estimate the amount of data to be sorted, and pass that estimate to the sort routine, as described in “Sort file size parameters” on page 154

You can override either installed SMCORE value at runtime. For more information, see SORTOPTS statement, SMCORE syntax option (page 151).

**RESINV=0K**

Specifies the amount of memory (below the 16-MB line) that Log Master instructs the sort routine to reserve for processing other than sort processing.

BMC Software recommends that you always specify this parameter as zero (the default value). This action allows Log Master to use default values to allocate sort memory. You can enter a value as a number of bytes, kilobytes (using the suffix K), or megabytes (using the suffix M).

You can override the installation option value at runtime by specifying the SORTOPTS statement, RESINV syntax option (page 151).
PARMSTOR=

 Specifies a valid Data Facility Storage Management System (DFSMS) storage class that Log Master uses for dynamic allocation of data sets other than sort work data sets. This option does not have a default value. Do not specify a value of VIO for this option. For more information about using this installation option with other installation options, see Table 2 on page 52.

 You can override the installation option value at runtime by specifying the SORTOPTS statement, PARMOPTS STORCLAS syntax option (page 152).

PARMUNIT=

 Specifies a unit name that Log Master uses for dynamic allocation of data sets other than sort work data sets. Specify any unit name where you can allocate temporary data sets. This option has no default value. If you do not specify a value, the storage management software in your environment uses its default values. For more information about using this installation option with other installation options, see Table 2 on page 52.

 You can override the installation option value at runtime by specifying the SORTOPTS statement, PARMOPTS UNIT syntax option (page 152).

WORKNUM=0

 Specifies the number of sort work data sets that Log Master uses. The default value is 0, which indicates that the number of data sets is set dynamically by the sort routine. You can specify any number from 1 to 255.

 You can override the installation option value at runtime by specifying the SORTOPTS statement, WORKOPTS NUM syntax option (page 153).

WKSTOR=

 Specifies a valid DFSMS storage class that Log Master uses for dynamic allocation of large temporary files and sort work data sets. This option has no default value. For more information about using this installation option with other installation options, see Table 2 on page 52.

 Specify any valid DFSMS storage class.

 You can override the installation option value at runtime by specifying the SORTOPTS statement, WORKOPTS STORCLAS syntax option (page 153).

WKUNIT=

 Specifies the unit name that Log Master uses for dynamic allocation of sort work data sets. Specify any unit name where you can allocate temporary DASD data sets. This option has no default value. If you do not specify a value, the storage management software in your environment uses its default values. For more information about using this installation option with other installation options, see Table 2 on page 52.
Specify any valid unit.

You can override the installation option value at runtime by specifying the SORTOPTS statement, WORKOPTS UNIT syntax option (page 153).

**SORTDYN=USE**

Indicates whether Log Master uses the sort routine’s dynamic allocation rules or allows a user to specify allocation characteristics. Values for SORTDYN are as follows:

**USE**

This value allows a user to override the sort routine’s dynamic allocation of space that is used for sort work data sets. The default value is USE.

**SYSTEM**

This value indicates that the sort routine always determines the dynamic allocation of space that is used for sort work data sets.

You can override the installation option value at runtime by specifying the SORTOPTS statement, SORTDYN syntax option (page 152).

**KSMEMORY=100M, 10M**

Specifies the amount of memory that Log Master uses for a temporary working storage area called a *key store*. If you experience large amounts of logging on your DB2 subsystem, or if you perform large log scans, allocate larger amounts of memory. This action reduces the possibility of Log Master spilling to DASD storage and degrading performance. The KSMEMORY installation option requires you to specify two values:

- The first value specifies the amount of memory that Log Master uses to store internal data as it processes log records. You might need to override this default value frequently, because the memory required for efficient processing can change dramatically based on the number of DB2 log data records that Log Master is processing.

- The second value specifies a number. Log Master does not currently use this value, but reserves it and requires it to be present for correct syntax. If you specify the first value without the second value, the Log Master installation options assembly macro produces a syntax error and terminates processing with return code 8.

You can enter both values as a number of bytes, kilobytes (using the suffix K) or megabytes (using the suffix M).
Allocation amounts are site specific. However, if you set aside insufficient memory for this working storage area, product performance might be degraded. The amount that you allocate is distributed between key stores based on the values of the xxPCT installation options or the values specified in the MEMPERCENT keyword of the STOREOPTS statement. For more information, see “URPCT=20” on page 53 or “MEMPERCENT” on page 142.

You can override the installation option value at runtime by specifying the STOREOPTS statement, MEMORY syntax option (page 140).

**KSALLOCU=CYLS**

Specifies how Log Master allocates VSAM data sets to use as overflow storage for key store work areas. The default value is CYLS (cylinders). Use the TRKS keyword to specify tracks.

You can override the installation option value at runtime by specifying the STOREOPTS statement, CYLINDERS | TRACKS syntax option (page 141).

**KSSPACE=(100,100)**

Specifies the primary and secondary amounts of space that Log Master uses to allocate overflow data sets for key stores (temporary working storage areas). Log Master creates these data sets when it must process more log records than it can place in memory. This installation option works with the KSALLOCU option. The default values of KSSPACE are 100 and 100.

Allocation amounts are site specific. However, if you allocate insufficient space to process all of the DB2 log records that are needed for a specific log scan, your Log Master job can fail with an allocation error.

Specify allocation amounts in either tracks or cylinders.

You can override the installation option value at runtime by specifying the STOREOPTS statement, SPACE syntax option (page 141).

**KSCLUST=&SYSUID..CLUSTER**

Establishes a prefix name that Log Master uses to derive the cluster name for a VSAM data set. Log Master concatenates the prefix name that you enter with the job number and an internal identifier to construct a cluster name. Log Master uses this data set as overflow storage for key store work areas. Values for KSCLUST are as follows:

- any valid name, up to 34 characters long
- a symbolic of &SYSUID to substitute a TSO prefix for the prefix name
You can override the installation option value at runtime by specifying the STOREOPTS statement, CLUSTER syntax option (page 142).

**KSDATA=&&SYSUID..DATA**

Specifies a prefix that Log Master uses to derive the data portion name for a VSAM data set. Log Master concatenates the prefix name that you enter with the job number and an internal identifier to construct a data name. Log Master uses this data set as overflow storage for key store work areas. Values for KSDATA are as follows:

- any valid name, up to 34 characters long
- a symbolic of &SYSUID to substitute a TSO prefix for the prefix name

**NOTE**
If you use &SYSUID, enter &&SYSUID in the actual option macro to prevent the assembler from flagging the definition as an error.

You can override the installation option value at runtime by specifying the STOREOPTS statement, DATA syntax option (page 142).

**DSNSYMPRE=YES**

Forces the prefix for &TABNAME and &WORKID in input and output data set names.

Values for DSNSYMPRE are as follows:

**YES**
Always adds the prefix. The default value is YES.

**NO**
Adds the prefix only if the first character of the node is not an alphabetic or a national character.

**KSSTOR=**

Specifies a valid DFSMS storage class that Log Master uses to allocate VSAM data sets. Log Master allocates these data sets when it must process more log records than it can place in memory, and requires overflow storage for key store work areas.

Specify any valid DFSMS storage class. This option has no default value.
You can override the installation option value at runtime by specifying the STOREOPTS statement, STORCLAS syntax option (page 141).

Table 2 lists installation options that allocate data sets by using either DFSMS storage or management classes, or individual volumes or units.

BMC Software recommends that you do not use installation options from both sets in the same job. Depending on the options that you specify, you might encounter errors when you assemble the installation options module.

Table 2 Installation options for data set allocation

<table>
<thead>
<tr>
<th>DFSMS installation option</th>
<th>Individual volume/unit installation option</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSDACLS</td>
<td>KSVOLS</td>
</tr>
<tr>
<td>KSMGMT</td>
<td>KSVOLS</td>
</tr>
<tr>
<td>KSSTOR</td>
<td>KSVOLS</td>
</tr>
<tr>
<td>LOBDATA</td>
<td>LOBVOLS</td>
</tr>
<tr>
<td>LOBMGMT</td>
<td>LOBVOLS</td>
</tr>
<tr>
<td>LOBSTOR</td>
<td>LOBVOLS</td>
</tr>
<tr>
<td>PARMSTOR</td>
<td>PARMUNIT</td>
</tr>
<tr>
<td>WKSTOR</td>
<td>WKUNIT</td>
</tr>
</tbody>
</table>

**KSDACLS=**

Specifies a valid DFSMS data class that Log Master uses to allocate VSAM data sets. Log Master allocates these data sets when it must process more log records than it can place in memory. It uses the data sets as overflow storage for key store work areas. This option has no default value. For more information about using this installation option with other installation options, see Table 2 on page 52.

Specify any valid DFSMS data class.

You can override the installation option value at runtime by specifying the STOREOPTS statement, DATAACLAS syntax option (page 141).

**KSMGMT=**

Specifies a valid DFSMS management class that Log Master uses to allocate VSAM data sets. Log Master allocates these data sets when it must process more log records than it can place in memory. It uses the data sets as overflow storage for key store work areas. This option has no default value. For more information about using this installation option with other installation options, see Table 2 on page 52.

Specify any valid DFSMS management class.

You can override the installation option value at runtime by specifying the STOREOPTS statement, MGMTCLAS syntax option (page 142).
**KSVOLS=**

Specifies the volumes that Log Master uses to allocate VSAM data sets. Log Master uses these data sets as overflow storage for key store work areas. Log Master uses the volumes that you enter in the VOLUMES parameter of a DEFINE CLUSTER command. For more information about using this installation option with other installation options, see Table 2 on page 52.

Specify any valid VOLSER as a value for KSVOLS.

You can override the installation option value at runtime by specifying the STOREOPTS statement, VOLUME syntax option (page 142).

**URPCT=20**

Specifies the percentage of key store memory that is allocated to the URID (UR) key store. Enter an integer value between 1 and 99. The default value is 20.

In the UR key store, Log Master maintains information about each unit of recovery that it encounters within the range of a log scan.

Before changing the distribution of key store memory, see STOREOPTS statement, MEMPERCENT syntax option (page 142).

**Considerations for all xxPCT values**

All xxPCT values represent percentages of the total amount of key store memory. (For more information, see “KSMEMORY=100M, 10M” on page 49.) The percentage values of the xxPCT installation options (URPCT, LRPCT, FCPCT, BCPCT, and ANPCT) must add up to 100 percent, including any default values that you do not specify explicitly. To avoid confusion, specify a value for all xxPCT options even if you are not changing the default value.

When a job reads input from a logical log file, Log Master redistributes all available key store memory to the UR key store, regardless of your specifications. Log Master takes this action because it does not require the other key stores to process logical log input.

**LRPCT=40**

Specifies the percentage of key store memory that is allocated to the log record (LR) key store. Enter an integer value between 1 and 99. For more information about specifying all xxPCT installation option values, see “Considerations for all xxPCT values.”

In the LR key store, Log Master maintains a copy of log records that require row completion processing. The default value is 40.
Before changing the distribution of key store memory, see STOREOPTS statement, MEMPERCENT syntax option (page 142).

**FCPCT=10**

Specifies the percentage of key store memory that is allocated to the forward completion (FC) key store. Enter an integer value between 0 and 98. For more information about specifying all xxPCT installation option values, see “Considerations for all xxPCT values” on page 53.

In the FC key store, Log Master maintains copies of complete row images that might be used for row completion processing. The default value is 10.

Before changing the distribution of key store memory, see related material in STOREOPTS statement, MEMPERCENT syntax option (page 142).

**BCPCT=10**

Specifies the percentage of key store memory that is allocated to the backward completion (BC) key store. Enter an integer value between 0 and 98. For more information about specifying all xxPCT installation option values, see “Considerations for all xxPCT values” on page 53.

In the BC key store, Log Master maintains copies of complete row images that might be used for row completion processing. The default value is 10.

Before changing the distribution of key store memory, see related material in STOREOPTS statement, MEMPERCENT syntax option (page 142).

**ANPCT=20**

Specifies the percentage of key store memory that is allocated to the anomaly (AN) key store. Enter an integer value between 0 and 98. For more information about specifying all xxPCT installation option values, see “Considerations for all xxPCT values” on page 53.

In the AN key store, Log Master maintains information used to produce the Backout Integrity report. If a job does not specify a Backout Integrity report as output, Log Master redistributes this memory evenly to the FC and BC key stores. The default value is 20.

Before changing the distribution of key store memory, see related material in STOREOPTS statement, MEMPERCENT syntax option (page 142).
**QBLRBUF=48M**

Determines the maximum amount of memory that Log Master uses for the log record buffer. If Log Master fills up this amount of memory, and more log records remain to be processed, Log Master writes log records (spills) to DASD storage and performance can be degraded.

The log record buffer is an internal memory and file structure. For more information about this buffer, see “LOGRECORD BUFSIZE nnn” on page 133.

Specify the amount of memory in bytes, kilobytes (use the suffix K), or megabytes (use the suffix M). The default value is 48M. The minimum value is 512K.

--- **NOTE**

BMC Software recommends retaining the default value unless you experience repeated performance problems related to memory usage. For help with possible QBLRBUF changes, contact BMC Software Customer Support.

You can override the installation option value at runtime by specifying the OPTION statement, LOGRECORD BUFSIZE syntax option (page 133).

**DICTSPC=50M**

Specifies a limit on the amount of memory that Log Master uses to store compression dictionaries during processing. Log Master does not allocate any memory until it accesses a compressed table space. Adjusting this value can change the performance of log scans that read log records of compressed table spaces. The default value is 50 MB.

The following considerations apply to the DICTSPC option:

- The minimum value is 192 KB (the size of three dictionaries). If you enter a value that is less than the minimum, Log Master ignores your specified value and uses the minimum value.

- Allocation amounts are site specific. Use the following guidelines to calculate a DICTSPC value:

  - For each compressed table space that is used by a job, multiply the number of partitions times the page size (for example, 32K), and then multiply that value times 16.
  
  - Add the values for all compressed table spaces that are used by the job.
To determine whether to increase an existing (or the default) DICTSPC value, examine your job’s BMC097677 messages. If these messages indicate that a large number of dictionaries are deleted or reloaded, consider increasing the DICTSPC value. Use the following guidelines to determine a more optimal DICTSPC value:

— Find the total number of dictionaries loaded (listed in the BMC097677 messages).
— Multiply the number of dictionaries times 16 (the number of pages in a dictionary).
— Divide the result by 256 (the number of dictionary pages in a megabyte).
— Round up the result to a number of megabytes that is evenly divisible by 10 (for example, round 32.375M to 40M).

Log Master uses this memory dynamically, loading compression dictionaries as required. When it reaches the DICTSPC limit, it discards the least recently used dictionaries. With a low DICTSPC value and large numbers of compressed table spaces, Log Master can load the same dictionary more than once, resulting in degraded performance.

Specify allocation amounts in kilobytes (using the suffix K) or megabytes (using the suffix M).

Log Master honors the DICTSPC limit during most processing, but can exceed the limit when reading a compression dictionary from the DB2 log or from an image copy during completion processing.

You can override the installation option value at runtime by specifying the OPTION statement, DICTIONARYSPACE syntax option (page 128).

**IMAGESRC=ANY**

Determines how Log Master attempts to perform row completion or decompression processing. The default value is ANY. For more information about row completion, see “IMAGESOURCE” on page 124, or see the chapter on Log Master for DB2 expert information in the *Log Master for DB2 User Guide*.

Values for IMAGESRC are as follows:

**ANY**

Directs Log Master to determine the best approach for completing or decompressing rows based on available resources. The CSTFILE, CSTMOUNT, and CSTSEG installation options influence the approach that Log Master takes when you specify ANY.

**TABLESPACE**

Specifies that Log Master performs row completion or decompression only by locating the row on the table space and rolling back intervening log records. This choice increases the risk that Log Master can terminate with either BMC097386 “unable to decompress” or one of several “unable to complete” error messages.
**SYSCOPY**
Specifies that Log Master uses the SYSIBM.SYSCOPY table in the DB2 catalog to locate resources to perform row completion or decompression. SYSCOPY resources include image copies or other events, such as LOAD LOG YES actions. Log Master obtains a row image from the image copy or log records of the event, and then applies any intervening log records. Log Master cannot use DFSMS concurrent image copies for row completion or decompression. For more information, see “Reading image copies” on page 125.

**LOGONLY**
Specifies that Log Master performs row completion or decompression using only information available on the log. This method of row completion or decompression is successful only when the entire row is logged within the range of the log scan.

You can override the installation option value at runtime by specifying the OPTION statement, IMAGESOURCE syntax option (page 124).

**CSTFILE=2000**
Defines the relative cost of processing an additional data set, expressed in terms of reading one page from the DB2 log. This installation option works with the CSTMOUNT, CSTSEG, and IMAGESRC options. The default value is 2000.

When Log Master must choose between opening a data set (for example, a table space) and reading more pages in the log, Log Master uses the CSTFILE value to help make the decision. For example, the default value indicates that the cost in your environment to open and process a single data set is equivalent to reading 2000 pages from the log.

The value of CSTFILE must be between 0 and 1000000:

- A value of 0 indicates that the cost of opening a data set is the same as reading one page of the DB2 log.
- A value of 1000000 indicates that opening a data set is equivalent to reading 1,000,000 pages of the DB2 log.

**NOTE**
BMC Software recommends consulting BMC Software Customer Support before changing the default value.
Installation option descriptions

To improve performance, as Log Master chooses between available image copy data sets, it assigns a lower CSTFILE value to any Instant Snapshot image copies that are available. Instant Snapshot image copies are created on intelligent hardware storage devices by the BMC Software COPY PLUS for DB2 product with Snapshot Upgrade Feature (SUF).

You can override the installation option value at runtime by specifying the OPTION statement, FILECOST syntax option (page 126).

**CSTMOUNT=25000**

Defines the relative cost of a single tape mount, expressed in terms of the cost to read one page from the DB2 log. This installation option works in conjunction with the CSTFILE, CSTSEG, and IMAGESRC options.

When Log Master must choose between mounting a single tape and reading more pages in the log, Log Master uses the CSTMOUNT value to help make the decision. For example, the default value indicates that the cost in your environment to mount a single tape is equivalent to reading 25,000 pages from the log.

The value of CSTMOUNT must be between 0 and 1000000, as follows:

- A value of 0 indicates that the cost of mounting a tape is the same as reading one page of the DB2 log.
- A value of 1000000 indicates that mounting a tape is equivalent to reading 1,000,000 pages of the DB2 log.

**NOTE**

BMC Software recommends consulting BMC Software Customer Support before changing the default value.

You can override the installation option value at runtime by specifying the OPTION statement, FILECOST syntax option (page 126).

**CSTSEG=2000000000**

Assigns a relative cost to obtaining information from a segmented table space, expressed in terms of the cost to read one page from the DB2 log. The default value is 2,000,000,000.

Use this installation option to manage the risks associated with segmented or universal table spaces that are affected by mass delete actions (DELETE statements with no WHERE clause, or TRUNCATE statements). If Log Master uses a segmented or universal table space as a source for row completion, and a mass delete action affects that table space, Log Master cannot successfully complete log records.
Set the CSTSEG value higher to avoid failures in completion processing, but be aware that jobs requiring row completion might run slower, because Log Master tends to use image copies for row completion rather than the table space. Set the CSTSEG value lower to improve the performance of jobs that require row completion, but be aware that you are increasing the risk of row completion failure.

The value of CSTSEG must be between 0 and 4000000000, as follows:

- A value of 0 indicates that the cost of reading a segmented or universal table space is the same as reading one page of the DB2 log.
- A value of 4000000000 indicates that reading a segmented or universal table space is equivalent to reading 4,000,000,000 pages of the DB2 log.

**NOTE**

BMC Software recommends retaining the default value unless you know how many mass delete actions your environment experiences.

You can override the installation option value at runtime by specifying the OPTION statement, FILECOST syntax option (page 126).

**CMPLGRNG=YES**

Indicates whether Log Master uses the SYSIBM.SYSLGRNX tables to reduce the amount of log that Log Master reads during row completion processing. During row completion processing, Log Master might need to read additional log records to assemble a complete image for all log records that are selected in your log scan. If this installation option is set to YES, Log Master can use the SYSLGRNX tables to determine which portions of the log contain no activity related to the log records that it needs to complete. Log Master can then avoid reading unnecessary log ranges and improve performance.

Values for CMPLGRNG are as follows:

**YES**

Indicates that Log Master can use the SYSLGRNX tables to reduce the amount of log that Log Master reads during row completion processing. The default value is YES.

**NO**

Indicates that Log Master does not use the SYSLGRNX tables to reduce the amount of log that Log Master reads during row completion processing.

Specifying YES normally improves Log Master performance. However, Log Master can experience degraded performance reading the SYSLGRNX table if you do not maintain that table (with a DB2 Modify utility). Use the elapsed time value that is provided by message BMC097168 to determine the performance of Log Master SYSLGRNX processing. To improve performance in this situation, specify NO.
Installation option descriptions

**FCUSE=YES**

Determines whether Log Master uses the forward completion (FC) key store to perform a type of row completion processing called **heuristic completion**. This type of processing is separate from and different than the more extensive row completion processing that Log Master performs. To determine the effects of heuristic completion in your environment, examine your job’s output for a series of messages that start with message BMC97396 and list the key store as FCUSE.

---

**NOTE**

BMC Software recommends consulting BMC Software Customer Support before changing the default value.

**YES**

Allows Log Master to perform heuristic completion. The default value is YES.

**NO**

Prevents Log Master from performing heuristic completion.

You can override the installation option value at runtime by specifying the OPTION statement, HEURISTIC FORWARDCOMPLETION syntax option (page 132).

**MINLOGPT=NO**

Specifies how Log Master determines the log scan end point when it runs in a data sharing environment. Set this installation option to YES to obtain the same end point across all members of the data sharing group. This installation option applies only in a data sharing environment.

Values for MINLOGPT are as follows:

**NO**

Indicates that Log Master does not require a common end point for all members. Log Master uses the end point that you specify. The default value is NO. When you run an ongoing log scan in a data sharing environment, Log Master dynamically sets the value of this option to YES (regardless of the value that you specify).

**YES**

Indicates that Log Master requires a common end point for all members of the data sharing group. A common end point is important for ongoing log scans, where Log Master requires common end points and start points to select all desired log records across multiple runs.

For each member, Log Master determines an end point based on the logging activity for that member. Log Master then uses the earliest end point found on any member of the data sharing group.
The following diagram shows the end points that Log Master selects depending on the value of MINLOGPT. Note that when MINLOGPT is YES, Log Master selects only some of the log records on MEMBER01 and does not select any of the log records on MEMBER03. If the log scan is an ongoing log scan, subsequent runs select the log records omitted during this run.

You can override the installation option value at runtime by specifying the OPTION statement, MINLOGPT syntax option (page 126).

**USELGRNG=NO**

Indicates whether Log Master uses the SYSIBM.SYSLGRNX tables to determine the ranges for a log scan. Use this installation option only in a data sharing environment. Log Master can use the SYSLGRNX tables during the initial log scan to determine whether the DB2 log of a data sharing member contains information about your selected database objects. Log Master reads the SYSLGRNX table only when a WHERE clause or filter refers to specific DB2 objects (columns, tables, table spaces, or databases).

Values for USELGRNG are as follows:

**NO**
Indicates that Log Master does not use the SYSLGRNX tables to determine valid ranges for a log scan. The default value is NO.
**YES**
Indicates that Log Master uses the SYSLGRNX tables to determine valid ranges for a log scan.

You can override the installation option value at runtime by specifying the OPTION statement, USELGRNG syntax option (page 127).

**USELOGS=(ACT1, ACT2, ARCH1, ARCH2)**

Specifies the order in which Log Master reads active and archive log files. For all log scans, Log Master searches the log files in the order that you enter. Specify log keywords in any order. Omit keywords for log files that you do not want Log Master to consider. If you omit keywords, Log Master can terminate with error messages if it cannot find required log records in the log files that you have specified. The default order is ACT1, ACT2, ARCH1, and ARCH2.

You can override the installation option value at runtime by specifying the OPTION statement, RESOURCE SELECTION LOGS syntax option (page 134).

Values for USELOGS are as follows:

**ACT1**
Indicates the primary active log file.

**ACT2**
Indicates the dual (secondary) active log file.

**ARCH1**
Indicates the primary archive log file.

**ARCH2**
Indicates the dual (secondary) archive log file.

**ULOGORD=ASCENDING**

Specifies the order in which Log Master reads any DB2 log files that you explicitly define. (To explicitly define log files, use the DB2LOG keyword of the INPUT statement.) Log Master can read the log files in the order that you specify, or it can reverse your order (reading the last-specified log file first). The default value is ASCENDING.
Values for ULOGORD are as follows:

**ASCENDING**
Log Master reads log files in the order in which you specify them with the INPUT DB2LOG keyword. When ULOGORD is ASCENDING, you must list the DB2 log files in order, from lowest RBA/LRSN value to highest. For example, three DB2 log files in sequence from the lowest RBA/LRSN value to highest are LOGFILE01, LOGFILE02, and LOGFILE03. If the ULOGORD value is ASCENDING, the correct order in the INPUT statement is as follows:

```
INPUT DB2LOG (LOGFILE01, LOGFILE02, LOGFILE03)
```

**DESCENDING**
Log Master reads log files starting with the last file that you specified and reads in reverse order. When ULOGORD is DESCENDING, you must list the DB2 log files from highest RBA/LRSN value to lowest. For example, three DB2 log files in sequence from the lowest RBA/LRSN value to highest are LOGFILE01, LOGFILE02, and LOGFILE03. If the ULOGORD value is DESCENDING, the correct order in the INPUT statement is as follows:

```
INPUT DB2LOG (LOGFILE03, LOGFILE02, LOGFILE01)
```

Log Master expects to read log files from the lowest RBA/ LRSN value to the highest. Therefore, if you specify a list of log files that is not in RBA/LRSN order after Log Master applies the ULOGORD setting, you might encounter errors.

**LOGSORT=WHENEVER**
Determines how Log Master will process DB2 log data sets from multiple members of a DB2 data sharing group. Values for LOGSORT are as follows:

**WHENEVER**
Directs Log Master to use the value of the LOGTAPES installation option to determine whether to sort the DB2 log data sets. If you specify this value, Log Master always uses the LOGTAPES value, regardless of the type of storage device on which the log data sets are stored. The LOGSORT option and the LOGTAPES option work together. For a description of their interaction, see “LOGSORT” on page 138. The default value is WHENEVER.

**NEVER**
Directs Log Master to avoid sorting the log. Log Master processes all log data sets concurrently. When this installation option is NEVER, Log Master ignores any LOGTAPES value and allocates as many units as it needs to perform a merge (not a sort) action. Depending on the number of data sharing members and the number of tape units in your environment, this value can cause contention for tape units.

**ALWAYS**
Directs Log Master to always sort the log. Log Master does not attempt to process multiple log data sets concurrently.
You can override the installation option value at runtime by specifying the LOGOPTS statement, LOGSORT syntax option (page 138).

**LOGTAPES=0**

Specifies the maximum number of tape units or DASD data sets that Log Master allocates for reading log files. LOGTAPES does not limit the number of physical tape devices that Log Master can use, only the number of tape units that Log Master allocates for log processing at any one time. The default value is zero.

Log Master allocates the value that you specify or a value equal to the number of members in the data sharing group, whichever value is less.

This installation option applies only in a data sharing environment. When the LOGSORT installation option is set to NEVER, Log Master ignores any LOGTAPES value. This option interacts with the LOGSORT value in other ways as well. For more information about how the options work together, see “LOGSORT” on page 138.

---

**NOTE**

LOGTAPES is ignored if DB2 log files to be read are DASD files, or the number of actual tape units is not greater than the LOGTAPES value in a subsystem with mixed tapes/DASD log files.

Values for LOGTAPES are as follows:

**nnnn**
Any numeric value between one and the total number of tape drives that are accessible to the system.

**0**
Log Master sets LOGTAPES to the number of members within the data sharing group. The default value is 0.

You can override the installation option value at runtime by specifying the LOGOPTS statement, LOGTAPES syntax option (page 139).

**UNITWAIT=YES**

Determines whether Log Master sets the z/OS allocation flags to direct the operating system to wait for allocation to occur in the event of an unavailable unit, volser, or data set, or immediately fail allocation.

Values for UNITWAIT are as follows:

**NO**
Directs Log Master to immediately fail allocation.
**YES**
Directs Log Master to wait for the required resources to become available. The default value is YES.

**DB2CAT=NO**

Defines whether Log Master processes log records that are created as a result of updates to DB2 catalog tables. Values for DB2CAT are as follows:

**NO**
Prevents Log Master from processing log records for the DB2 catalog (unless the job specifically sets the value of the DB2CAT keyword to YES in the LOGSCAN statement). For compatibility with earlier versions, the default value is NO.

**YES**
Allows Log Master to process log records for the DB2 catalog if they meet the specified filter criteria.

**NEVER**
Prevents Log Master from processing log records for the DB2 catalog.

When a job operates on objects in the DB2 catalog, Log Master performs row completion processing on related log records. The large number of log records that are related to the catalog can cause a job to perform more row completion and run longer than a job that does not read the DB2 catalog. If you frequently operate on DB2 catalog objects, you can improve performance by creating more frequent image copies of the catalog, or by defining tables in the DB2 catalog with Data Capture Changes (DCC). Table 3 lists some DB2 catalog table spaces and a recommended data capture setting for each. To improve performance of jobs that read the DB2 catalog, define the catalog tables within the table space with the recommended setting.

**Table 3  DB2 catalog table spaces and recommended data capture settings for tables (part 1 of 2)**

<table>
<thead>
<tr>
<th><strong>DB2 catalog table space</strong></th>
<th><strong>DCC / DCN setting for tables within table space</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNDB06.SYSCONTX</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSCOPY</td>
<td>optional</td>
</tr>
<tr>
<td>DSNDB06.SYSCOPY</td>
<td>optional</td>
</tr>
<tr>
<td>DSNDB06.SYSDBASE</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSDBAUT</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSDDF</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSGPAUT</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSGROUP</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSOBJ</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSPKAGE</td>
<td>optional</td>
</tr>
<tr>
<td>DSNDB06.SYSPLAN</td>
<td>optional</td>
</tr>
<tr>
<td>DSNDB06.SYSROLES</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSEQ</td>
<td>DCC</td>
</tr>
</tbody>
</table>
Table 3  DB2 catalog table spaces and recommended data capture settings for tables (part 2 of 2)

<table>
<thead>
<tr>
<th>DB2 catalog table space</th>
<th>DCC / DCN setting for tables within table space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNDB06.SYSSEQ2</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSSTATS</td>
<td>DCN</td>
</tr>
<tr>
<td>DSNDB06.SYSSTR</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSTARG</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSUSER</td>
<td>DCC</td>
</tr>
<tr>
<td>DSNDB06.SYSVIEWS</td>
<td>DCC</td>
</tr>
</tbody>
</table>

You can override the installation option value at runtime by specifying the LOGSCAN statement, DB2CATALOG syntax option (page 174).

**EXECMODE=CURRENT**

Determines whether Log Master processes log records for DB2 objects that have been dropped and possibly re-created. Values for EXECMODE are as follows:

**CURRENT**

Specifies that Log Master processes DB2 log records only for objects that are currently defined in the DB2 catalog.

**OVERTIME**

Specifies that Log Master can process DB2 log records for objects that are no longer defined in the DB2 catalog. For more information about processing in overtime mode, see “Overtime mode” on page 135, or the chapter on objects over time in the Log Master for DB2 User Guide.

You can override the installation option value at runtime by specifying the OPTION statement, EXECUTION MODE syntax option (page 121).

**FLTRMTHD=STATIC**

Determines how and when Log Master obtains DB2 catalog information (DBIDs, OBIDs, or PSIDs) for the DB2 objects that are named in a filter. Log Master reads the DB2 catalog to resolve the names of DB2 objects into numeric identifiers. Log Master can read the catalog either once during the initial analyze phase of processing for all objects, or repeatedly as it encounters each object in the DB2 log.

**STATIC**

Directs Log Master to obtain DB2 catalog information one time (during the analyze phase of processing). Log Master obtains information for all DB2 objects that the filter explicitly names. The default value is STATIC.

**DYNAMIC**

Directs Log Master to obtain DB2 catalog information dynamically, as it scans the DB2 log. Log Master obtains information for the DB2 objects that are present in the scanned log records, and selects log records for only the objects that are named in the filter.
For more information about performance considerations, see the OPTION statement, FILTER METHOD syntax option (page 123).

**URIDTHR=0**

Specifies the maximum number of data changes that Log Master processes within one unit of work, as represented by one unit of recovery identifier (URID). In this context, a data change is an insert, update, or delete action that affects one or multiple rows in a database. When a change affects multiple rows, Log Master counts one change for each affected row.

Use this installation option to limit the negative results that can occur when a URID contains an excessive number of data changes. Use this installation option only if you experience such negative results.

When Log Master scans a URID that contains an extremely large number of data changes, Log Master can require more memory than is available, and perform poorly or terminate abnormally. The point where these results occur depends on factors in your environment, such as the amount of memory that you allocate to Log Master or the size of your log records. Values for URIDTHR are as follows:

0  
Log Master processes any number of data changes, and does not check for a threshold value. The default value is 0.

nnnn  
Log Master checks the number of data changes within any single URID against the value that you enter. If the number of changes exceeds your threshold value, Log Master takes the following actions:

- issues a warning message
- ignores any remaining data changes within the current URID
- continues processing the next URID

If Log Master ignores data changes within a URID, it might not be able to complete all of the log records that it requires. When Log Master cannot complete all log records, it terminates with an error message.

**NOTE**

To help you detect URIDs that contain large numbers of changes, Log Master does not honor the URIDTHR installation option value when it generates an Object Activity Summary report.
**GENMDEL=**YES

Determines what output Log Master generates when it encounters the DB2 log records that reflect a LOAD REPLACE action. When a DB2 Load utility runs with the REPLACE option, the log contains records that Log Master can interpret as a “mass delete” action (similar to a DELETE statement with no WHERE clause or a TRUNCATE statement). This installation option controls whether Log Master includes the mass delete action in your generated output.

**YES**

Indicates that Log Master should include the mass delete action in the generated output. The default value is YES.

**NO**

Indicates that Log Master should not include the mass delete action.

Use this option to prevent Log Master from including a generated DELETE or TRUNCATE statement in some situations. For example, for a LOAD REPLACE LOG NO action, DB2 writes log records corresponding to the mass delete action, but does not write subsequent log records that Log Master can interpret as a “load” action (a series of INSERT statements). If you generate MIGRATE SQL in this situation and apply it to a target database, the generated statement removes all previous data from a table, but there are no generated INSERT statements to provide the loaded data. In some environments (such as with auditing or historical databases) the mass delete action is not appropriate.

You can override the installation option value at runtime by specifying the OPTION statement, GENERATE MASSDELETE syntax option (page 131).

**BYPASSLL=**NO

Determines whether Log Master includes or excludes certain database changes in Summary reports. When a DB2 Load utility is run with the LOG option set to YES, the DB2 log contains log records that reflect each page that the utility loads into a table space (“page format” log records). Log Master processes these log records to derive and report an INSERT action for each row that is loaded into each table. In some situations or environments, you might want to exclude these INSERT actions from your Summary reports to concentrate on more pertinent information.

Log Master ignores the value of this installation option when a log scan includes any form of output other than a Summary report.

**YES**

Indicates that Log Master should generate Summary reports that exclude the INSERT actions that result from a LOAD LOG YES action.

**NO**

Indicates that Log Master should generate Summary reports that include the INSERT actions that result from a LOAD LOG YES action. The default value is NO.
LOADFMT=LOGMAST

Specifies the format of load data files and load control files. Several formats are available to help you move log information to other platforms or database software. Values for LOADFMT are as follows:

**LOGMAST**
Indicates that Log Master formats load data files and load control files using its own internal format. The BMC Software LOADPLUS for DB2 product can also read a load file in this format. The default value is LOGMAST.

**UNLOADP**
Indicates that Log Master formats load data and load control files using the native format of the BMC Software UNLOAD PLUS® for DB2 product. For more information about the differences between the LOGMAST and ULOADP formats, see page 234.

**CSV**
Indicates that Log Master formats load data files and load control files using the Comma Separated Value (CSV) format. Individual fields in the load records are separated by commas.

**SDF**
Indicates that Log Master formats load data files and load control files using the Standard Definition Format (SDF) format. Individual fields in the load records are filled so that each field has the same position and length in all load records.

You can override the installation option value at runtime by specifying the LOGSCAN statement, LOAD FORMAT syntax option (page 234).

LOBALLOC=CYLINDERS

Specifies the units that Log Master uses to allocate LOB or XML VSAM files. Log Master creates these files to store data from LOB or XML columns in selected log records. The default value is CYLINDERS, which specifies cylinders. Use the TRACKS keyword to specify tracks.

You can override the installation option value at runtime by specifying the LOBOPTS statement CYLINDERS | TRACKS syntax option (page 145) for LOB data, or the XMLOPTS statement CYLINDERS | TRACKS syntax option (page 148) for XML data.

LOBDATA=

Specifies a valid Data Facility Storage Management System (DFSMS) data class that Log Master uses to allocate LOB or XML VSAM files. Log Master creates these files to store data from LOB or XML columns in selected log records. For more information about using this installation option with other installation options, see Table 2 on page 52.
Specify any valid DFSMS data class.

**NOTE**

When one of the following conditions exists, you must specify a value for LOBDATA, LOBSTOR, or LOBVOLS:

- You generate DDL in DB2 Version 10 new-function mode. (DB2 catalog changes require LOB processing to accommodate DDL for which text strings were moved to LOB columns.)
- You use Log Master to process LOB column data in any version of DB2.

If you specify a value for LOBVOLS, *do not* specify a value for LOBSTOR or LOBDATA.

You can override this installation option value at runtime by specifying the LOBOPTS statement DATACLAS syntax option (page 146) for LOB data, or the XMLOPTS statement DATACLAS syntax option (page 149) for XML data.

**LOBDUPDS=ERROR**

Determines how Log Master responds to a duplicate data set condition as it attempts to create a LOB or XML VSAM file. This option determines the severity of the message that Log Master displays, and whether Log Master continues processing.

**INFO**

Specifies that Log Master issues an informational message, overwrites the existing file, and continues processing.

**WARN**

Specifies that Log Master issues a warning message, overwrites the existing file, and continues processing.

**ERROR**

Specifies that Log Master issues an error message, saves any temporary LOB or XML VSAM files permanently to disk, and terminates processing. The default value is ERROR.

You can override the installation option value at runtime by specifying the LOBOPTS statement DUPLICATE DATASET syntax option (page 146) for LOB data, or the XMLOPTS statement DUPLICATE DATASET syntax option (page 149) for XML data.

**LOBLIMIT=10**

Determines the maximum number of LOB or XML VSAM files that Log Master can create to store the LOB or XML data of one LOB or XML column (or one partition of one LOB or XML column). Log Master allocates a new LOB or XML VSAM file only when it has filled the initial data set and all possible extents, but more LOB or XML
column data remains to be written. If the number of data sets required to store a LOB or XML column’s data exceeds this limit, Log Master issues an error message, saves any temporary LOB or XML VSAM files permanently to disk, and terminates processing.

Specify a number of data sets between 1 and 26. The default value is 10.

You can override the installation option value at runtime by specifying the LOBOPTS statement LOBLIMIT syntax option (page 147) for LOB data, or the XMLOPTS statement XMLLIMIT syntax option (page 150) for XML data.

**LOBMGMT**=

Specifies a valid DFSMS management class that Log Master uses to allocate LOB or XML VSAM files. Log Master creates these files to store data from LOB or XML columns in selected log records. This option has no default value. For more information about using this installation option with other installation options, see Table 2 on page 52.

Specify any valid DFSMS management class.

You can override the installation option value at runtime by specifying the LOBOPTS statement MGMTCLAS syntax option (page 146) for LOB data, or the XMLOPTS statement MGMTCLAS syntax option (page 149) for XML data.

**LOBPREF=&SYSUID..LOB**

Specifies a prefix that Log Master uses in the names of LOB VSAM files. Log Master constructs the data set name by using this prefix, the database identifier (DBID) of the LOB column, the page set identifier (PSID) of the column, and a sequentially incremented data set number. To avoid duplicate data sets during repeated log scans, consider using symbolic values such as &DATE. and &TIME. in the prefix that you define.

Values for LOBPREF are as follows:

- Any valid name up to 34 characters long.
- A symbolic substitution (such as &SYSUID).

**NOTE**

If you use a symbolic substitution, such as &SYSUID, enter &&SYSUID in the actual option macro to prevent the assembler from flagging the definition as an error.

You can override the installation option value at runtime by specifying the LOBOPTS statement, PREFIX syntax option (page 146).
**LOBSPACE=(10,10)**

Specifies the primary and secondary amounts of space that Log Master uses to allocate LOB or XML VSAM files. Log Master creates these files to store data from LOB or XML columns in selected log records. This installation option works with the LOBALLOC option. The default values of LOBSPACE are 10 and 10. Allocation amounts can be either tracks or cylinders.

Allocation amounts are site specific. However, if you allocate insufficient space to store all of the LOB or XML data in the log records that are selected by a log scan, your Log Master job can fail with an allocation error.

You can override the installation option value at runtime by specifying the LOBOPTS statement SPACE syntax option (page 145) for LOB data, or the XMLOPTS statement SPACE syntax option (page 148) for XML data.

**LOBSTOR=**

Specifies a valid DFSMS storage class that Log Master uses to allocate LOB or XML VSAM data sets. Log Master creates these files to store data from LOB or XML columns in selected log records. For more information about using this installation option with other installation options, see Table 2 on page 52.

Specify any valid DFSMS storage class.

**NOTE**

When one of the following conditions exists, you must specify a value for LOBDATA, LOBSTOR, or LOBVOLS:

- You generate DDL in DB2 Version 10 new-function mode. (DB2 catalog changes require LOB processing to accommodate DDL for which text strings were moved to LOB columns.)

- You use Log Master to process LOB column data in any version of DB2.

If you specify a value for LOBVOLS, do not specify a value for LOBSTOR or LOBDATA.

You can override the installation option value at runtime by specifying the LOBOPTS statement STORCLAS syntax option (page 146) for LOB data, or the XMLOPTS statement STORCLAS syntax option (page 149) for XML data.

**LOBVOLS=**

Specifies the volumes that Log Master uses to allocate LOB or XML VSAM files. Log Master creates these files to store data from LOB or XML columns in selected log records. Log Master uses the volumes that you enter in the VOLUMES parameter of a DEFINE CLUSTER command. For more information about using this installation option with other installation options, see Table 2 on page 52.
Specify any valid VOLSER as a value for LOBVOLS.

---

**NOTE**

When one of the following conditions exists, you must specify a value for LOBDATA, LOBSTOR, or LOBVOLS:

- You generate DDL in DB2 Version 10 new-function mode. (DB2 catalog changes require LOB processing to accommodate DDL for which text strings were moved to LOB columns.)

- You use Log Master to process LOB column data in any version of DB2.

If you specify a value for LOBVOLS, **do not** specify a value for LOBSTOR or LOBDATA.

You can override the installation option value at runtime by specifying the LOBOPTS statement VOLUME syntax option (page 146) for LOB data, or the XMLOPTS statement VOLUME syntax option (page 149) for XML data.

**XMLPREF=&SYSUID..XML**

Specifies a prefix that Log Master uses in the names of XML VSAM files. Log Master constructs the data set name by using this prefix, the database identifier (DBID) of the XML column, the page set identifier (PSID) of the column, and a sequentially incremented data set number. To avoid duplicate data sets during repeated log scans, consider using symbolic values such as &DATE. and &TIME. in the prefix that you define.

Values for XMLPREF are as follows:

- any valid name, up to 34 characters long
- a symbolic substitution (such as &SYSUID)

---

**NOTE**

If you use a symbolic substitution, such as &SYSUID, enter &&SYSUID in the actual option macro to prevent the assembler from flagging the definition as an error.

You can override the installation option value at runtime by specifying the XMLOPTS statement, PREFIX syntax option (page 149).

**DDLOBJ=NO**

Determines whether Log Master includes data definition language (DDL) information in all output logical log files. When this installation option is set to YES, Log Master writes objects representing the DDL activity that is selected by your filter into the logical log data file.
The DDL objects that Log Master writes in an output logical log file are not the same as the DDL statements that it writes in an output DDL (or SQL) file. A DDL object can represent multiple DDL statements, and a DDL statement can represent multiple insert, update, or delete actions against several tables in the DB2 catalog. A DDL object written by Log Master can be interpreted as input by either Log Master itself (using the LLOG keyword of the INPUT statement) or by the High-speed Apply Engine that is distributed with Log Master.

Values for DDLOBJ are as follows:

**YES**
Directs Log Master to include DDL information in output logical log files.

**NO**
Directs Log Master to exclude DDL information from output logical log files. The default value is NO.

You can override the installation option value at runtime by specifying the LOGSCAN statement, INCLUDE DDL syntax option (page 204).

**REBUILD=NO**

Determines the keyword that Log Master includes in the recovery syntax that it generates as part of an automated drop recovery job step. To accomplish a drop recovery, Log Master creates syntax for the BMC Software RECOVER PLUS for DB2 product. RECOVER PLUS supports two keywords for recovering indexes.

Values for REBUILD are as follows:

**NO**
Indicates that Log Master includes the RECOVER INDEX keyword in the RECOVER PLUS syntax. The default value is NO.

**YES**
Indicates that Log Master includes the REBUILD INDEX keyword in the RECOVER PLUS syntax.

**TAPES=(tape device name list)**

Specifies the tape device names of data sets that Log Master allocates for reading log files. Enter a list of valid tape device names and separate multiple names with commas. The default values are CART and TAPE.

**NOTE**
LOGTAPES is ignored if DB2 log files to be read are DASD files, or the number of actual tape units is not greater than the LOGTAPES value in a subsystem with mixed tapes/DASD log files.
ALPOFDSN=

Specifies the location of a data set that contains the Log Master product options file (POF). Log Master can read this optional file to obtain certain default values for the online interface. In the current version of Log Master, the POF supports default values that are related to output data set names or STEPLIB information (data set names on the Output Defaults, Automated Drop Recovery Defaults, or Generate JCL panels). This option has no default value.

The value for ALPOFDSN can be a data set or a member of a partitioned data set. Log Master does not require a specific name for the data set or member.

You can specify multiple POF data sets, such as one per DB2 subsystem or data sharing group, by using the following symbolic variables:

- &SSID (subsystem ID)
- &GATN (group attach name)

Use this file to establish subsystem-wide default values that apply to all users of the Log Master online interface. Many environments use the POF to match default data set names with site naming conventions. For the default values currently supported in the POF, see Figure 1 on page 76.

The following considerations apply to the POF:

- Log Master also stores online interface defaults in your ISPF profile. If you change a default value during an online session, your change will be preserved for any future sessions under your TSO ID. The order of precedence for online defaults is ISPF profile value, POF value, and product base default values.

- For the current version, the Log Master POF does not apply to any other products from BMC Software.

- The POF is not generated during installation. To create and use a POF, you must complete the following tasks:
  
  — Set the ALPOFDSN installation option equal to your desired POF data set name.

  — Edit the job that is stored in member HLQ.DBSAMP(ALPPOFX) to insert your POF name. (The HLQ value represents a qualifier that your environment assigns during installation.)

  — Submit the job. Log Master creates a POF with Log Master base default values.

  — Edit the POF to set your desired default values. (The POF stores default values in the form of XML elements and attribute values.)
The next time you run the Log Master online interface, POF default values will appear (unless your TSO profile already contains a default value for a particular field). To remove the TSO default value for a field, run the online interface, delete the content in that field, and exit the interface.

Figure 1 shows a sample POF with all currently supported POF default values.

**Figure 1  Sample POF with supported default values**

```xml
<BMCOptions>
  <OptionSet Product="ALP">
    <!-- Log Master Options... -->
    <Option Name="APFONLIN" Value="YES" />
  </OptionSet>
</BMCOptions>
```

**APFONLIN=YES**

Determines whether the online interface runs as an authorized TSO command, and whether Log Master checks to ensure that it has proper APF authorization. When the Log Master online interface runs as an authorized TSO command, Log Master can dynamically obtain the data set name of the BSDS from DB2. The default value is YES. Values for APFONLIN are as follows:
**YES**
Indicates that the online interface runs as an authorized TSO program. Log Master verifies that it has the proper APF authorization, and can dynamically obtain the name of the BSDS from DB2. For more information, see “APF authorization for the online interface” on page 30.

**NO**
Indicates that the online interface does not run as an authorized TSO program. If you set APFONLIN to NO, Log Master cannot dynamically obtain the name of the BSDS from DB2. You must enter the name and location of the BSDS on the Product Options panel (option 41 on the Main Menu).

**DBCS=NO**
Directs Log Master to use only uppercase characters in the online interface and the product-generated portions of reports (for example, in report headings and field labels). BMC Software provides this installation option for environments that display or print double-byte characters. This option prevents Log Master from inadvertently displaying or printing lowercase characters as incorrect double-byte characters.

**NO**
Log Master displays and prints both uppercase and lowercase characters. The default value is NO.

**YES**
Log Master displays and prints only uppercase characters.

**OSRCHCNT=3**
Determines how Log Master responds if the user running the online interface does not enter a log file name when specifying DB2 log data sets. This installation option applies only to the online interface, and only in a data sharing environment.

This option determines how many members of the data sharing group Log Master searches for eligible log files. Depending on the OSRCHCNT value, Log Master displays a list of eligible log file names for each member of the data sharing group. The user running the online interface can then select log files. The default value is 3. Use the following guidelines when specifying a value for OSRCHCNT:

- If the value is less than or equal to the number of members in the data sharing group, Log Master lists the eligible log file names for each data sharing member.
- If the value is greater than the number of members in the data sharing group, Log Master does not list the eligible log file names.
TRCHARS=>>&gt;

Specifies the character string that Log Master uses to indicate truncated DB2 object names within the online interface. When DB2 object names are longer than the fields that you use to view or edit them, Log Master initially displays truncated versions of the names. These characters identify truncated names and mark the position of the truncated portion. To display the complete DB2 object names, position the cursor in a field and press F4.

Enter a string up to 8 characters long. The default value is &gt;>.

TRPOS=END

Defines which part of DB2 object names Log Master truncates for display within the online interface. When DB2 object names are longer than the fields that you use to view or edit them, Log Master initially displays truncated versions of the names. Use this installation option to include the meaningful or unique parts of DB2 object names in the truncated names, depending on the object naming standards in your environment. Values for TRPOS are as follows:

BEGIN
Log Master truncates the start of DB2 object names and displays the ending characters of the names.

MIDDLE
Log Master truncates the middle part of names and displays both the start and end of DB2 object names.

END
Log Master truncates the end of DB2 object names and displays the starting characters of DB2 object names. The default value is END.

ERLYRCL=YES

Directs Log Master to issue requests to recall archived data sets before it needs the data sets for log processing. This action decreases delays in log processing by giving storage management software in your environment time to retrieve the required data sets. The default value is YES. You must set ERLYRCL to YES to enable any of the other Log Master early recall installation options (such as MIGRATE or DASDDSNS).

The SMSTASKS installation option limits the number of early recall requests that Log Master issues at any one time. Before you enable early recall of archived data sets, evaluate the settings of the SMSTASKS, MIGRATE, and DASDDSNS options to ensure that they are appropriate for your environment.

The Log Master early recall feature works with most storage management software. (The related data set migration feature can migrate only data sets that are managed by the IBM product DFSMShsm.) Values for ERLYRCL are as follows:
NO
Prevents Log Master from issuing early recall requests.

YES
Directs Log Master to issue early recall requests.

You can override the installation option value at runtime by specifying the OPTION statement, EARLY RECALL syntax option (page 129).

SMSTASKS=0

Limits the number of early recall subtasks that Log Master creates. Use this installation option to optimize performance when log processing requires large numbers of migrated files.

Enter a maximum number of subtasks. To allow Log Master to determine the number of subtasks, enter 0 (the default value). When you specify 0, Log Master sets the SMSTASKS value as follows:

- In a non–data sharing environment, Log Master sets SMSTASKS to 1.
- In a data sharing environment, Log Master sets SMSTASKS equal to the number of data sharing members.

You can override the installation option value at runtime by specifying the OPTION statement, SMSTASKS syntax option (page 129).

MTPRMBR=NO

Specifies the scope of the SMSTASKS value. This installation option applies only in a data sharing environment. Values for MTPRMBR are as follows:

YES
Indicates that your SMSTASKS value applies to each member of a data sharing environment. If you specify YES, and the SMSTASKS value is currently set to zero, and Log Master
- runs in a non–data sharing environment, Log Master sets SMSTASKS to one
- runs in a data sharing environment, Log Master sets SMSTASKS equal to the number of members in the data sharing group

NO
Indicates that your SMSTASKS value applies to the entire data sharing group. The default value is NO.

You can override the installation option value at runtime by specifying the OPTION statement, PER MEMBER syntax option (page 129).
Installation option descriptions

**MIGRATE=**YES

Directs Log Master to request your storage management software to migrate any recalled data sets to their original status. Log Master issues migrate requests at the end of a log scan or when a job requires a greater number of data sets than the DASDDSNS value. Values for MIGRATE are as follows:

**YES**
Directs Log Master to issue migrate requests. The default value is YES.

**NO**
Prevents Log Master from issuing migrate requests (unless the DASDDSNS value is exceeded).

The DASDDSNS value defines limits for this installation option. If a job requires more data sets than the DASDDSNS value permits, Log Master migrates data sets even when this installation option is set to NO.

Log Master always attempts to migrate the data set back to its original migration level. The Log Master data set migration feature can migrate only data sets that are managed by the IBM product DFSMShsm. (The related early recall feature works with most storage management software.)

Before you enable migration of recalled data sets, evaluate the settings of the SMSTASKS, ERLYRCL, and DASDDSNS options to ensure that they are appropriate for your environment.

You can override the installation option value at runtime by specifying the OPTION statement, MIGRATE syntax option (page 130).

**MIGRWAIT=**NO

Determines whether Log Master terminates at the end of processing, or waits for data set migration requests to complete. Values for MIGRWAIT are as follows:

**YES**
Indicates that your Log Master job does not terminate until all data set migration requests are complete.

**NO**
Indicates that your Log Master job terminates at the end of processing, regardless of the status of any data set migration requests. The default value is NO.

You can override the installation option value at runtime by specifying the OPTION statement, WAIT syntax option (page 130).
**DASDDSNS=10**

Determines the maximum number of recalled data sets that Log Master maintains on DASD at one time. Use this installation option to minimize DASD requirements for large numbers of archived files. Enter a maximum number of data sets. To avoid imposing a limit, enter zero.

To honor the DASDDSNS value, Log Master issues requests to migrate any recalled data sets to their original migration status and level. If a job requires more data sets than this value, Log Master issues requests to migrate data sets even if the value of the MIGRATE keyword is NO.

Depending on how quickly your environment processes migrate requests, for short periods of time, the number of recalled data sets on DASD might be greater than the DASDDSNS value. Log Master can migrate only data sets that are managed by the IBM product DFSMShsm. If DFSMShsm is not available in your environment, Log Master might not be able to honor the DASDDSNS limit value.

You can override the installation option value at runtime by specifying the OPTION statement, DASD DATASETS syntax option (page 131).

**DDPRMBR=YES**

Specifies the scope of the DASDDSNS value. This installation option applies only in a data sharing environment. Values for DDPRMBR are as follows:

**YES**
Indicates that the value is a maximum limit for each member of a data sharing environment. The default value is YES.

**NO**
Indicates that the value is a maximum limit for the entire data sharing group.

You can override the installation option value at runtime by specifying the OPTION statement, PER MEMBER syntax option (page 131).

**TZRULE=NONE**

Defines the rule to determine when DST begins and ends, enabling Log Master to calculate the dates and times for the changes each year.

This installation option enables Log Master to adjust day and time values when you run Log Master after a DST change, but scan log records that are from a period before the time change, which is called scanning across a DST boundary.

The default value, NONE, indicates that your locale does not observe DST or that you do not want Log Master to make adjustments for DST. To enable DST adjustment, you must define day and time values as a character string in the following format, including quotation marks:
‘startDayRule/time,endDayRule/time’

**Specifying TZRULE**

This section describes valid formats for specifying TZRULE. In all formats, each time field describes the local time in effect when the change from one time to the other occurs. Enter the time in the format `hh[:mm[:ss]]`. The value is always interpreted as a decimal number. The hour (`hh`) is required and must be a value between 0 and 23. The minutes (`mm`) and seconds (`ss`) are optional and can be a value between 0 and 59.

- **M m.n.d**

  This is the standard format for most countries. Using this format, specify the month, week, and day of the week on which DST begins and ends, as follows:

  — The variable `m` represents the month; valid values are 1 through 12 (January through December).

  — The variable `n` represents the week of the month in which the day `d` occurs; valid values are 1 through 5. Week 1 is the first week in which day `d` occurs, and week 5 specifies the last day in the month.

  — The variable `d` represents the day; valid values are 0 through 6 (Sunday through Saturday).

  In the following example, DST begins on the second Sunday of March at 2:00 a.m., and ends on the first Sunday of November at 2:00 a.m.:

  ```
  TZRULE='M3.2.0/2,M11.1.0/2'
  ```

  The default conversion time is 2:00 a.m.; therefore, the following example, which omits the time specification, is also valid:

  ```
  TZRULE='M3.2.0,M11.1.0'
  ```

- **Jn**

  Specify the Julian day `n`, where `n` is a value from 1 through 365. This format does not account for leap days; that is, in all years, including leap years, February 28 is day 59 and March 1 is day 60. You cannot explicitly refer to February 29.

  In the following example, DST begins on April 1 at 12:34:56 p.m., and ends on November 1 at 2:34:56 a.m.:

  ```
  TZRULE='J91/12:34:56,J305/2:34:56'
  ```
■ \( n \)

Specify the zero-based Julian day \( n \), where \( n \) is a value from 0 through 365. In this format, leap days are counted, and you can refer to February 29.

In the following example, DST begins on April 1 (leap years), or April 2 (nonleap years) at 12:34:56 p.m., and ends on November 1 at 2:34:56 a.m.:

```
TZRULE='91/12:34:56,305/2:34:56'
```

**Considerations for DST adjustment**

To understand DST adjustment, remember that Log Master uses the operating system to translate date and time values into the operating system’s store clock format. The operating system uses an offset based on your time zone to translate the values. For example: assume that you are in the northern hemisphere, you run Log Master in May (after the change to DST), and you define a scan range from 15:00 to 16:00 on February 14 (before the time change).

■ If you set TZRULE to enable adjustment, Log Master knows that the time zone offset in May is *different* than the time zone offset that was used when DB2 wrote the log records in February (the difference is one hour). Log Master adjusts for this difference, selects log records from 15:00 to 16:00 on February 14, and prints the time values of those records in reports starting with 15:nn.

■ If you do not set TZRULE to enable adjustment, Log Master uses the time zone offset for May. Because the offset was different in February, Log Master selects log records from 14:00 to 15:00 on February 14, and prints the time value of those log records in reports starting with 15:nn. The discrepancy in the time zone offset creates a discrepancy between the log records that you request and the log records that Log Master provides in output.

Be aware of the following additional points regarding DST adjustment and the TZRULE installation option:

■ Log Master provides adjustment because the operating system does not account for changes in the time zone offset that have occurred in the past. The operating system always uses the current time zone offset when it converts a date and time value into store clock format.

■ By default, adjustment is *not* enabled. To enable it, set the TZRULE installation option value.

■ The options migration feature of the Installation System cannot carry over DST installation option values from a previous release of Log Master. You must provide a value for TZRULE if you want to enable DST adjustment.
You might not need DST adjustment. Without adjustment, a difference of one hour occurs when you scan across a DST boundary. If you define scan ranges in terms of days or weeks, the one-hour difference might not be significant.

Log Master adjusts only *date and time values* that you enter as you define a scan range. The discrepancies do not occur (and Log Master does not adjust values) when you define a scan range with RBA/LRSN values or log mark names.

Adjustment can compensate for one-hour changes for DST if your environment adjusts the operating system’s clock by exactly one hour.

If your environment adjusts the operating system’s clock in increments *other than* one hour, and you do *not* set the option to enable automatic adjustment, Log Master might experience discrepancies in selecting log records. You can avoid the discrepancies by using log mark names or RBA/LRSN values to define your scan range. If you are aware of the time changes in your environment, you can also adjust accordingly the date and time values that you enter.

Adjustment affects only how Log Master handles the standard timestamp values that DB2 stores in all log records (using the operating system’s store clock format). Adjustment *does not affect* any DATE, TIME, or TIMESTAMP columns that you define in your database, nor does it affect any column-level filtering.

---

**ZIIP=ENABLED**

Controls whether Log Master attempts to use IBM® System z® Integrated Information Processors (zIIPs). Log Master 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 Log Master jobs. To enable and use zIIP processing with Log Master, you must

- have an installed and authorized version of the EXTENDED BUFFER MANAGER (XBM) product or the SNAPSHOT UPGRADE FEATURE (SUF) technology
- start and maintain an XBM subsystem in your environment
- have a zIIP available in your environment

For more information about the XBM component that enables the use of zIIPs, see the XBM technical documentation.

**ENABLED**

If a zIIP is available, Log Master attempts to offload eligible processing to the zIIP. If the zIIP is busy or not available, normal processing continues on a general-purpose processor. The default value is ENABLED.

**DISABLED**

Log Master will not attempt to use zIIP processing.
You can override the installation option value at runtime by specifying the OPTION statement, zIIP syntax option (page 135).

**LOCCPSEL=(FC,LP,LB)**

When running in a local site, specifies the order in which Log Master uses image copy resources for both completion processing and compression dictionary access for data decompression. The default sequence is IBM FlashCopy (FC), local primary (LP), and local secondary (LB). You can specify one to five values from FC, LP, LB, RP (remote primary), and RB (remote secondary) in the sequence in any order.

**NOTE**
The image copy resources are registered in SYSCOPY and BMCXCOPY. FlashCopies are registered in SYSCOPY and are associated with the IBM DB2 COPY utility. Only one FC exists for a given table space/RBA.

**REMCPSEL==(RP,RB,FC)**

When running in a remote site, specifies the order in which Log Master uses image copy resources for both completion processing and compression dictionary access for data decompression. The default sequence is remote primary (RP), remote secondary (RB), and IBM FlashCopy (FC). You can specify one to five values from FC, LP (local primary), LB (local secondary), RP, and RB in the sequence in any order.

**NOTE**
The image copy resources are registered in SYSCOPY and BMCXCOPY. FlashCopies are registered in SYSCOPY and are associated with the IBM DB2 COPY utility. Only one FC exists for a given table space/RBA.

**DYNALOC installation option**

The DYNALOC installation option of BMCSORT provides information for dynamically allocating SORTWK data sets. BMCSORT deallocates these data sets at the end of each sort. The content of the $AUPSMAC macro in $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
DYNAMIC ALLOC OPTIONS FOR SORT
```

The values that you specify in this macro apply to all invocations of BMCSORT. BMCSORT uses the same options module for all BMC products that invoke BMCSORT.
Table 4 on page 86 describes each parameter of the DYNALOC option. These parameters are positional. The values that you specify for these parameters should correspond to your site’s standards for any system sort routine.

BMCSORT overrides the values that you supplied if BMCSORT determines that it can complete sorting more efficiently than the specified values allow. An invoking product’s options might also override the BMCSORT options values that you specify when one of the following conditions exists:

- The values in the invoking product’s dynamic allocation installation options or corresponding command options conflict with the values that you specify.
- You turn on BMCSORT SORTWK dynamic allocation from the product that invokes BMCSORT, and you specify OFF for the position 3 parameter.

BMCSORT dynamically allocates SORTWK files as necessary.

### Table 4  DYNALOC parameters (part 1 of 2)

<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 Log Master for DB2 should dynamically allocate SORTWK data sets. This parameter applies only when the Data Facility Storage Management System (DFSMS) product from IBM is not installed or is not active for temporary DASD work data sets. If DFSMS is active, use the SC parameter.</td>
<td>SYSDA</td>
<td>Use a unit name up to 8 characters.</td>
</tr>
<tr>
<td>position 2</td>
<td>Do not change this value. Log Master for DB2 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>
<tr>
<td>position 3</td>
<td>This parameter tells Log Master for DB2 whether to dynamically allocate SORTWK files. Note: BMC recommends that you not change this value.</td>
<td>ON</td>
<td>ON dynamically allocates SORTWK. OFF does not dynamically allocate SORTWK.</td>
</tr>
<tr>
<td>position 4</td>
<td>Do not change this value. Log Master for DB2 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. Log Master for DB2 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>
</tbody>
</table>
### Table 4  DYNALOC parameters (part 2 of 2)

<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 6</td>
<td>Do not change this value. Log Master for DB2 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>
</tbody>
</table>
| position 7                 | This parameter specifies the DASD type with the smallest track capacity that a dynamically allocated SORTWK data set might encounter at your site.                                                            | 3390         | 3380, track capacity of 47968  
3390, track capacity of 56664  
9345, track capacity of 46456 |
| SC                         | This parameter specifies the name of the DFSMS storage class from which to dynamically allocate SORTWK. If DFSMS is active and you do not specify a value for this parameter, Log Master for DB2 uses the value from the first DYNALOC parameter. | blank        | Use any valid DFSMS storage class.                                                                |
| RETRY                      | This parameter specifies how you want Log Master for DB2 to handle retry attempts for SORTWK dynamic allocation:                                                                                           | (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 Log Master for DB2 to retry the request.  
0 through 15 for the second subparameter  
0 indicates that you do not want Log Master for DB2 to retry the request. |

---

### Sample files and other information

As you install Log Master, the Installation System generates a job that copies information from the distribution media and creates the Log Master sample library, `HLQ.DBSAMP`.
The sample library contains sample JCL files, sample programs, and other information. To determine the files that are included in the library, view the member named HLQ.DBSAMP(ALP$$IX).

Log Master provides also sample report templates for Summary, Audit, and Detail reports with control breaks on Table Name and Create Timestamp. The control break header and trailer information includes the following identifying information:

- table creator
- table name
- DBID
- OBID
- PSID
- create and drop RBA/LRSN
- create and drop timestamp

The report templates are downloaded during installation and are located in the Log Master product-specific XML library (ALPXML). They are also in a BMC-specific XML library (BMCXML). A directory file, ALPRT$$$.XML, is included that lists the sample report template names along with a brief description of the report. The report template names conform the following naming convention:

ALP + RT + Report Type code + Sequence ID.xml

The Report Type codes are shown in Table 5, and the Sequence ID is a 2-character sequence ID (01-99).

Table 5  Report template codes

<table>
<thead>
<tr>
<th>Report Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit</td>
<td>A</td>
</tr>
<tr>
<td>BIC</td>
<td>B</td>
</tr>
<tr>
<td>Commit</td>
<td>C</td>
</tr>
<tr>
<td>Detail</td>
<td>D</td>
</tr>
<tr>
<td>Catalog Activity</td>
<td>E</td>
</tr>
<tr>
<td>Open Transaction</td>
<td>F</td>
</tr>
<tr>
<td>DCC</td>
<td>G</td>
</tr>
<tr>
<td>Drop Recovery</td>
<td>H</td>
</tr>
<tr>
<td>Image Copy</td>
<td>I</td>
</tr>
<tr>
<td>Object Activity</td>
<td>J</td>
</tr>
<tr>
<td>Command</td>
<td>K</td>
</tr>
<tr>
<td>Log Bytes</td>
<td>L</td>
</tr>
<tr>
<td>BIC Summary</td>
<td>M</td>
</tr>
<tr>
<td>Quiet Point</td>
<td>Q</td>
</tr>
</tbody>
</table>
As you implement new DB2 versions, you must migrate the DB2 catalog from compatibility mode through enabling-new-function mode to new-function mode. Log Master runs in all catalog modes, with the following limitations:

- In enabling-new-function mode, you cannot read log records of changes to the DB2 catalog. For example, the following actions are not available in this mode because they require Log Master to read DB2 catalog log records:
  - generating a DDL output file
  - running in overtime mode and encountering DB2 catalog activity
  - generating a Catalog Activity report
  - updating the Repository with object structure definitions (REPOS UPDATE)

Functions that do not require the product to read DB2 catalog log records are available in enabling-new-function mode.

- After the DB2 catalog is migrated to new-function mode, you cannot read DB2 catalog log records that were created in compatibility mode before the migration. If you attempt to read such log records, the product can terminate abnormally or generate invalid data based on the log records.

These limitations apply only to jobs that read log records of DB2 catalog changes. If your jobs do not read log records of changes to the DB2 catalog, you can run Log Master for DB2 normally, regardless of the DB2 catalog mode.

To ensure optimal Log Master processing as you migrate the DB2 catalog, BMC Software makes the following recommendations:

- If your Log Master jobs read DB2 catalog log records, limit the amount of time that you spend in enabling-new-function mode.

- Execute and successfully complete all jobs that read DB2 catalog log records before you leave compatibility mode. This practice is particularly important for any ongoing jobs.

---

**Table 5 Report template codes**

<table>
<thead>
<tr>
<th>Report Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rollback</td>
<td>R</td>
</tr>
<tr>
<td>Summary</td>
<td>S</td>
</tr>
</tbody>
</table>
Before you run CATMAINT, stop any DDL or DSND06 extraction processes. Resume the processes after running CATMAINT, but skip the CATMAINT log records.

CATMAINT updates the DB2 catalog differently than normal DDL processing. Because the order does not match its DDL generation expectations, Log Master might misinterpret the logging by CATMAINT. Using Log Master to process CATMAINT log records might result in errors or abends.

After you enter new-function mode, use Log Master syntax to reset the start point of any ongoing jobs that read DB2 catalog log records. For more information, see the RESET keyword of the LOGSCAN statement on page 178.

Improving Log Master performance

Log Master requires information from the following types of DB2 data sets, in addition to the logs:

- image copies
- bootstrap data sets (BSDS)
- table space data sets
- Repository tables
- several catalog tables

To enable Log Master to execute with the greatest efficiency, observe the following practices (listed in order of highest to lowest importance):

1. Do not delete records from SYSIBM.SYSCOPY (using the MODIFY utility) or the BSDS (using the DSNJU003 utility) until the records are older than any archive logs that are to be scanned. For example, if you need to scan a log that is one month old, Log Master requires the log, image copies, and SYSCOPY information for one month. If the SYSCOPY information is missing, Log Master cannot perform row completion processing (acquiring complete before and after images of a row update record). Decompression processing might fail if you are not using DATA CAPTURE CHANGES and KEEPDICTIONARY. If BSDS information is missing for archive logs, Log Master cannot access those archive logs.

2. Use DATA CAPTURE CHANGES for the tables of interest for Log Master processing. If you use DATA CAPTURE NONE, for Log Master to provide any output that contains detailed column data, such as SQL, LOAD, logical log, and detailed reports, Log Master might be required to perform multiple log scans across various ranges, and read image copy data sets. By altering on DATA CAPTURE CHANGES, you can avoid the overhead of row completion processing for partially logged updates.
3. Keep image copies on DASD until they are no longer required for Log Master processing. Reading tape data sets can be time consuming. If multiple image copies are stacked on tape, Log Master attempts to optimize for completion processing, but not for compression dictionary acquisition.

4. Regularly run REPOS UPDATE with INCLUDE DICTIONARY to store the following types of information in the Repository:

- table schema information (used for DROP RECOVERY processing)
- compression dictionaries (used to avoid image copy processing)

For more information, see “REPOS UPDATE” on page 167.

5. Regularly run REPOS DELETE to purge obsolete data from the Log Master Repository. For more information, see “REPOS DELETE” on page 171.

6. If Log Master reloads compression dictionaries during processing (see messages BMC097677I at the end of ALPPRINT), increase the dictionary space allocation. To do so, change the value of the DICTSPC installation option (page 55), or use the OPTIONS statement, DICTIONARYSPACE syntax option (page 128).

7. In a data sharing environment, if DB2 processing for your tables of interest occurs only on a small subset of the data sharing members, use the OPTION statement, USELGRNG YES syntax option (page 127) to avoid scanning the logs of all members of the data sharing group. Consider experimenting with this option if you are unsure about the nature of the DB2 processing.

**Solution integration**

Log Master for DB2 (including the High-speed Apply Engine) is a component of the Recovery Management for DB2 solution. The Recovery Management for DB2 solution integrates the features of the following BMC Software products:

- RECOVERY MANAGER for DB2
- RECOVER PLUS for DB2
- R+/CHANGE ACCUM
- COPY PLUS for DB2
- Log Master for DB2
- SNAPSHOT UPGRADE FEATURE
Customers who use this solution benefit from all of the features of these individual products, as well as additional features available when one Recovery Management component can rely on the presence of all others. For more information, see the *Recovery Management for DB2 User Guide* document.
Building and running Log Master jobs

This chapter contains the following topics:

Building Log Master jobs ................................................................. 94
Specifying the JOB statement ....................................................... 96
Specifying the EXEC statement .................................................... 97
Specifying the STEPLIB DD or JOBLIB DD statements ................. 100
Specifying the DD statements ..................................................... 100
Defining sort work data sets .......................................................... 102
  Controlling allocation with SORTOPTS keywords ..................... 102
  Controlling allocation with DD statements ............................... 103
Specifying Log Master syntax ....................................................... 106
  SYSIN syntax in-stream in JCL .................................................. 107
  SYSIN syntax in a separate data set ......................................... 107
  Comments in SYSIN syntax ...................................................... 108
  Long names in Log Master syntax ............................................ 109
Running Log Master jobs ............................................................... 109
Log Master termination processing .............................................. 110
Cleaning up after a Log Master job ............................................. 112
Building Log Master jobs

To build a job that uses Log Master for DB2, you must create the appropriate job control language (JCL) statements. You can use either of the following methods to create the JCL:

- Use the Log Master online interface to create the JCL.

  If you have limited experience with Log Master or JCL, you can use the online interface to complete the steps needed to create the appropriate JCL for the tasks that you want to accomplish. For information about using the Log Master online interface, see the Log Master for DB2 User Guide.

  Use the online interface when you first create a special type of Log Master job that contains an ongoing log scan. The online interface assigns an ongoing handle ID and updates the Log Master Repository with important information. For more information about ongoing jobs, see “Ongoing log scans” on page 179.

- Create your own JCL.

  If you are comfortable creating your own JCL, you can bypass the online interface. For example, you might prefer to edit the JCL of an existing Log Master job. However, even if you are experienced with JCL, you might use the online interface to generate JCL for a task that you have not performed before. This chapter summarizes the process of using the batch syntax to run Log Master jobs.

To build a Log Master job, create JCL with the following elements:

- a JOB statement

- an EXEC statement that includes Log Master parameters

  For information on the parameters of the EXEC statement, see “Specifying the EXEC statement” on page 97.

- Data definition (DD) statements to specify the following required and optional data sets:

  — Log Master load libraries, for example HLQ.DBLINK, HLQ.XXLINK, HLQ.BBLINK, where HLQ is a qualifier that your environment assigns during installation.

  — DB2 load library that Log Master uses

  — ALPPRINT output data set to contain product messages

  — SYSOUT output data set to contain task-level messages
— *(optional)* ALPDUMP and SYSUDUMP/SYSMDUMP output data sets to contain diagnostic and troubleshooting information

— *(optional)* sort work data sets to enable sorting

Declare these data sets if dynamic allocation of sort work data sets is not appropriate for your environment.

- SYSIN input that contains the Log Master statements and keywords

*Figure 2* shows sample JCL for a basic Log Master job.

**Figure 2  JCL for a simple Log Master for DB2 job (part 1 of 2)**

```jcl
//ALPUSER  JOB(0000), 'LOGSCAN'
//******************************************************************************
/* LOG MASTER FOR DB2 V10.1.00 */
/* DSN: 'ALPUSER.JCL(WKMIGR)' */
/* GENERATED BY USER: ALPUSER */
/* ON DATE: 2012/12/18 */
/* AT TIME: 13:49 */
******************************************************************************
//LOGMSTR EXEC PGM=ALPMAIN,
// PARM='DBAT',
// REGION=0M
//STEPLIB DD DISP=SHR, DSN=product.libraries
// DD DISP=SHR, DSN=DB2.DSNEXIT
// DD DISP=SHR, DSN=DB2.DSNLOAD
//ALPPRINT DD  SYSOUT=*  
//SQLPRINT DD  SYSOUT=*  
//SYSOUT DD  SYSOUT=*  
//ALPDUMP DD  SYSOUT=*  
//SYSTERM DD  SYSOUT=*  
//SYSUDUMP DD  SYSOUT=*  
//SYSIN DD *, DLM=##
/* ALPUSER.$$WORKID0009 */
/* LOGSCAN MIGR SQL */
OPTION
FILTERREL AND EXECUTION MODE CURRENT

WORKID ALPUSER.$$WORKID0009
DESC "LOGSCAN - MIGRATE SQL"

LOGSCAN
SQL
MIGRATE
  DATASET &SYSUID..D&DATE..T&TIME..MIGRATE.SQL NEW
  CYLINDERS SPACE(1,1) UNIT(SYSDA) RELEASE
  TEMPLATE &SYSUID..D&DATE..T&TIME..MIGRATE.TEMPLATE NEW
  CYLINDERS SPACE(1,1) UNIT(SYSDA) RELEASE
  INCLUDE RI NO
```
Specifying the JOB statement

The JOB statement starts with a job name, and includes standard JOB statement parameters such as accounting information and a name that identifies the run. For example:

```
//LMDB2RUN JOBacctngInfo,prgmmrInfo,CLASS=A
// REGION=0M,MSGLEVEL=(1,1)
// USER=DB2DBA,PASSWORD=password
```

If you generate a job through the online interface, Log Master saves this information with your ISPF profile and restores it whenever you log on with the same TSO user ID.

**REGION parameter**

The REGION parameter in JCL, in conjunction with the System Management Facility (SMF) parameter MEMLIMIT, establishes the maximum limit on the memory that a job can use. Include a REGION parameter in your JOB statement if you need to specify an amount of memory available to the job. If you omit the REGION parameter from the JOB statement, you can include it in the EXEC statement.

BMC Software recommends that you specify one of the following values for the REGION parameter:

- 0 (as 0M or 0K)

  This value automatically allocates the maximum amount of memory that is allowed for a job in your environment.

- at least 50M

  In some environments, individual users are not permitted to set REGION to 0M, or entering a value of 0M results in an amount of allocated memory less than 50M.
In either case, ensure that the operating system allocates at least 50M of virtual storage to your Log Master jobs. You might need to specify a value greater than 50M depending on which types of output your job creates, how many different types of output it creates, or the number of log records that the job scans. For more information about your jobs and the REGION parameter, see the chapter on expert information in the *Log Master for DB2 User Guide*.

**NOTE**

If all of the following conditions exist, Log Master might terminate abnormally when it attempts to access virtual storage above the bar:

- You are using DB2 Version 9.1 or later.
- You do not specify a value of 0 for the REGION parameter.
- Your environment does not change the default value for the MEMLIMIT parameter in SMF.

For more information, see “Setting the MEMLIMIT parameter” on page 26.

### Specifying the EXEC statement

The EXEC statement specifies the load module (ALPMAIN) for Log Master. This statement includes the parameters described in this section and has the following format:

```bash
//step EXEC PGM=ALPMAIN,REGION=0M,
   //PARM='ssid,,,MSGLEVEL(msgLevel),ALPOPTS(optsMod)'
```

A Log Master EXEC statement includes the following parameters. The first two parameters are positional; you must specify them in the order shown. The remaining parameters are not positional.

- *(optional)* DB2 subsystem ID (*ssid*) or group attachment name for data sharing
  
  For more information, see “DB2 subsystem identification (*ssid*)” on page 98.

- a utility ID (*utilityID*), reserved for future use
  
  The utility ID is not used, but you must honor its position.

- *(optional)* message level parameter (*msgLevel*)

- *(optional)* alternate installation options module (*optsMod*) for the job
The following example shows two EXEC statements. The first statement specifies an SSID value and an alternate installation options module, but not the message level. The second statement specifies an SSID value and a message level, but not an alternate installation options module.

```
//stepname EXEC PGM=ALPMAIN,REGION=0M, 
// PARM='DB2A,,ALPOPTS(MYOPTS)'

//stepname EXEC PGM=ALPMAIN,REGION=0M, 
// PARM='DB2A,,MSGLEVEL(2)'
```

### DB2 subsystem identification (ssid)

This parameter identifies the DB2 subsystem to which Log Master connects. Log Master uses the subsystem identifier (SSID) to locate the DB2 address space and the DB2 logs. The SSID can be one to four characters long. You can also use a group attachment name (for DB2 data sharing) for this parameter. By using a group attachment name, you can run the same JCL on any member of a data sharing group. When you submit a job with a group attachment name, a DB2 subsystem in the group that is represented by the name must be running on the system where you submit the job.

Log Master does not require this parameter. If you omit the SSID, Log Master examines the libraries that are listed in your STEPLIB or JOBLIB statements, attempts to load a DSNHDECP module from those libraries, and then obtains the SSID from that module.

### MAINT parameter

In some situations, you might need to use a special EXEC statement parameter called MAINT instead of the SSID. The MAINT parameter prints the control section (CSECT) that tracks maintenance that is applied to Log Master. This information can help to determine whether a fix has been applied to your installation of the product. Log Master does not perform any other processing if you specify the MAINT parameter. The following example shows an EXEC statement that uses the MAINT parameter:

```
//stepname EXEC PGM=ALPMAIN,REGION=0M, 
// PARM='MAINT'
```

**NOTE**

Log Master does not require a working connection to DB2 to print the maintenance information.
UtilityID (utilityID)

This parameter is reserved for future use. You must honor its position in your list of EXEC parameters.

Message level (msgLevel)

This parameter determines which messages are returned in the ALPPRINT data set. Message level 0 causes Log Master to issue minimal procedural messages associated with the job. Message levels 1 and 2 cause Log Master to issue additional messages that you can use to tune job performance, or to diagnose problems.

Because message level 2 provides more of the information necessary to diagnose any potential problems, BMC Software recommends that you use message level 2 during normal operation. If you do not specify a message level, Log Master uses a default value of 2.

Installation options module ALPOPTS(optsMod)

This parameter directs Log Master to use an alternate installation options module for this job instead of the default module, ALPSOPTS. The optsMod name can be up to eight characters long, and Log Master responds to the name length as follows:

- If the name is more than four characters long, Log Master loads the specified installation options module.
- If the name is four characters long or less, Log Master interprets the name as a suffix override for the installation options module. For example, specifying ALPOPTS(TEMP) causes Log Master to load ALPS$TEMP as the installation options module.

Use an alternate installation options module to specify the option values that you need for a specific type of job. For example:

- If a job requires a large amount of row completion processing, increase the values of the KSMEMORY or KSSPACE installation options. This specific parameter can also be overridden by OPTION parameters at runtime.
- If a job requires a different value for an installation option, but you cannot change the value by using syntax in the SYSIN input, use a different installation options module. You can override many installation options in the SYSIN input, but not all of them. For example, you cannot override the CMPLGRNG installation option.

For more information about the options that you cannot override, and the content of the installation options module, see “Installation option descriptions” on page 42.
Specifying the STEPLIB DD or JOBLIB DD statements

Use STEPLIB DD statements or JOBLIB DD statements to specify the load libraries for Log Master and the DB2 subsystem where it will run. For example:

```
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
```

If you generate a job through the online interface, Log Master saves this information with your ISPF profile and restores it whenever you log on with the same TSO user ID.

As discussed in “APF authorizations for batch programs” on page 30, Log Master load libraries must be APF-authorized libraries. Ensure that all libraries in the STEPLIB or JOBLIB concatenation are APF authorized. If one library is not APF authorized, the operating system treats all of the libraries as if they are not APF authorized.

Specifying the DD statements

Log Master uses certain data sets depending on the type of processing that you request. You must specify some data sets by using a DD statement in the JCL, while Log Master allocates other data sets dynamically. Table 6 describes the data sets that are allocated in JCL.

Table 6  Data sets that are allocated in JCL (part 1 of 2)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Defines the input data set that contains the Log Master batch syntax keywords and their values. Always specify this data set in the JCL. The record format must be fixed blocked (RECFM=FB) with a record length of 80 (LRECL=80). Columns 73-80 are reserved for sequence numbers, and Log Master ignores any characters in those columns. You can choose to enter Log Master syntax within the JCL itself (in-stream). For more information about the Log Master syntax keywords and values, see Chapter 3, “Log Master for DB2 syntax.”</td>
</tr>
<tr>
<td>ALPPRINT</td>
<td>Defines the output data set that contains Log Master messages. Always specify this data set in the JCL.</td>
</tr>
<tr>
<td>SYSOUT</td>
<td>Defines the output data set that contains messages from the sort routine. Normally, you can specify this data set as SYSOUT=*.</td>
</tr>
</tbody>
</table>
### Table 6  Data sets that are allocated in JCL (part 2 of 2)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPDUMP</td>
<td><em>(optional)</em> Defines the output data set that contains a SNAP dump that Log Master might generate in response to internal errors. BMC recommends that you include this optional data set in your JCL as a troubleshooting aid.</td>
</tr>
<tr>
<td>SYSUDUMP</td>
<td><em>(optional)</em> Defines a data set that is used for system dumps that are produced when a system error occurs. BMC recommends that you include either SYSUDUMP or SYSMDUMP in your JCL as a troubleshooting aid.</td>
</tr>
<tr>
<td>SYSMDUMP</td>
<td><em>(optional)</em> Defines a data set that is used for system dumps that are produced when a system error occurs. Use either the SYSMDUMP data set or the SYSUDUMP data set. For SYSMDUMP, Log Master produces a system dump that can be read by the IBM Interactive Problem Control System (IPCS) product. Specify the attributes of the SYSMDUMP data set as indicated in the appropriate JCL documentation. BMC recommends that you include either SYSUDUMP or SYSMDUMP in your JCL as a troubleshooting aid.</td>
</tr>
<tr>
<td>ALPLRDMI</td>
<td>Defines a data set that is used by BMC Software Customer Support to diagnose row completion processing problems. Log Master dynamically allocates this data set as needed. You might be asked to provide this data set if your job displays BMC097461 messages.</td>
</tr>
<tr>
<td>SQLCODES</td>
<td><em>(optional)</em> Defines an input data set that contains specifications for handling SQL codes. As the product executes generated SQL, it uses these specifications to determine how to respond to the SQL codes. The DD statement is required if your job executes generated SQL and you want to change the default actions. For more information about defining these specifications, see “SQLCODES data set syntax” on page 328, or the chapter on executing SQL in the Log Master for DB2 User Guide.</td>
</tr>
<tr>
<td>SQLXLAT</td>
<td><em>(optional)</em> Defines an input data set that contains translation information. The product uses this information as it generates SQL. Using generated SQL and the translation information in this data set, you can migrate database changes that affect one set of objects to a similar set of objects with different names on another DBMS system. For more information about translating DB2 object names, see “SQLXLAT data set syntax” on page 328, or the chapter on defining the log scan step in the Log Master for DB2 User Guide.</td>
</tr>
</tbody>
</table>
| SQLPRINT      | *(optional)* Defines an output data set that the product uses as it executes generated SQL. The data set contains the results of SQL execution in the format that is generated by DSNTEP2 processing. If your job does not execute SQL, specify the data set as SYSOUT=*.
For more information, see “SQLPRINT” on page 307.
Defining sort work data sets

During normal operations, Log Master dynamically allocates sort work data sets using the specified values of several installation options. To improve the performance of Log Master sort actions, use one of the techniques described in this section to control the allocation. If you do not allocate enough sort work space, or if you place the sort work data sets on different units inefficiently, Log Master can perform at less than optimum levels.

You can also use the following methods to affect the performance of sort actions:

- You can control how much memory Log Master allocates for each invocation of the sort routine. For more information, see “SMCORE=0M,0K” on page 46.
- You can estimate the amount of data to be sorted and pass that estimate to the sort routine. For more information, see “Sort file size parameters” on page 154.

Controlling allocation with SORTOPTS keywords

For each Log Master job, you can override the installation options and control the dynamic allocation of sort work data sets. Use the SORTOPTS statement keywords listed in Table 7. For more information about each keyword, see “SORTOPTS statement” on page 150.

Table 7  SORTOPTS keywords for sort work data sets

<table>
<thead>
<tr>
<th>SORTOPTS keyword</th>
<th>Default value</th>
<th>Description</th>
<th>Associated installation option</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTDYN USE</td>
<td></td>
<td>Indicates whether Log Master uses the dynamic allocation rules of the sort</td>
<td>SORTDYN</td>
</tr>
<tr>
<td></td>
<td>USE</td>
<td>routine, or allows the user running the product to specify allocation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>characteristics.</td>
<td></td>
</tr>
<tr>
<td>WORKOPTS UNIT</td>
<td>SYSALLDA</td>
<td>Specifies the unit name that Log Master uses for dynamic allocation of</td>
<td>WKUNIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sort work data sets.</td>
<td></td>
</tr>
<tr>
<td>WORKOPTS STORCLAS</td>
<td>(no default)</td>
<td>Designates the DFSMS storage class that Log Master uses for dynamic</td>
<td>WKSTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>allocation of large temporary files and sort work data sets.</td>
<td></td>
</tr>
<tr>
<td>WORKOPTS NUM</td>
<td>0</td>
<td>Specifies the number of sort work data sets that Log Master uses.</td>
<td>WORKNUM</td>
</tr>
</tbody>
</table>
Controlling allocation with DD statements

You can include DD statements for sort work data sets in the JCL. If you include DD statements, the following conditions apply:

- The DD names must conform to the convention described in “Naming sort work data sets in DD statements” on page 104.

- The number of data sets varies depending on the number of sort actions in the job.

Guidelines for allocating sort work data sets with DD statements

Use the following guidelines when you include DD statements for sort work data sets in the JCL:

- To help the sort routine perform optimally, provide a file size estimate. Use the FILSZ keyword of either the LOGSCAN or SORTOPTS statement. For more information, see “Sort file size parameters” on page 154.

- When you include DD statements in your JCL, pay particular attention to the WER124I message that the sort routine issues. This message describes whether you allocated excess or insufficient space for your sort work data sets.

- Allocate sort work data sets in units of cylinders.

- Allocate the first sort work data set as CYL(1,1), which ensures that the sort routine uses the file size estimate passed by Log Master. If you allocate too much space to the first sort work data set, the sort routine might choose an inefficient sort algorithm. If your job uses an incorrect sort algorithm, the job might perform poorly or terminate abnormally when it exceeds sort capacity.

- Allocate all sort work data sets on one device type, preferably the device with the fastest data transfer rate.

- If possible, allocate sort work data sets on different VOLSERs across different channel paths.

- Do not use VIO data sets for sort work data sets.

- Allocate enough sort work space in primary storage to contain all of the input data.

- Start by allocating four sort work data sets. Allocate more only if you must (because of problems in obtaining extents). If you allocate more than 12 sort work data sets, high overhead can degrade performance. This limit applies even if the sort routine accepts more than 12 sort work data sets.
When you must sort a large volume of data (over one gigabyte), allocate four to twelve sort work data sets on separate devices, depending on the file size. If possible, allocate an equivalent number of channel paths for optimal performance.

If your job has a very large sort (over two gigabytes) and performance and elapsed time are important, consider turning off dynamic allocation by specifying SORTDYN as USE and WORKOPTS NUM as 0 on the SOROPTS statement. Use DD statements in your JCL to allocate the sort work data sets. Be sure to allocate enough space to process all of the data.

### Naming sort work data sets in DD statements

The DD names of sort work data sets must use the following convention:

`SWxxWKyy`

The variables are defined as follows:

- `xx` represents the number of a sort action, and ranges from one to the number of sort actions required by the specific job. For more information on determining the number of sort actions, see page 105.

- `yy` represents the sequence number of a sort work data set, and ranges from one to the maximum number of sort work data sets that are allocated for each required sort action. Log Master determines the maximum number from the WORKNUM installation option. You can override WORKNUM by using the WORKOPTS NUM keyword of the SOROPTS statement.

In some cases, you must allocate an additional group of sort work data sets to sort DB2 log information. The DD names of the additional sort work data sets must use the following convention:

`LOGSWKyy`

Allocate the additional data sets when any of the following sets of conditions exist:

- The value of the LOGSORT option is ALWAYS.

- Your job step reads input from the bootstrap data set (the default value for input), and all of the following conditions exist:
  - Your job step runs in a data sharing environment.
  - The value of the LOGSORT option is WHENEVER.
  - The value of the LOGTAPES option is greater than zero, but less than the number of members in your data sharing group.
Controlling allocation with DD statements

- Your job reads input from multiple DB2 log files (by using the DB2LOG keyword of the INPUT statement), and all of the following conditions exist:

  — Your job runs in a data sharing environment.

  — The value of the LOGSORT option is WHENEVER.

  — The value of the LOGTAPES option is greater than zero, but less than the number of members in your data sharing group.

  — Your job step uses a character string as a member name to group your log files.

Numbering sort work data sets in DD statements

To determine the number of sort actions required by a given job, use the information in Table 8 to compute a total number.

Table 8  Computing the required number of sorts for a job (part 1 of 2)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Add x sorts</th>
<th>Total sorts for this condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A log scan step requests a report.</td>
<td>1 for each report</td>
<td></td>
</tr>
<tr>
<td>A log scan step requests a Backout Integrity report.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(In addition to the sorts required for any other reports.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A log scan step requests a Catalog Activity report.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(In addition to the sorts required for any other reports.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A log scan step requests any SQL output files.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>The product reuses the same sort for MIGRATE, UNDO, or REDO SQL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A log scan step requests any separate DDL output files.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>The product reuses the same sort for MIGRATE or UNDO DDL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A log scan step creates any load files as output.</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
As an example, assume that a job has the following conditions:

- one log scan step that generates a Detail report for one time range
- a second log scan step that generates a Backout Integrity report
- UNDO SQL for a different time range

The job in this example requires the following sorts:

- two sorts (one for each report in the two log scan steps)
- one additional sort because one of the reports is a Backout Integrity report
- one sort for the SQL in the second log scan step

The total number of sorts required by this job is four. Assume that the value of the WORKNUM installation option is 2. The number of sort work data sets required is eight, and the job’s JCL must allocate them using the following DD names:

<table>
<thead>
<tr>
<th>Sort Data Set Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW01WK01</td>
</tr>
<tr>
<td>SW01WK02</td>
</tr>
<tr>
<td>SW02WK01</td>
</tr>
<tr>
<td>SW02WK02</td>
</tr>
<tr>
<td>SW03WK01</td>
</tr>
<tr>
<td>SW03WK02</td>
</tr>
<tr>
<td>SW04WK01</td>
</tr>
<tr>
<td>SW04WK01</td>
</tr>
</tbody>
</table>

### Specifying Log Master syntax

Place Log Master statements in the location specified by the SYSIN DD statement, either in a data set or in-stream in the JCL. For more information about the actual syntax to enter, see Chapter 3, “Log Master for DB2 syntax.”
SYSIN syntax in-stream in JCL

For an example showing Log Master SYSIN syntax within your JCL (in-stream), see Figure 2 on page 95.

SYSIN syntax in a separate data set

You can include all or part of the SYSIN syntax in a separate data set. For example, Figure 3 shows SYSIN syntax that includes a separate file as part of a Log Master filter. This technique allows you to dynamically generate a list of tables for each run of a job without changing your JCL.

Figure 3  JCL with SYSIN syntax in a separate data set

```
//SYSIN     DD *
/* DB2DBA.$$WORKID0010 */
/* DB2DBA 2012-12-18 14.21.58 */

OPTION
  FILTERREL AND EXECUTION MODE CURRENT

SORTOPTS
  FILSZ NONE
  HISTORY WRITE

WORKID DB2DBA.$$WORKID0010
  DESC "DB2DBA 2012-12-18 14.21.58 MIGRATE"

LOGSCAN
  REPORT TYPE SUMMARY
  SYSOUT
    CLASS(*) NOHOLD
  DB2CATALOG NO
  FROM DATE(2012-12-18) TIME(12.39.00.000000)
  TO  DATE(2012-12-18) TIME(13.50.00.000000)
  WHERE
    TABLE NAME IN (
    //    DD DSN=DB2DBA.TABLIST.DATA,DISP=SHR
    //    DD *,DLM=##
    )
  #
  //SQLCODES DD DUMMY
```
Comments in SYSIN syntax

You can enter comments within your SYSIN syntax by using any of the following comment conventions:

- **multiple line, C style**

  Log Master ignores all characters between /* and */ regardless of the number of lines that you include. The comment delimiters can occur in any position. For example:

  ```c
  /* Comment text */
  CYLINDERS SPACE(1,1) UNIT(SYSDA) RELEASE
  INCLUDE RI NO
  
  ********************
  * Next comment text *
  ********************
  ```

- **single line, C++ style**

  Log Master ignores all characters between // and the end of the current line. The opening comment delimiter can occur in any position except column one. For example:

  ```c
  // Comment out old syntax CYLINDERS SPACE(1,1)
  
  // Comment text
  INCLUDE RI NO  // regarding referential integrity setting.
  ```

- **single line, DB2 style**

  Log Master ignores all characters between -- and the end of the current line. The opening comment delimiter can occur in any position. For example:

  ```c
  -- Comment out old syntax CYLINDERS SPACE(1,1)
  
  -- Comment text
  INCLUDE RI NO  -- regarding referential integrity setting.
  ```
Long names in Log Master syntax

In SYSIN input, Log Master supports the longer name lengths for DB2 objects that are defined in DB2. Log Master parses long names and displays them in output messages.

In the SYSIN input of your job, long object names must appear in columns 1 to 72. Columns 73 to 80 are ignored. You can split long names in SYSIN across lines. When an object name is split across lines, the name must continue to column 72, and the remainder of the name must start in column 1 on the next line. Use delimiters to include spaces in a DB2 object name. For example:

```
WHERE
  TABLE_NAME = DB2DBADB.DB2DBA_YEAR_END_FINANCIAL_DATA_AND_QUARTERLY_ESTIMATES_CONSOLIDATION_TABLE
  OR
  TABLE_NAME = DB2DBADB."DB2DBA YEAR END FINANCIAL DATA AND MONTHLY ESTIMATES CONSOLIDATION TABLE"
```

Running Log Master jobs

To run a Log Master batch job, use the same procedures that you use to run standard JCL jobs. Table 9 shows the requirements for running a Log Master job.

Table 9  Requirements for running Log Master jobs (part 1 of 2)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>To run a Log Master job, you must have the proper authorization.</td>
<td>“Authorization needed to use Log Master” on page 27</td>
</tr>
</tbody>
</table>
| Starting a job            | Specify the module ALPMAIN on the EXEC statement of your JCL. You must also specify some required parameters and some DD statements. | ■ “Specifying the EXEC statement” on page 97  
                              ■ “Specifying the DD statements” on page 100.                                      |
| Displaying the status of a job | Log Master does not require special commands or procedures to monitor a Log Master job. Use the standard operating system commands to monitor the job’s progress. | none                                                       |
Log Master termination processing

Table 9  Requirements for running Log Master jobs (part 2 of 2)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restarting a job</td>
<td>You can rerun or re-execute Log Master jobs, but you cannot restart them. Log Master supports a type of processing referred to as ongoing processing. Use this type of job to repeatedly run the same set of log processing. Log Master saves information about each run of the job in the Repository, and succeeding jobs can begin processing where the previous job ended. Log Master supports the RERUN and USE keywords to re-execute previous runs of ongoing jobs.</td>
<td>“ONGOING HANDLE handleID” on page 176</td>
</tr>
<tr>
<td>Terminating a job</td>
<td>Log Master does not require special commands or procedures to terminate a Log Master job. Use the standard operating system commands to cancel the job if necessary.</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Log Master termination processing

Table 10 on page 111 summarizes how Log Master attempts to respond when it encounters a return code 8 (error) or greater. Log Master produces output data sets and updates the Repository differently in response to these return codes depending on whether the return code is 8 (error), or 12 (severe), and whether a work ID contains at least one ongoing log scan.

NOTE

This section provides general information about Log Master termination processing in response to error conditions. Depending on the circumstances in your environment, or the circumstances of a particular error condition, the product might respond differently or might not be able to respond as described here.
Log Master termination processing

Table 10  Log Master output handling at termination

<table>
<thead>
<tr>
<th>Return code</th>
<th>Work ID includes at least one ongoing log scan</th>
<th>Actions during termination processing</th>
</tr>
</thead>
</table>
| 8 no        | - write and catalog all output data sets (see “Note 1: Write and catalog output files”)
            | - update the Repository with history information about the work ID |
|             | If the work ID includes multiple log scans, the product takes these actions for all log scans (including log scans that did not encounter an error). |
| 8 yes       | - purge and uncatalog logical log, load, SQL, and DDL output data sets
            | - write and catalog all reports (see “Note 1: Write and catalog output files”)
            | - update the Repository with history information about the work ID
            | - discard updates to the ALPURID table of the Repository (see “Note 2: Update ALPURID repository table” on page 112) |
|             | If the work ID includes multiple log scans, the product takes these actions for all log scans (including log scans that did not encounter an error and log scans that are not ongoing). |
| 12 no       | - purge and uncatalog logical log, load, SQL, and DDL output data sets
            | - write and catalog all reports (see “Note 1: Write and catalog output files”)
            | - update the Repository with history information about the work ID |
|             | If the work ID includes multiple log scans, the product takes these actions for all log scans (including log scans that did not encounter an error). |
| 12 yes      | - purge and uncatalog logical log, load, SQL, and DDL output data sets
            | - write and catalog all reports (see “Note 1: Write and catalog output files”)
            | - update the Repository with history information about the work ID
            | - discard updates to ALPURID table of Repository (see “Note 2: Update ALPURID repository table” on page 112) |
|             | If the work ID includes multiple log scans, the product takes these actions for all log scans (including log scans that did not encounter an error and log scans that are not ongoing). |

Note 1: Write and catalog output files

Log Master processes different forms of output simultaneously (in parallel). Because of this behavior, the output data sets that Log Master writes and catalogs after an error can contain complete or partial output. If processing for a form of output encounters return code 8 or greater, that data set might contain only output that was generated before the product encountered the error condition.
Note 2: Update ALPURID repository table

Log Master tracks each run of an ongoing log scan by using the ALPURID table in the Repository. When an ongoing log scan encounters return code 8 or greater, the product assumes that the log scan must be run again so that all output files include the same set of transactions. Because of this assumption, the product does not update the ALPURID table with information from the log scan that encountered the error. This action ensures that the next run of the ongoing log scan uses the same start point as the run that encountered the error.

Cleaning up after a Log Master job

Many environments occasionally use tools and actions to clean up after a Log Master job, or after an extended series of Log Master jobs. Log Master stores a small amount of information in the Repository each time it runs a job. Over time, the Repository grows. To reduce the size of the Repository, use one of the methods described in “Reduce Repository size when required” on page 385.
This chapter contains the following topics:

Using the syntax diagrams .................................................. 114
Log Master high-level syntax .............................................. 115
OPTION statement ............................................................ 119
  Overtime mode ............................................................... 135
LOGOPTS statement .......................................................... 137
STOREOPTS statement ....................................................... 139
LOBOPTS statement .......................................................... 144
XMLOPTS statement .......................................................... 147
SORTOPTS statement .......................................................... 150
  Sort file size parameters .................................................. 154
WORKID statement ............................................................. 156
IMAGECOPY statement ....................................................... 157
  Image copy parameters .................................................... 159
INPUT statement ............................................................... 161
  DB2 log input definition .................................................. 162
LOGSCAN statement ........................................................... 164
  Ongoing log scans .......................................................... 179
LOGSCAN report definition .................................................. 181
  ORDER BY definition ....................................................... 194
  SYSOUT definition ........................................................ 197
LOGSCAN logical log output definition .................................... 200
LOGSCAN SQL file definition ................................................. 208
  SQL type/output definition .............................................. 214
  Considerations for output files and SQL ................................ 225
  Considerations for output files and Unicode .......................... 229
LOGSCAN load file definition ............................................... 231
  Layout of records in load files ......................................... 249
  Additional Informational Columns in Load Files ...................... 250
LOGSCAN DDL file definition ................................................ 255
LOGSCAN output definition ................................................ 260
  Allocation parameters .................................................... 262
LOGSCAN column include/exclude definition ................................ 267
LOGSCAN scan range definition .......................................... 269
  Range definition ........................................................... 272
Using the syntax diagrams

This chapter contains diagrams of the Log Master for DB2 batch interface syntax, and brief descriptions of the purpose and use of the syntax.

Enter product syntax through the SYSIN DD statement of your JCL. Follow the required order of operation for each basic statement. For more information, see Chapter 2, “Building and running Log Master jobs.”

For a description of the conventions used in the syntax diagrams, see page 17. In addition to the general conventions, the following information can help you use the diagrams and descriptions more efficiently:

- This chapter lists the statements and syntax groups in normal order of appearance in the SYSIN input.

- The syntax descriptions for each unit of syntax follow the syntax diagram. The descriptions list keywords in roughly the same sequence as they appear in the diagram, moving left to right, and then top to bottom. The page number of the description appears beside the major keywords. In the example in the following paragraph, the description of the HISTORY keyword is on page 153.
Log Master high-level syntax

Figure 4 on page 116 shows the overall, high-level syntax for Log Master. To define a Log Master job, enter job-level statements (for example, an OPTION statement), and then specify step-level statements (for example, a LOGSCAN statement).

Job-level statements define conditions for the entire Log Master job. You can enter each statement only once for each job.

Step-level statements define one step within a Log Master job. Enter as many LOGSCAN, LOGMARK, or EXECSQL statements as you need. You can enter only one DROPRECOVERY statement for each job.

Place Log Master syntax in the location specified by the SYSIN DD statement (either in a data set or in-stream in the JCL). For more information about entering comments within your SYSIN syntax, see “Specifying Log Master syntax” on page 106.
WORKID

Defines a unique identifier that Log Master uses to identify a unit of work. Create the work ID using the online interface, or define it explicitly in batch syntax. You can use only one WORKID statement for each Log Master job. For more information, see “WORKID statement” on page 156.

OPTION

Specifies options that are set globally for use in a Log Master job or job step. Use the OPTION statement to override installation options for one run of a Log Master job. If needed, use one OPTION statement for each Log Master job. Enter the statement at the beginning of the SYSIN syntax. For more information, see “OPTION statement” on page 119.
LOGOPTS

Specifies options that control how Log Master processes DB2 log files. Use the LOGOPTS statement to override installation options for one run of a Log Master job. You can use only one LOGOPTS statement for each job. For more information, see “LOGOPTS statement” on page 137.

STOREOPTS

Specifies options that control how Log Master handles internal memory or file structures called key stores. Use the STOREOPTS statement to override installation options for one run of a Log Master job. You can use only one STOREOPTS statement for each job. For more information, see “STOREOPTS statement” on page 139.

LOBOPTS

Specifies options that control how Log Master maintains the VSAM files that it uses to store large object (LOB) data. Use the LOBOPTS statement to override installation options for one run of a Log Master job. You can use only one LOBOPTS statement for each job. For more information, see “LOBOPTS statement” on page 144.

XMLOPTS

Specifies options that control how Log Master maintains the VSAM files that it uses to store XML data. Use the XMLOPTS statement to override installation options for one run of a Log Master job. You can use only one XMLOPTS statement for each job. For more information, see “XMLOPTS statement” on page 147.

SORTOPTS

Specifies options that control how Log Master allocates memory for sort operations and sort work data sets. Use the SORTOPTS statement to override installation options for one run of a Log Master job. You can use only one SORTOPTS statement for each job. For more information, see “SORTOPTS statement” on page 150.

IMAGECOPY

Specifies exact locations for image copies that Log Master reads for input. Use this statement to specify input image copies that are not currently defined in the DB2 catalog. You can use only one IMAGECOPY statement for each Log Master job. For more information, see “IMAGECOPY statement” on page 157.
Log Master high-level syntax

**LOGSCAN**

Defines how Log Master reads DB2 log input, how it selects individual log records, and what forms of output it produces. Much of the product’s normal work is controlled using various keywords of the LOGSCAN statement. For more information about the high-level syntax of the LOGSCAN statement, see “LOGSCAN statement” on page 164.

**LOGMARK**

Directs Log Master to associate the current relative byte address or log record sequence number (RBA/LRSN) value with a name that you enter. You can use the log mark name in subsequent product syntax in place of the RBA/LSRN value. For more information, see “LOGMARK statement” on page 305.

**EXECSQL**

Directs Log Master to apply a given set of generated Structured Query Language (SQL) statements to a DB2 database. The SQL statements must be generated by Log Master. For more information, see “EXECSQL statement” on page 306.

**DROPRECOVERY**

Directs Log Master to obtain information from the DB2 log and generate several forms of output, including a set of recovery syntax that can be passed to the BMC Software RECOVER PLUS for DB2 product, or DSN1COPY. Using the syntax and information, the Recover utility can recover a DB2 object (for example, a table) that has been dropped from the DB2 catalog.

You can have only one DROPRECOVERY statement for each job. For more information, see “DROPRECOVERY statement” on page 308.
OPTION statement

Figure 5 shows the OPTION statement syntax. The OPTION statement specifies global options for use in a Log Master job. You can use the OPTION statement to override some installation options. If needed, enter one OPTION statement for each job. The OPTION statement must appear before any other statements.

Figure 5  OPTION statement syntax diagram (part 1 of 2)
**REPOS**

Determines whether Log Master updates the Repository during the current job step.

**YES**

Log Master updates the Repository. The default value is YES.

**NO**

Log Master does not update the Repository. If you specify NO, you cannot specify the REPOS UPDATE or REPOS DELETE keywords on the LOGSCAN statement.

**DATEFMT**

Determines the format that Log Master uses to display date and time data on all reports. Log Master supports the following date and time formats. The default value is ISO.

**USA**

MM/DD/YYYY/HH:MM:SS.nnnnnn

**EUR**

DD.MM.YYYY.HH.MM.SS.nnnnnn

**ISO**

YYYY-MM-DD-HH.MM.SS.nnnnnn

**JIS**

YYYY-MM-DD-HH:MM:SS.nnnnnn

When running on DB2 Version 10 and later, Log Master supports precision timestamps up to 12 digits, and inclusion of a time zone in the timestamp (YYYY-MM-DD-HH:MM:SS.nnnnnnnnnnn±HH:MM).

**EXECUTION MODE**

Specifies the execution mode for Log Master.

**CURRENT**

Directs Log Master to run in current mode. In this mode, Log Master selects log records relating to the objects that you specify, but only if the objects currently exist in the DB2 catalog. This is the default value.
OPTION statement

OVERTIME
Directs Log Master to run in overtime mode. In this mode, Log Master selects log records relating to all of the objects that you specify, regardless of whether the objects currently exist in the DB2 catalog. For more information about overtime mode, see “Overtime mode” on page 135, or see the chapter on objects over time in the Log Master for DB2 User Guide.

To update the Repository, you must select overtime mode. If you specify more than one logical log control file as input (directly or by specifying a GDG base), Log Master automatically runs in overtime mode.

The ATTEMPT COMPLETION keyword enables Log Master to plan for and use image copies during row completion processing in overtime mode. By default, when Log Master runs in overtime mode it does not perform row completion on any log records associated with objects that do not exist in the DB2 catalog.

To complete log records for objects that do not exist in the DB2 catalog, take two actions: specify this keyword and include an IMAGECOPY statement (to specify the names of image copy data sets that contain the desired objects). For more information about the IMAGECOPY statement, see page 157. Depending on the objects you select and the activity related to those objects, you might need to specify multiple image copy data sets. These actions increase the chances of successful row completion processing.

See also “EXECMODE=CURRENT” on page 66.

OLD OBJECTS dataSetName
Specifies the name of the old objects data set that Log Master uses. You can create an old objects data set to hold structure definitions of DB2 objects that are not currently defined in the DB2 catalog (old objects). The old objects data set should be used only when Log Master is operating in overtime mode. For more information about the syntax used within this data set, see “Old objects data set syntax” on page 326.

FILTERREL
Specifies the relational operator that Log Master uses to connect multiple filters.

AND
Directs Log Master to connect filters with an AND relational operator. This is the default value.

OR
Directs Log Master to connect filters with an OR relational operator.
FILTER METHOD

Determines how and when Log Master obtains DB2 catalog information (DBIDs, OBIDs, or PSIDs) for the DB2 objects that are named in a filter. Log Master reads the DB2 catalog to resolve the names of DB2 objects into numeric identifiers. Log Master can read the catalog either once during the initial analyze phase of processing for all objects, or repeatedly as it encounters each object in the DB2 log.

**STATIC**

Directs Log Master to obtain DB2 catalog information during the analyze phase of processing. Log Master obtains information for all DB2 objects that the filter explicitly names. This is the default value.

**DYNAMIC**

Directs Log Master to obtain DB2 catalog information dynamically as it scans the DB2 log. Log Master obtains information for the DB2 objects that are present in the scanned log records and selects log records for only the objects that are named in the filter.

Consider the following performance implications as you choose a value:

- When you choose STATIC, Log Master can experience degraded performance during the analyze phase when all of the following conditions exist:
  
  — Your DB2 subsystem contains a very large number of objects (for example, some enterprise resource planning applications generate tens of thousands of objects).
  
  — Your filter uses a LIKE or NOT LIKE operator (for example, `TABLE NAME LIKE OWNER.%`).
  
  — The number of objects that are actually updated during the range of the log scan is significantly smaller than the number of objects that are named by the filter.

- When you choose DYNAMIC, Log Master can incur extra processing overhead during the log scan as it regenerates the filter each time it encounters a new DB2 object.

**NOTE**

BMC Software recommends retaining the default value unless you experience degraded performance during the Log Master analyze phase. Use the elapsed time values provided by message BMC097024 ANALYZE FINISHED to determine the performance of the analyze phase.
If you select dynamic filtering (DYNAMIC), do not specify the following items. The processing required for these items is not compatible with dynamic filtering.

- The GENERATE EMPTY FILES keyword for load file output.
- A value of YES for the USELGRNG keyword of the OPTION statement.
- The LASTQUIESCE keyword of the scan range definition for REDO SQL output.

See also “FLTRMTHD=STATIC” on page 66.

**IMAGESOURCE**

Specifies the source that Log Master uses to perform row completion or decompression processing. If you do not enter a value for this keyword, Log Master uses the value of the IMAGESRC installation option. Valid values are as follows:

- **ANY**
  Perform row completion or decompression from any source available, including the DB2 log, the table space itself, or an image copy. When you select this value, Log Master uses the values of the FILECOST keyword to select a row completion or decompression source.

- **TABLESPACE**
  Perform row completion or decompression from the table space only. This value increases the risk that Log Master can terminate with either BMC097386 “unable to decompress,” or one of several “unable to complete” error messages.

- **SYSCOPY**
  Perform row completion or decompression using only resources from the SYSIBM.SYSCOPY catalog table (including image copies or other events such as LOAD LOG YES actions). For more information on the types of image copies that Log Master can read, see “Reading image copies.”

- **LOGONLY**
  Perform row completion or decompression from the DB2 log only.

Log Master performs row completion or decompression processing to rebuild a complete image of a table row at a given point in time. Unless a table is defined with the Data Capture Changes (DCC) attribute, the log record of an update action usually contains only part of the table row (enough to include the changed data). Log Master uses the record ID (RID) value in the log record to obtain information about the row from other sources. For more information about row completion or decompression processing, see the chapter on Log Master for DB2 expert information in the Log Master for DB2 User Guide.

See also “IMAGESRC=ANY” on page 56.
Reading image copies

Log Master uses image copies for row completion processing or to obtain compression dictionaries. Be aware of the following points regarding how Log Master uses image copies:

- When Log Master requires image copies, it attempts to read image copies in the following order:
  - for local sites, default order is (FC, LP, LB, RP, RB)
  - for remote sites, default order is (FC, RP, RB, LP, LB)

You can change these defaults using syntax overrides or by changing installation option values. For more information, see the following sections:

  - “LOCCPSEL=(FC,LP,LB)” on page 85
  - “REMCPSEL==(RP,RB,FC)” on page 85
  - “RESOURCE SELECTION” on page 134

- Log Master can read Instant Snapshot image copies that were created on intelligent hardware storage devices by the BMC Software COPY PLUS for DB2 product with Snapshot Upgrade Feature (SUF).

- Log Master can read encrypted image copies created by COPY PLUS if the name of the key data set is provided by using the KEYDSNAM installation option.

- Log Master can read cabinet copies created by COPY PLUS. Cabinet copies contain a group of table spaces and indexes within a single cabinet file to provide performance improvements when managing large numbers of small table spaces.

- To read Instant Snapshot, encrypted, or cabinet image copies, both Log Master and COPY PLUS must use the same instance of the BMC Software BMC_BMCXCOPY table.

- Log Master cannot read Data Facility Storage Management System (DFSMS) concurrent image copies, regardless of how they were created (by using the CONCURRENT keyword of a DB2 Copy utility, or by using DFSMS outside of DB2). If the only available source for row completion processing or a dictionary is a concurrent image copy, Log Master might encounter errors or terminate abnormally.

- Consider running regular jobs to update the Log Master Repository with copies of compression dictionaries. This action can improve overall performance by enabling Log Master to avoid mounting image copies to retrieve dictionaries to process log records of compressed table spaces.
FILECOST

Assigns a relative cost to the act of reading a separate file, mounting a tape, or reading a segmented table space. Log Master uses the FILECOST values only if the IMAGESOURCE keyword is set to ANY.

Log Master uses the cost values as it chooses a source for row completion processing. Use this syntax to adapt Log Master row completion processing to your environment. To enter cost values, express them in terms of the cost of reading one page from the DB2 log.

**costFile**

Assigns a relative cost to processing an additional data set, expressed in terms of the cost to read one page from the DB2 log. In many cases, the data set contains image copy information. To increase the probability that Log Master uses image copies for row completion, enter a lower value. The default value is 2,000.

To improve performance, as Log Master chooses between available image copy data sets, it assigns a lower `costFile` value to any Instant Snapshot image copies that are available. Instant Snapshot image copies are created on intelligent hardware storage devices by the BMC Software COPY PLUS for DB2 product with Snapshot Upgrade Feature (SUF).

See also “CSTFILE=2000” on page 57.

**costMount**

Assigns a relative cost to performing a single tape mount, expressed in terms of the cost to read one page from the DB2 log. The default value is 25,000.

See also “CSTMOUNT=25000” on page 58.

**costSeg**

Assigns a relative cost to obtaining information from a segmented or universal table space, expressed in terms of the cost to read one page from the DB2 log. The default value is 2,000,000,000.

For more information about the risks of row completion failure with mass delete actions in these types of table spaces, see “CSTSEG=2000000000” on page 58.

MINLOGPT

Specifies how Log Master determines the log scan end point when it runs in a data sharing environment. Specify YES to obtain the same end point across all members of the data sharing group.
**NO**
Indicates that Log Master does not require a common end point for all members. Log Master uses the end point that you specify. This is the default value. When you run an ongoing log scan in a data sharing environment, Log Master dynamically sets the value of this keyword to YES, regardless of the value that you specify.

**YES**
Indicates that Log Master requires a common end point for all members of the data sharing group. A common end point is important for ongoing log scans, where Log Master requires common end points and start points to select all of the desired log records across multiple runs.

If you do not enter a value for MINLOGPT, Log Master uses the value of the corresponding installation option. For more information (including a diagram), see “MINLOGPT=NO” on page 60.

**USELGRNG**
Determines whether Log Master uses the SYSIBM.SYSLGRNX table in the DB2 directory to determine the ranges for a log scan. Use this keyword only in a data sharing environment.

Log Master reads the SYSLGRNX table only when a WHERE clause or filter refers to specific DB2 objects (columns, tables, table spaces, or databases). Log Master uses this table to determine whether the DB2 log of a data sharing member contains information about the database structures that are defined in your log scan. With this information, Log Master can avoid reading log files of members that show no activity during the initial log scan (before row completion processing). This action can improve overall performance. If you do not enter a value for USELGRNG, Log Master uses the value of the corresponding installation option.

**YES**
Indicates that Log Master uses the SYSLGRNX table to determine valid ranges.

**NO**
Indicates that Log Master does not use SYSLGRNX to determine valid ranges.

Specifying YES can improve Log Master performance. However, Log Master can experience degraded performance reading the SYSLGRNX table if that table is not maintained (with a DB2 Modify utility). Use the elapsed time value provided by message BMC097168 to determine the performance of Log Master SYSLGRNX processing. To improve performance in this situation, specify NO.

See also “USELGRNG=NO” on page 61.
OPTION statement

PROCESS PITS

Specifies that, as Log Master scans the log, it will include any log records that fall within the range of a Point-in-Time (PIT) recovery. By default, Log Master does not select log records within a PIT range, regardless of your WHERE clause or filter. In rare situations, you might need log records from within a PIT range.

A PIT recovery is a partial recovery (performed with a DB2 Recover utility) that restores a set of DB2 objects to their state at a previous point in time. After a PIT recovery is performed on a set of objects, the DB2 log contains a range of log records for those objects that are no longer valid (because the objects were recovered to a point before the log records were created). This range of invalid log records is called a PIT range. Information about PIT ranges is stored in the SYSIBM.SYSCOPY table of the DB2 catalog.

**WARNING**

BMC Software does not recommend using log records from within a PIT range. Exercise caution as you select log records within a PIT range or use the information contained in those log records.

If you take actions or apply changes to your database based on the output of a log scan that combines PIT and normal log records, you can corrupt the data in your database.

To process from log records within a PIT range, process the PIT log records in a separate log scan that covers only the PIT range.

DICTIONARYSPACE value

Specifies a limit on the amount of memory that Log Master uses to store compression dictionaries during processing. Adjusting the value can change the performance of log scans that read compressed table spaces.

If you do not specify a value for DICTIONARYSPACE, Log Master uses the value of the DICTSPC configuration option, which defaults to 50 M.

To override the value of DICTSPC at runtime, enter an amount of memory in kilobytes (use the suffix K) or megabytes (use the suffix M).

For more information about how Log Master uses this memory and suggestions for calculating a DICTIONARYSPACE value, see “DICTSPC=50M” on page 55. For information about estimating overall memory, see the chapter on Log Master for DB2 expert information in the Log Master for DB2 User Guide.

**NOTE**

Log Master cannot obtain compression dictionaries from a DFSMS concurrent image copy. For more information, see “Reading image copies” on page 125.
EARLY RECALL

Directs Log Master to determine whether any log files that it requires have been migrated (marked as archived in the ICF catalog). If so, Log Master issues requests to recall the required data sets before they are needed for log processing. This action avoids delays in log processing by giving the storage management software in your environment time to retrieve the required data sets.

You must set EARLY RECALL to YES to enable any of the other early recall keywords (such as MIGRATE or DASD DATASETS). The SMSTASKS keyword defines limits for the EARLY RECALL keyword. The Log Master early recall feature works with most storage management software. The related data set migration feature can migrate only data sets that are managed by the IBM DFSMShsm product.

Before you enable early recall of archived data sets, evaluate the settings of the SMSTASKS, MIGRATE, and DASDDSNs keywords to be sure that they are appropriate for your environment.

YES
Directs Log Master to issue early recall requests. This is the default value.

NO
Prevents Log Master from issuing early recall requests.

See also “ERLYRCL=YES” on page 78.

SMSTASKS nnn

Determines the maximum number of early recall subtasks that Log Master creates. Use this keyword to optimize performance when log processing requires large numbers of migrated files.

Enter a maximum number of subtasks. To allow Log Master to determine the number of subtasks, enter 0 (the default value). When you specify 0, Log Master sets the SMSTASKS value as follows:

- In a non–data sharing environment, Log Master sets SMSTASKS to 1.
- In a data sharing environment, Log Master sets SMSTASKS equal to the number of data sharing members.

See also “SMSTASKS=0” on page 79.

PER MEMBER

Specifies the scope of the SMSTASKS value. This keyword applies only in a data sharing environment.
YES
Indicates that the value is a limit for each member of a data sharing environment. If you specify YES and the SMSTASKS value is currently set to 0, Log Master sets the SMSTASKS value as follows:

- In a non-data sharing environment, Log Master sets SMSTASKS to a value of 1.
- In a data sharing environment, Log Master sets SMSTASKS to a value equal to the number of members in the data sharing group.

NO
Indicates that the value is a limit for the entire data sharing group. This is the default value.

See also “MTPRMBR=NO” on page 79.

MIGRATE
Directs Log Master to request that any recalled data sets be migrated to their original status. The default value is YES. The DASD DATASETS keyword places limits on the MIGRATE keyword. Log Master issues migration requests at the end of a job, or when a job requires a greater number of data sets than the DASD DATASETS value. If a job requires more data sets than the DASDDSNS value permits, Log Master migrates data sets even when this keyword is set to NO.

Log Master always attempts to migrate the data set back to its original migration level. The Log Master data set migration feature can migrate only data sets that are managed by the IBM DFSMShsm product. The related early recall feature works with most storage management software.

Before you enable migration of recalled data sets, evaluate the settings of the SMSTASKS, ERLYRCL, and DASDDSNS keywords to ensure that they are appropriate for your environment.

YES
Directs Log Master to issue migration requests.

NO
Prevents Log Master from issuing migration requests (unless the DASDDSNS value is exceeded).

See also “MIGRATE=YES” on page 80.

WAIT
Determines whether Log Master terminates at the end of processing or waits for all data set migration requests to complete. If you do not specify this keyword, Log Master uses the value of the MIGRWAIT installation option (see “MIGRWAIT=NO” on page 80).
**DASD DATASETS nnn**

Determines the number of recalled data sets that Log Master attempts to maintain on DASD at any one time. Use this keyword to minimize DASD requirements that might result when log processing requires large numbers of archived files.

To honor the DASD DATASETS value, Log Master issues requests to migrate any recalled data sets to their original migration status and level. If a job requires more data sets than this value, Log Master issues requests to migrate data sets even if the value of the MIGRATE keyword is NO.

Depending on how quickly your environment processes migrate requests, for short periods of time, the number of recalled data sets on DASD might be greater than the DASD DATASETS value. Log Master can migrate only data sets managed by the IBM DFSMShsm product. If DFSMShsm is not available in your environment, Log Master might not be able to honor the DASD DATASETS limit value.

Enter a number to limit the number of data sets that are maintained on DASD. To avoid imposing a limit, enter 0. The default value is determined by the DASDDSNS installation option (see “DASDDSNS=10” on page 81).

**PER MEMBER**

Specifies the scope of the DASD DATASETS value. This keyword applies only in a data sharing environment.

**YES**
Indicates that the value is a maximum limit for each member of a data sharing environment. This is the default value.

**NO**
Indicates that the value is a maximum limit for the entire DB2 subsystem, or the entire data sharing group.

See also “DDPRMBR=YES” on page 81.

**GENERATE MASSDELETE**

Determines the output that Log Master generates when it encounters DB2 log records that reflect a LOAD REPLACE action. When a DB2 Load utility runs with the REPLACE option, the DB2 log contains log records that Log Master can interpret as a “mass delete” action (similar to a DELETE statement with no WHERE clause or a TRUNCATE statement). This keyword controls whether Log Master includes the mass delete action in the generated output.

**YES**
Directs Log Master to include the mass delete action in the generated output. This is the default value.
**OPTION statement**

**NO**
Prevents Log Master from including the mass delete action. (For example, mass delete actions might not be appropriate for auditing or historical databases.)

For more information about LOAD REPLACE actions, see “GENMDEL=YES” on page 68.

**PROCESS COLD START URIDS**

Determines whether Log Master processes transactions that were in process (but were not terminated) when a DB2 subsystem was cold started. In this context, transactions are considered to be the same as URIDs. Use this keyword to obtain information about the unterminated transactions (for example, in a report). Specify a scan range that includes DB2 processing before and after the cold start.

**YES**
Directs Log Master to process transactions that were not terminated when the subsystem was cold started.

**NO**
Directs Log Master to ignore transactions that were not terminated when the subsystem was cold started. This is the default value.

Normally, Log Master does not process a transaction until that transaction is either committed or aborted. Because of this behavior, Log Master does not include transactions that are interrupted by a cold start in generated output. Similarly, Log Master can retain unterminated transactions in the Repository, causing ongoing log scans to read more log files than necessary as they search for commit or abort actions.

However, when you include this keyword, Log Master uses conditional restart information contained in the bootstrap data set (BSDS) to complete the following actions:

- Locate all unterminated transactions on the subsystem before the cold start.
- Mark the transactions as committed or aborted within the Log Master internal control structures.

Log Master can then process transactions, select them, include them in any generated output, and mark them as committed in the Repository.

**HEURISTIC FORWARDCOMPLETION**

Determines whether Log Master uses a key store to perform heuristic completion. A key store is a Log Master internal memory and file structure. Heuristic completion is a special type of row completion processing that is separate from and different than the more extensive row completion processing that Log Master performs. For more information about row completion, see “IMAGESOURCE” on page 124.
Heuristic completion imposes a small amount of overhead during the initial log scan to decrease the possibility of more overhead after the initial log scan. (After the initial scan, Log Master performs more row completion processing, which can include reading table spaces, reading image copies, or reading more log files). Depending on your environment, disabling heuristic completion can increase or decrease the performance of Log Master. To determine the effects of heuristic completion in your environment, examine your job’s output for a series of messages that start with message BMC97396 and list the key store as FCUSE.

The following conditions influence the results:

- Whether you define DB2 objects with Data Capture Changes (DCC).
- The patterns of insert, update, and delete activity on your DB2 objects.
- The amount of memory that you allocate to Log Master.
- The percentage of memory that you allocate to different key stores (for more information, see “MEMPERCENT” on page 142).

**NOTE**
BMC recommends that you consult BMC Software Customer Support before changing the default value.

**YES**
Directs Log Master to perform heuristic completion. This is the default value.

**NO**
Prevents Log Master from performing heuristic completion.

See also “FCUSE=YES” on page 60.

**LOGRECORD BUFSIZE nnn**

Determines the maximum amount of memory that Log Master uses for the log record buffer. If Log Master fills up this amount of memory, and more log records remain to be processed, Log Master writes log records (spills) to DASD storage; performance can be degraded.

The log record buffer is an internal memory and file structure. During its initial scan of the DB2 log, Log Master stores images of log records in this buffer when it has encountered the start of a transaction (unit of recovery), and has not yet encountered the end of the transaction.

Specify the amount of memory in bytes, kilobytes (use the suffix K), or megabytes (use the suffix M). The default value is 48M. The minimum value is 512K.

Be aware of the following points regarding the log record buffer:
This buffer is dynamic. Log Master uses only the memory and DASD storage that it needs. Log Master reuses memory and DASD storage as it encounters the ends of transactions.

To allocate data sets when Log Master writes log records (spills) to DASD storage, Log Master uses the values of the STOREOPTS statement keywords (or their corresponding installation options).

BMC Software recommends retaining the default value unless you experience repeated performance problems that are related to memory usage. For help with LOGRECORD BUFSIZE changes, contact BMC Software Customer Support.

See also “QBLRBUF=48M” on page 55.

**RESOURCE SELECTION**

Use the RESOURCE SELECTION option to specify the order in which Log Master uses image copy resources for both completion processing and compression dictionary access for data decompression.

**COPIES**

You can specify RESOURCE SELECTION COPIES as FC, LP, LB, RP, and RB in any order. LP and LB indicate the primary and secondary local image copies. RP and RB indicate the primary and secondary remote image copies. FC indicates an IBM FlashCopy. The default order is (FC, LP, LB) when operating as a local site, and (RP, RB, FC) when operating in a remote site. You can specify one to five values from FC, LP, LB, RP, and RB in the sequence in any order. You can omit references to copies of the resource that you do not want considered.

See also “LOCCPSEL=(FC,LP,LB)” on page 85 and “REMCPSEL==(RP,RB,FC)” on page 85.

**LOGS**

Specifies the order in which Log Master reads active and archive log files. For all log scans, Log Master searches the log files in the order that you enter. Specify log keywords in any order. Omit keywords for log files that you do not want Log Master to consider. If you omit keywords, Log Master can terminate with error messages if it cannot find required log records in the log files that you have specified. The default order is ACT1, ACT2, ARCH1, and ARCH2. You must specify at least one of the values.

Use OPTIONS RESOURCE SELECTION LOGS to specify a value to override the USELOGS installation option value (page 62).
Overtime mode

Log Master provides the ability to run in overtime mode. In overtime mode, Log Master reads all of the log records that are related to selected objects, regardless of whether the objects exist in the DB2 catalog. In normal operation, (called current mode), Log Master reads the DB2 catalog to get information about the structure of selected objects. However, when an object is dropped, DB2 deletes all references to the object from the DB2 catalog. In overtime mode, Log Master must use other sources to obtain structure definitions.

Use overtime mode when the current time is after a drop action or a drop and re-create action, but you need to retrieve log records that were written before the action.
The following considerations apply to overtime mode:

- An overtime job typically uses more resources and experiences more processing overhead than a job that runs in current mode.

- Log Master refers to dropped DB2 objects (or DB2 objects that have been dropped and re-created) as old objects. Overtime mode enables Log Master to process log records that are related to old objects.

- In overtime mode, Log Master uses other sources to obtain structure definitions of old objects. You must perform at least one of the following types of extra processing to update these sources:
  - Run periodic jobs that update the Log Master Repository with structure definitions for your old objects (proactive method).
  - Run an extra log scan that updates the Repository immediately before you use overtime mode (reactive method).
  - Perform manual research and data entry to create an old objects data set.

- Log Master refers to the version of a DB2 object that exists between a create action and the following drop action as an instance of that object. Depending on the time frame of your log scan, and the times when an object is dropped and re-created, Log Master can encounter log records that are related to multiple instances of the same object.

- By default, Log Master does not perform row completion processing when it runs in overtime mode. Optionally, you can specify the ATTEMPT COMPLETION keyword and the IMAGECOPY statement to direct Log Master to perform row completion processing using available image copy data sets.

- Overtime mode is not the same as the Log Master automated drop recovery feature. Drop recovery restores dropped objects. Overtime mode enables you to retrieve data from the DB2 log that is associated with old objects, and to generate output that is based on the data. Overtime mode does not restore the old objects.

- If you use the proactive method to update the Repository, schedule the update jobs to run before the following processing:
  - Regular production processing. For example, if you run a set of jobs every week, you should run a job to update the repository before you run the weekly processing jobs.
  - DB2 Load or Reorg actions that update compression dictionaries, or that might assign table rows to different record ID (RID) values.
If you update the Repository regularly, BMC Software recommends that you also run regular jobs to delete old or unusable data (particularly any compression dictionaries) from the Old Objects Table. Alternately, you can delete (or display) information from this table by using an option on the Main Menu of the Log Master online interface.

**WARNING**

If you store compression dictionaries in the repository, and then stop updating the repository, delete any residual compression dictionaries. Using an outdated compression dictionary from the repository can cause Log Master to fail with a S0C7 abend in member LZCOMPRS, which Log Master uses for decompressing data.

For system-time temporal tables, the Log Master Repository does not maintain the versioning relationship between the system-maintained table and its associated history table. Therefore, for overtime processing, your filter must include both the base table and the history table.

For more information about overtime mode, see the chapter on processing objects over time in the *Log Master for DB2 User Guide*.

**LOGOPTS statement**

Figure 6 shows the LOGOPTS statement syntax. The LOGOPTS statement controls how Log Master processes DB2 log files. The keywords of this statement correspond to installation options that are described in Chapter 1, “Operational considerations and installation information.”

![LOGOPTS statement syntax diagram](image)
LOGOPTS statement

LOGSORT

Determines how Log Master processes DB2 log records in a data sharing environment. Specify one of the following sort modes.

WHenever

Directs Log Master to use the value of the LOGTAPES keyword to determine whether to sort multiple DB2 log files. If you specify this value, Log Master uses the LOGTAPES value, regardless of the type of storage device on which the log data sets are stored. This is the default value.

The following table summarizes whether Log Master sorts multiple DB2 log files in data sharing and non–data sharing environments.

<table>
<thead>
<tr>
<th>LOGTAPES value</th>
<th>Data sharing system</th>
<th>Non–data sharing system</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGTAPES = 0</td>
<td>Merge</td>
<td>Merge</td>
</tr>
<tr>
<td>LOGTAPES &lt; (number of data sharing members with tape logs)</td>
<td>Sort</td>
<td>Merge</td>
</tr>
<tr>
<td>LOGTAPES &gt;= (number of data sharing members with tape logs)</td>
<td>Merge</td>
<td>Merge</td>
</tr>
</tbody>
</table>

NEVER

Directs Log Master to avoid sorting the log (by processing all log data sets concurrently). When LOGSORT is NEVER, Log Master ignores any LOGTAPES value, and allocates as many units as it needs to perform a merge (not a sort) action. Depending on the number of data sharing members and the number of tape units in your environment, this setting can cause contention for tape units.

ALWAYS

Directs Log Master to always sort the log. Log Master does not attempt to process multiple log data sets concurrently. If LOGSORT is ALWAYS, Log Master uses LOGTAPES only to determine how many storage devices it can use to read log records for the sort action.

See also “LOGSORT=WHENEVER” on page 63.
LOGTAPES (maximumUnits)

Specifies the maximum number of tape units or DASD data sets that Log Master allocates to read log files at one time. This value applies only in a data sharing environment. (LOGTAPES does not limit the number of physical tape devices that Log Master can use, only the number of tape units that Log Master allocates for log processing at one time.) The default value is 0.

**NOTE**

Log Master allocates the value that you specify, or a value equal to the number of members in the data sharing group, whichever value is less.

When the LOGSORT installation option is set to NEVER, Log Master ignores any LOGTAPES value. For more information about how the LOGTAPES and LOGSORT keywords interact, see “LOGSORT” on page 138.

Values used for the maximum number of units or data sets are as follows:

*nnnn*

Any numeric value between one and the total number of tape drives accessible to the system.

*0*

Log Master automatically sets LOGTAPES to the number of members within the data sharing group.

See also “LOGTAPES=0” on page 64.

STOREOPTS statement

Figure 7 on page 140 shows the STOREOPTS statement syntax. The STOREOPTS statement controls how Log Master maintains internal memory and file structures called key stores. These internal structures are built and maintained in memory, but if the allocated memory space fills up while more records remain to be processed, the internal key store can spill to DASD storage.

The STOREOPTS statement affects both the amount of memory that is allocated for key stores, and the data sets that Log Master creates when it spills to DASD storage. The keywords of this statement correspond to installation options described in Chapter 1, “Operational considerations and installation information.”
MEMORY(<data>, <index>)

Defines the total amount of memory that Log Master uses for key stores. If you specify this keyword, ensure that it is the first keyword in the STOREOPTS statement. The keyword has the following values:

<data>
Specifies the amount of memory that Log Master uses to store internal data as it processes log records. Enter an amount of memory in kilobytes (use the suffix K) or megabytes (use the suffix M). You might need to override the default value of this keyword frequently, because the amount of memory required for efficient processing can increase or decrease dramatically based on how many DB2 log data records that Log Master is processing.

<index>
Specify a numeric value. Log Master does not currently use this value, but reserves it for future use, and requires it to be present for correct syntax. Enter a number as bytes, kilobytes (use the suffix K) or megabytes (use the suffix M). If you specify <data> without <index>, the Log Master installation option assembly macro produces a syntax error and terminates processing with return code 8.
Log Master distributes the memory that you allocate between key stores based on the values that are specified for the MEMPERCENT keyword (page 142) or the xxPCT installation options (page 53).

See also “KSMEMORY=100M, 10M” on page 49.

**TRACKS | CYLINDERS**

Specifies how Log Master allocates internal key store overflow data sets.

- **TRACKS**
  The overflow data sets are allocated in tracks.

- **CYLINDERS**
  The overflow data sets are allocated in DASD cylinders.

See also “KSALLOCU=CYLS” on page 50.

**SPACE (priSpace, secSpace)**

Specifies how Log Master allocates disk space for internal key store overflow files. Use this keyword along with the CYLINDERS or TRACKS keyword. Log Master allocates these data sets in the same way that VSAM data sets are allocated in JCL. Log Master allocates priSpace first. If Log Master needs more space, it allocates an additional amount of secSpace (often called an extent). Log Master can allocate up to 128 extents.

See also “KSSPACE=(100,100)” on page 50.

**STORCLAS (storageClass)**

Specifies the Data Facility Storage Management System (DFSMS) storage class that Log Master uses to store internal key store overflow files.

See also “KSSTOR=” on page 51.

**DATACLAS (dataClass)**

Specifies the DFSMS data class that Log Master uses to store internal key store overflow files.

See also “KSDACLSS=” on page 52.
STOREOPTS statement

**MGMTCLAS** *(mgmtClass)*

Specifies the DFSMS management class that Log Master uses to store internal key store overflow files.

See also “KSMGMT=” on page 52.

**VOLUME** *(volser)*

Specifies the volumes that Log Master uses to store internal key store overflow files. Enter a list of valid volsers. Separate multiple volsers with commas. Log Master uses the volumes that you specify in an IDCAMS DEFINE CLUSTER command.

See also “KSVOLS=” on page 53.

**CLUSTER** *(clusterHLQ)*

Defines a prefix that Log Master uses in the cluster name of a VSAM data set. Log Master allocates this data set when it spills to DASD storage.

See also “KSCLUST=&&SYSUID..CLUSTER” on page 50.

**DATA** *(dataHLQ)*

Defines a character string that Log Master uses as the high-level qualifier of a VSAM data set. Log Master allocates the data set as overflow storage when it spills to DASD storage.

See also “KSDATA=&&SYSUID..DATA” on page 51.

**MEMPERCENT**

Determines how Log Master distributes memory among its individual key stores. Each value represents a percentage of the total amount of key store memory. For more information, see “MEMORY(data, index)” on page 140.
NOTE
BMC Software recommends that you do not change key store memory distribution unless you experience repeated performance problems related to key store use (for example, if insufficient memory frequently causes Log Master to spill to DASD storage).

For each key store that it uses, Log Master displays sets of output messages that contain allocation and use information. Use this information to distribute key store memory optimally for your data and environment. The allocation messages begin with message BMC097390. The usage messages begin with message BMC097396.

For more information about redistributing key store memory, see the chapter on Log Master for DB2 expert information in the Log Master for DB2 User Guide, or contact BMC Software Customer Support.

If you use the MEMPERCENT keyword, you must specify percentage values for each key store as an integer, using the following guidelines:

- URID and LOGRECORD are required; the valid range is 1 through 99.
- FORWARDCOMPLETION, BACKWARDCOMPLETION, and ANOMALY are optional; the valid range is 0 through 98.
- The percentage values must add up to 100 percent (including any default values for key stores that you do not specify).
- To avoid confusion, specify a percentage value for each key store, even if the value is 0 or unchanged from the default value.
- Do not specify any key store more than once.

**URID**
Specifies the percentage of key store memory that is allocated to the URID (UR) key store. In the UR key store, Log Master maintains information about each unit of recovery that it encounters within the range of a log scan. If you do not specify this percentage value, Log Master uses the value of the URPCT installation option (page 53).

When your job reads input from a logical log file, Log Master redistributes all available key store memory to the UR key store, regardless of your specifications. Log Master takes this action because it does not require the other key stores to process logical log input.

**LOGRECORD**
Specifies the percentage of key store memory that is allocated to the log record (LR) key store. In the LR key store, Log Master maintains a copy of log records that require row completion processing. If you do not specify this percentage value, Log Master uses the value of the LRPCT installation option (page 53).
FORWARDCOMPLETION
Specifies the percentage of key store memory that is allocated to the forward completion (FC) key store. In the FC key store, Log Master maintains copies of complete row images that might be used for row completion processing. If you do not specify this percentage value, Log Master uses the value of the FCPCT installation option (page 54).

BACKWARDCOMPLETION
Specifies the percentage of key store memory that is allocated to the backward completion (BC) key store. In the BC key store, Log Master maintains copies of complete row images that might be used for row completion processing. If you do not specify this percentage value, Log Master uses the value of the BCPCT installation option (page 54).

ANOMALY
Specifies the percentage of key store memory that is allocated to the anomaly (AN) key store. In the AN key store, Log Master maintains information that is used to produce the Backout Integrity report. If you do not specify this percentage value, Log Master uses the value of the ANPCT installation option (page 54). When your job does not include a Backout Integrity report, Log Master disregards your specification and redistributes the memory of the AN key store to the FC and BC key stores.

LOBOPTS statement

Figure 8 on page 145 shows the LOBOPTS statement syntax. The LOBOPTS statement controls how Log Master maintains the LOB VSAM files that it uses to store large object (LOB) data. Log Master creates one set of VSAM files for each LOB column that occurs in the log records that are selected by a log scan. If a LOB column occurs in a partitioned table space, Log Master creates one set of VSAM files for each partition of each LOB column. Log Master deletes the LOB VSAM files at the end of processing unless the log scan includes an output logical log file.

The LOB VSAM files are different than the external files that Log Master can write when a LOB column is included in generated output load files. For more information about the external files, see “TEMPLATE” on page 245.
Figure 8  LOBOPTS statement syntax diagram

**TRACKS | CYLINDERS**

Specifies the units that Log Master uses to allocate LOB VSAM files.

**TRACKS**
The LOB VSAM files are allocated in tracks.

**CYLINDERS**
The LOB VSAM files are allocated in DASD cylinders.

See also “LOBALLOC=CYLINDERS” on page 69.

**SPACE (priSpace, secSpace)**

Specifies how Log Master allocates disk space for LOB VSAM files. Use this keyword along with the CYLINDERS or TRACKS keyword. Log Master allocates these data sets in the same way that VSAM data sets are allocated in JCL. Log Master allocates `priSpace` first. If Log Master needs more space, it allocates an additional amount of `secSpace` (often called an extent). Log Master can allocate up to 128 extents.

See also “LOBSPACE=(10,10)” on page 72.
LOBOPTS statement

VOLUME (volser)

Specifies the volumes that Log Master uses to store LOB VSAM files. Enter a list of valid volsers. Separate multiple volsers with commas. Log Master uses the volumes that you specify in an IDCAMS DEFINE CLUSTER command.

See also “LOBVOLS=” on page 72.

STORCLAS (storageClass)

Specifies the Data Facility Storage Management System (DFSMS) storage class that Log Master uses to store LOB VSAM files.

See also “LOBSTOR=” on page 72.

MGMTCLAS (mgmtClass)

Specifies the DFSMS management class that Log Master uses to store LOB VSAM files.

See also “LOBMGMT=” on page 71.

DATACLAS (dataClass)

Specifies the DFSMS data class that Log Master uses to store LOB VSAM files.

See also “LOBDATA=” on page 69.

PREFIX (clusterPrefix)

Defines a prefix that Log Master uses in the names of LOB VSAM files. Log Master constructs the data set name by using this prefix, the database identifier (DBID) of the LOB column, the page set identifier (PSID) of the column, and a sequentially incremented data set number. To avoid duplicate data sets during repeated log scans, consider using symbolic values such as &DATE. and &TIME. in the prefix that you define.

See also “LOBPREF=&SYSUID..LOB” on page 71.

DUPLICATE DATASET

Determines how Log Master responds to a duplicate data set condition as it attempts to create a LOB VSAM file. This keyword determines the severity of the message that Log Master displays, and whether Log Master continues processing. To avoid duplicate data sets during repeated log scans, consider using symbolic values such as &DATE. and &TIME. in the prefix that you define with the PREFIX keyword.
INFO
Specifies that Log Master issues an informational message, deletes the existing file, creates a new file, and continues processing.

WARN
Specifies that Log Master issues a warning message, deletes the existing file, creates a new file, and continues processing.

ERROR
Specifies that Log Master issues an error message, saves any temporary LOB VSAM files permanently to disk, and terminates processing. This is the default value.

See also “LOBDUPDS=ERROR” on page 70.

LOBLIMIT nn
Determine the maximum number of LOB VSAM files that Log Master can create to store the LOB data of one LOB column (or one partition of one LOB column). Log Master allocates a new LOB VSAM file only when it has filled the initial data set and all possible extents, but more LOB column data remains to be written. If the number of data sets (with full extents) that are required to store a LOB column’s data exceeds this limit, Log Master issues an error message, saves any temporary LOB VSAM files permanently to disk, and terminates processing.

Specify a number of data sets between 1 and 26. The default value is 10.

See also “LOBLIMIT=10” on page 70.

XMLOPTS statement

Figure 9 on page 148 shows the XMLOPTS statement syntax. The XMLOPTS statement controls how Log Master maintains the XML VSAM files that it uses to store data from XML columns. Log Master creates one set of VSAM files for each XML column that occurs in the log records that are selected by a log scan. If an XML column occurs in a partitioned table space, Log Master creates one set of VSAM files for each partition of each XML column. Log Master deletes the XML VSAM files at the end of processing unless the log scan includes an output logical log file.

The XML VSAM files are different than the external files that Log Master can write when an XML column is included in generated output load files. For more information about the external files, see “TEMPLATE” on page 245.
**NOTE**
Log Master does not support partially logged XML documents that are associated with XML multi-versioning and XML update processing.

**Figure 9  XMLOPTS statement syntax diagram**

**TRACKS | CYLINDERS**

Specifies the units that Log Master uses to allocate XML VSAM files.

**TRACKS**
The XML VSAM files are allocated in tracks.

**CYLINDERS**
The XML VSAM files are allocated in DASD cylinders.

| See also “LOBALLOC=CYLINDERS” on page 69.

**SPACE (priSpace, secSpace)**

Specifies how Log Master allocates disk space for XML VSAM files. Use this keyword along with the CYLINDERS or TRACKS keyword. Log Master allocates these data sets in the same way that VSAM data sets are allocated in JCL. Log Master allocates priSpace first. If Log Master needs more space, it allocates an additional amount of secSpace (often called an extent). Log Master can allocate up to 128 extents.

| See also “LOBSPACE=(10,10)” on page 72.
VOLUME (volser)

Specifies the volumes that Log Master uses to store XML VSAM files. Enter a list of valid volsers. Separate multiple volsers with commas. Log Master uses the volumes that you specify in an IDCAMS DEFINE CLUSTER command.

See also “LOBVOLS=” on page 72.

STORCLAS (storageClass)

Specifies the Data Facility Storage Management System (DFSMS) storage class that Log Master uses to store XML VSAM files.

See also “LOBSTOR=” on page 72.

MGMTCLAS (mgmtClass)

Specifies the DFSMS management class that Log Master uses to store XML VSAM files.

See also “LOBMGMT=” on page 71.

DATACLAS (dataClass)

Specifies the DFSMS data class that Log Master uses to store XML VSAM files.

See also “LOBDATA=” on page 69.

PREFIX (clusterPrefix)

Defines a prefix that Log Master uses in the names of XML VSAM files. Log Master constructs the data set name by using this prefix, the database identifier (DBID) of the XML column, the page set identifier (PSID) of the column, and a sequentially incremented data set number. To avoid duplicate data sets during repeated log scans, consider using symbolic values such as &DATE. and &TIME. in the prefix that you define.

See also “XMLPREF=&&SYSUID..XML” on page 73.

DUPLICATE DATASET

Determines how Log Master responds to a duplicate data set condition as it attempts to create an XML VSAM file. This keyword determines the severity of the message that Log Master displays, and whether Log Master continues processing. To avoid duplicate data sets during repeated log scans, consider using symbolic values such as &DATE. and &TIME. in the prefix that you define with the PREFIX keyword.
INFO
Specifies that Log Master issues an informational message, deletes the existing file, creates a new file, and continues processing.

WARN
Specifies that Log Master issues a warning message, deletes the existing file, creates a new file, and continues processing.

ERROR
Specifies that Log Master issues an error message, saves any temporary XML VSAM files permanently to disk, and terminates processing. This is the default value.

INFO
Specifies that Log Master issues an informational message, deletes the existing file, creates a new file, and continues processing.

WARN
Specifies that Log Master issues a warning message, deletes the existing file, creates a new file, and continues processing.

ERROR
Specifies that Log Master issues an error message, saves any temporary XML VSAM files permanently to disk, and terminates processing. This is the default value.

See also “LOBDUPDS=ERROR” on page 70.

XMLLIMIT nn

Determines the maximum number of XML VSAM files that Log Master can create to store the data of one XML column (or one partition of one XML column). Log Master allocates a new XML VSAM file only when it has filled the initial data set and all possible extents, but more XML column data remains to be written. If the number of data sets (with full extents) that are required to store an XML column’s data exceeds this limit, Log Master issues an error message, saves any temporary XML VSAM files permanently to disk, and terminates processing.

Specify a number of data sets between 1 and 26. The default value is 10.

See also “LOBLIMIT=10” on page 70.

SORTOPTS statement

Figure 10 on page 151 shows the SORTOPTS statement syntax. The SORTOPTS statement controls how Log Master dynamically allocates the memory and data sets that are related to sort actions (including sort routine memory and sort work data sets). The keywords of this statement correspond to installation options described in Chapter 1, “Operational considerations and installation information.”
SMCORE

Specifies the maximum amount of memory that Log Master requests when it uses the sort routine. To specify this keyword, enter both of the following values.

**mainSize**
Specifies the total amount of memory that Log Master sort actions can use (both above and below the 16-MB line).

**vscoret**
Specifies the amount of memory that Log Master sort actions can use below the 16-MB line.

See also “SMCORE=0M,0K” on page 46.

RESINV *memValue*

Specifies the amount of memory (below the 16-MB line) that Log Master instructs the sort routine to reserve for processing other than sort processing. Specify the *memValue* amount as bytes, kilobytes (use the suffix K), or megabytes (use the suffix M).

See also “RESINV=0K” on page 47.
PARMOPTS

Specifies how Log Master dynamically allocates data sets other than sort work data sets.

STORCLAS (storageClass)
Specifies the storage class that Log Master uses for storage of other dynamically allocated data sets. Use this keyword only with DFSMS. Do not specify a value of VIO.

See also “PARMSTOR=” on page 48.

UNIT (unitName)
Specifies the storage device (or group of devices) where Log Master requests space for other dynamically allocated data sets. Enter any valid unit name or a generic unit name where you can allocate temporary data sets. Do not use this keyword with DFSMS.

See also “PARMUNIT=” on page 48.

SORTDYN

Determines whether Log Master uses the sort routine’s dynamic sort allocation rules or accepts your allocation attributes.

SYSTEM
Directs Log Master to use the sort routine’s rules.

USE
Enables you to enter allocation attributes. This is the default value.

See also “SORTDYN=USE” on page 49.

Sort file size parameters

Specifies the estimated size of the data that Log Master must sort during a given job. You can also specify the size of the data to be sorted during an individual log scan (page 174). These parameters specify the estimated size of the data to be sorted or a technique that Log Master uses to calculate the estimate.

For more information, see “Sort file size parameters” on page 154.
HISTORY

Determines whether Log Master stores information in the Repository about the current job’s sort actions. The information includes the number of log records that were sorted and the average size of the records. If historical information is stored in the Repository from previous runs of the current work ID, Log Master can use that information to estimate the size of the data to be sorted during the current run. For more information, see “FILSZ” on page 154. Log Master passes the estimated size to the sort routine in your environment.

Log Master stores and retrieves historical information by using the work ID of the job. If you run a Log Master job repeatedly, you can use this keyword (and the related USE HISTORY value) to optimize the sort actions of your job.

WRITE
Specifies that Log Master writes information about the current job’s sort actions in the Repository. This is the default value.

NOWRITE
Specifies that Log Master does not write information in the Repository.

WORKOPTS

Specifies how Log Master dynamically allocates sort work data sets.

NUM (numOfFiles)
Specifies how many sort work data sets Log Master allocates for each sort action. You can specify any number from 1 to 255. The default value is 0.

See also “WORKNUM=0” on page 48.

STORCLAS (storageClass)
Specifies which storage class Log Master uses for storage of sort work data sets. Use this keyword only if you want DFSMS to manage the sort work data sets.

See also “WKSTOR=” on page 48.

UNIT (unitName)
Specifies the storage device (or group of devices) where Log Master requests space for sort work data sets. Enter any valid unit name or a generic unit name where you can allocate temporary data sets. Do not use this keyword if you want DFSMS to manage the sort work data sets.

See also “WKUNIT=” on page 48.
Sort file size parameters

Figure 11 shows the Sort file size parameters syntax that occurs in both the SORTOPTS and LOGSCAN statements. Use this syntax to improve the performance of Log Master sort actions as follows:

- When these parameters are included in a SORTOPTS statement, they affect the sort actions of the entire job.

- When these parameters are included in a LOGSCAN statement, they affect only the individual log scan.

You can also use the following methods to affect the performance of sort actions:

- You can control how much memory Log Master allocates for each invocation of the sort routine (for more information, see “SMCORE=0M,0K” on page 46).

- You can control how the sort routine allocates sort work data sets (for more information, see “Defining sort work data sets” on page 102).

**FILSZ**

Specifies the estimated size of the data that Log Master must sort during a given job or log scan. The value of this keyword determines the estimated size or the technique that Log Master uses to calculate an estimated size. Log Master passes the estimated size to the sort routine in your environment. The sort routine uses the estimated size value to efficiently allocate sort memory or sort work data sets.
NONE

Directs Log Master not to pass any file size estimate to the sort routine. When this value is NONE, the sort routine in your environment uses its own default settings to allocate memory and sort work data sets.

For compatibility with earlier versions of Log Master, this is the default value.

USE HISTORY

Directs Log Master to calculate a file size estimate based on historical information about the current work ID. To use this value, you must have previously directed Log Master to write the historical information in the Repository (using the related HISTORY keyword of the SORTOPTS statement). The historical information describes the sort actions of previous runs of the current work ID (including the number of log records sorted and the average size of the records).

Log Master calculates the estimate based on a weighted average of previous runs, giving greater weight to more recent runs. If you run a Log Master job repeatedly, you can employ the USE HISTORY value (and the related HISTORY keyword) to optimize the sort actions of your job. For more information, see “HISTORY” on page 153.

numOfRuns

Determines how many runs Log Master includes in its calculation of the file size estimate. The minimum value for numOfRuns is 1, the default value is 5, and the maximum value is 20.

LOGBYTES

Directs Log Master to calculate a file size estimate based on the total volume of the log records that the current work ID processes. Log Master examines the total volume of the log records (expressed as a number of bytes), not the number of log records.

Use the following keyword to give Log Master additional information that it can use to calculate a more precise file size estimate.

USE HISTORY

To calculate a file size estimate, Log Master uses the total volume of log records and historical information stored in the Repository about previous runs of the current work ID. The historical information includes the number of log records and the average length of the log records for previous runs. This is the default value.
**EST numOfRecs**

Directs Log Master to use the value that you enter for **numOfRecs** as an estimate of the number of log records to be sorted. Log Master passes the estimated value to the sort routine in your environment.

**AVGRECLEN lenValue**

Specifies the average record length of the log records to be sorted. Use this keyword to pass more precise file size estimate to the sort routine in your environment.

If you specify a FILSZ estimate, Log Master determines the default SMCORE values by using the minimum storage required by BMCSORT when all of the following conditions exist:

- You accept the default values of 0 for the SMCORE installation option.
- You specify a value for the FILSZ keyword in the SORTOPTS statement.
- You do not specify SMCORE values in the SORTOPTS statement.

---

**WORKID statement**

Figure 12 shows the WORKID statement syntax. The WORKID statement specifies a unique identifier for a unit of work performed within Log Master. Log Master keeps track of all job steps and actions using a combination of a job’s user ID and the work ID defined in this statement. BMC Software strongly recommends that you define different jobs with a unique combination of user ID and work ID, particularly if you run multiple jobs concurrently or if you define any jobs as ongoing processes.

**Figure 12 WORKID statement syntax diagram**

```
WORKID statement
WORKID (page 156) workID (page 157) DESC "description" (page 157)
```

**userID.**

Specifies an identifier representing the user or entity that started the current job. A user ID can be up to eight characters long. The default value of this keyword is your TSO prefix or TSO user ID. Separate the user ID from the work ID with a period.
workID

Specifies an identifier that Log Master associates with all work performed in the current job or job step. The following rules apply to work IDs:

- The name can be up to eighteen characters long.
- The first character must be an alphabetic or national ($#@) character.
- Subsequent characters can be alphabetic, numeric, or national characters.

The Log Master online interface assigns a default work ID name that contains the following sequence of information: a user ID, period, the character string $$WORKID, a four-digit sequence number. For example:

| RDAUSER.$$WORKID0001 |

The default work ID name is reserved for use by Log Master. You cannot create a work ID name that begins with the characters $$WORKID.

DESC "description"

Specifies up to 64 characters of comments that you can associate with the current work ID.

**IMAGECOPY statement**

*Figure 13 on page 158* shows the IMAGECOPY statement syntax. The IMAGECOPY statement specifies an exact location for input image copies.

Normally, Log Master uses the DB2 catalog to locate input image copies, and then uses the image copies to perform row completion or to obtain compression dictionaries. Use this statement when the record of an image copy has been deleted from the SYSIBM.SYSCOPY table of the DB2 catalog, but the image copy data sets themselves are still available. If you specify the image copy location, Log Master can use the image copies directly. You can specify multiple input image copies with one IMAGECOPY statement.
**TABLESPACE NAME**

Indicates that Log Master uses input from an image copy that contains the table space represented by `tableSpaceName`.

**DSNDB04 | databaseName**

Specifies the database associated with an input image copy.

**DSNDB04**

Directs Log Master to obtain input from an image copy associated with DB2’s default database. This is the default value.

**databaseName**

Directs Log Master to obtain input from an image copy associated with the database defined by `databaseName`.

**tableSpaceName**

Specifies the table space that is contained in an input image copy.

**Image copy parameters**

Specifies the location and other attributes of an input image copy.
Image copy parameters

Figure 14 shows the Image copy parameters syntax of the IMAGECOPY statement. The Image copy parameters specify the location and other attributes of an input image copy. The information from these parameters enables Log Master to use the image copy to perform row completion or to obtain a compression dictionary. Log Master can obtain dictionaries from the following sources, listed in order of preference:

- current table space
- DB2 log
- Old Objects Table of the Repository
- image copies of the table space

**NOTE**

Log Master cannot use DFSMS concurrent image copies. For more information, see “Reading image copies” on page 125.

**Figure 14** Image copy parameters syntax diagram

**DATASET dataSetName**

Specifies the name of a data set that contains the input image copy.

**IMRBA x 'imrba'**

Specifies the RBA/LRSN of the log point when the input image copy was created.

**DSNUM 0 | DSNUM n**

Specifies a partition of a partitioned table space that is contained in the image copy.
Image copy parameters

**DSNUM 0**
Indicates that the input image copy contains pages for all partitions of the associated table space. This is the default value.

**DSNUM n**
Indicates that the input image copy contains only the pages for the specified partition of the associated table space.

**ICTYPE**
Specifies the type of the input image copy.

**FULL**
Specifies that the input image copy is a full image copy. This is the default value.

**INLINE**
Specifies that the input image copy is an inline (or embedded) image copy, created during a load or reorganization action of the associated database.

**INCR**
Specifies that the input image copy is an incremental image copy.

**DEVT deviceType**
Specifies the type of the device that contains the input image copy. Enter a site-specific name for a type of storage device (for example, 3390 for a DASD device, or TAPE for a magnetic tape device). Use this keyword only if the image copy data set is uncataloged.

**VOLSER (volser)**
Specifies the VOLSER of the device that contains the input image copy. To enter multiple VOLSERs, separate each with a comma. Use this keyword only if the data set containing the input image copy is uncataloged.

**SEQNO n**
Specifies the sequence number if the input image copy is located on a tape volume. Use this keyword only if the data set containing the input image copy is uncataloged.
REMAP

Indicates that the input image copy occurred immediately after an event that remapped (modified) the record IDs (RIDs) in the table space. Typically, DB2 remaps RIDs when you run a DB2 Load or Reorg utility program.

INPUT statement

Figure 15 shows the INPUT statement syntax. The INPUT statement specifies the source where Log Master reads DB2 log records. This statement defines the input source for all log scans within the current work ID.

NOTE

For compatibility with earlier versions, Log Master also supports an INPUT keyword on the LOGSCAN statement. This keyword is deprecated and might be removed from future versions. (It defines the input source for an individual log scan, but the input source must always be the same for all log scans within a job step.)

Figure 15  INPUT statement syntax diagram

BSDS

Directs Log Master to scan the current DB2 log files that are listed in the bootstrap data set (BSDS) of DB2. This is the default value.

LLOG(cntlDataSetName)

Directs Log Master to read log records from one or a set of logical log files previously created by Log Master or an application program that conforms to the published format. Enter the name of the data set that contains the control file of the logical log. Log Master uses information in the control file to locate and read the logical log data file, and any other files associated with the logical log.
Enter a name that follows conventional guidelines for data set names. Separate multiple logical log data sets with commas. Be aware of the following points as you specify input logical log files:

- If you enter a GDG base as `cntlDataSetName`, Log Master includes all available generations of the file in the input.

- If you specify more than one logical log control file as input (directly or by specifying a GDG base), Log Master automatically runs in overtime mode.

- If you specify input logical log files, you cannot define an ongoing log scan.

**DB2LOG**

Directs Log Master to read log records from a specific DB2 log file or set of DB2 log files. You can define the exact location of the DB2 log files. For more information about specifying the DB2 log files, see “DB2 log input definition.”

**DB2 log input definition**

Figure 16 shows the DB2 log input definition syntax used in the INPUT statement. Use this syntax to define a specific DB2 log file or set of DB2 log files where Log Master obtains DB2 log records.

You can define the exact location of the DB2 log files. If you specify more than one log file, you must specify the files in the order defined by the ULOGORD installation option. (For more information, see “ULOGORD=ASCENDING” on page 62.)
Log Master either sorts input DB2 log files or processes them concurrently (performs a merge action) depending on the following criteria, (see Table 11 on page 163):

- whether you are running in a data sharing environment
- the settings of the Log Master LOGSORT and LOGTAPES values
- whether you specify a member name to group each member’s DB2 log files

**Table 11  Criteria for either sort or merge of DB2 log files**

<table>
<thead>
<tr>
<th>Group files with member name?</th>
<th>Data sharing environment</th>
<th>Non–data sharing environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Determine whether to sort based on LOGSORT and LOGTAPES values (page 138)</td>
<td>Merge</td>
</tr>
<tr>
<td>No</td>
<td>Merge (unless LOGSORT is NEVER; page 138)</td>
<td>Merge</td>
</tr>
</tbody>
</table>

When you define a specific DB2 log file or set of DB2 log files as input, you might need to specify more resources than you expect (such as additional DB2 log files or image copies). This requirement occurs for the following reasons:

- Unless the tables selected by your filter are defined with the Data Capture Changes (DCC) attribute, Log Master frequently requires resources other than your selected DB2 log files for row completion processing. The resources can include
  - additional DB2 log files from other members in a data sharing group (for example, if activity on another member affects the table rows selected by your filter)
  - additional DB2 log files that cover periods of time outside of your range definition (for example, log files covering the period between an image copy and the start point of your range)

- If the DB2 objects selected by your filter reside in compressed table spaces, Log Master can require image copies to obtain a valid compression dictionary. You can define the location of the image copies by using the IMAGECOPY statement. Log Master can also store copies of compression dictionaries in the Old Objects Table of the Repository. For more information, see “INCLUDE DICTIONARY” on page 168.

**dataSetName**

Specifies a data set containing DB2 log records. The name that you enter must follow conventional guidelines for data set names. To specify DB2 log files from a non–data sharing system (or from only one member of a data sharing group), do not enter a `memberName` value.
If you list multiple DB2 log files, specify them in the correct order. The correct order depends on the setting of the ULOGORD installation option. Log Master always expects to read log files from the lowest RBA/LRSN value to the highest. If you specify a list of log files that is not in RBA/LRSN order after Log Master applies the ULOGORD setting, you can encounter errors. For more information about the correct order with the ULOGORD installation option, see “ULOGORD=ASCENDING” on page 62.

**DEVT deviceType**

Specifies the type of device that contains the DB2 log data set. Enter a site-specific name for a type of storage device (for example, 3390 for a DASD device or TAPE for a magnetic tape device).

**VOLUME (volser)**

Specifies the VOLSER where the DB2 log data set is located. Separate multiple VOLSERs with commas.

**memberName**

Specifies the name of a member in a data sharing group. Use this keyword to specify DB2 log files that come from more than one member in a data sharing environment. Group the log files from each member together, preceding the group with the member name. Within each member, specify the log files in the correct order. For more information, see “dataSetName” on page 163. For example, if two members of a data sharing group are DHF1 and DHF2, a correct INPUT statement would be:

```plaintext
INPUT DB2LOG
   ( 
      DHF1( DSNDHF.DHF1.ARCLOG1.A0005229 , DSNDHF.DHF1.ARCLOG1.A0005230 )
      DHF2( DSNDHF.DHF2.ARCLOG1.A0005849 , DSNDHF.DHF2.ARCLOG1.A0005850 )
   )
```

**LOGSCAN statement**

Figure 17 on page 165 shows the basic, high-level syntax of the LOGSCAN statement. The LOGSCAN statement determines how Log Master scans the DB2 log. Use this statement to define:

- the portion of the DB2 log that Log Master scans
- the criteria Log Master uses to select log records
- the types of output Log Master generates (reports or output files)
- whether Log Master creates a log mark
- other important characteristics

Figure 17  LOGSCAN statement syntax diagram (part 1 of 2)
To define a valid log scan, you must specify at least one form of output or one form of action on the Repository. You can specify multiple LOGSCAN statements, but Log Master scans the DB2 log only once for each work ID, no matter how many log scans, reports, or other forms of output you define.

**REPORT**

Directs Log Master to create reports using the information obtained in the log scan. For more information about the available reports, see “LOGSCAN report definition” on page 181.

**LLOG**

Directs Log Master to create a logical log file by using the information obtained in the log scan. A logical log is a readable version of the DB2 log that contains before and after images of database changes. For more information about generating a logical log, see “LOGSCAN logical log output definition” on page 200.

**SQL**

Directs Log Master to create an SQL output file by using the information obtained in the log scan. An SQL output file contains ANSI-standard SQL statements that either duplicate or reverse changes recorded in the DB2 log. For more information about generating an SQL output file, see “LOGSCAN SQL file definition” on page 208.
LOAD

Directs Log Master to create a load file that contains the log records specified in the log scan, formatted for a DB2 Load utility. Although Log Master uses the DB2 Load utility format to create the load file, the content is not the same as a load file created by a DB2 Unload utility. (The Log Master load file reflects activity over a period of time; a DB2 Unload utility creates a load file that reflects a given point in time.)

A load file is usually accompanied by a load control file that contains parameters for a DB2 Load utility. For more information about generating a load file, see “LOGSCAN load file definition” on page 231.

DDL

Directs Log Master to create a data definition language (DDL) output file based on the information obtained in the log scan. A DDL output file contains ANSI-standard DDL statements that duplicate or reverse structural database changes recorded in the DB2 log. You can generate only MIGRATE or UNDO DDL; Log Master does not generate REDO DDL. For more information about generating a DDL output file, see “LOGSCAN DDL file definition” on page 255.

NOTE

When you generate DDL output, Log Master can generate a Catalog Activity report or update the Repository, but it cannot generate other forms of output in the same log scan (such as other reports or SQL). To generate other forms of output in the same job, include an additional log scan step.

REPOS UPDATE

Directs Log Master to update or insert rows in the Old Objects Table (ALPOLDO) of the Repository. Each set of rows in the Old Objects Table defines a DB2 object at a given point in time. When Log Master is running in overtime mode and it encounters log records related to an object that is not currently defined in the DB2 catalog (for example, after the object has been dropped), it can use the old objects information in the Repository to process those log records. Log Master can use compression dictionaries stored in the Repository when it runs in current or overtime mode unless the value of the REPOS keyword on the OPTION statement is NO.
You can update the old objects information in the Repository from the following sources:

- the DB2 log (used by default unless you specify the NOSCAN keyword)
- the DB2 catalog (specify the PRIME FROM DB2 CATALOG keyword)
- an old objects data set

You can create an old objects data set to hold structure definitions of DB2 objects that are not currently defined in the DB2 catalog (old objects). To define the data set, use special syntax in the old objects data set and specify the name on the OPTION statement. For more information, see “Old objects data set syntax” on page 326 and “OLD OBJECTS dataSetName” on page 122.

Values for REPOS UPDATE are as follows. If you specify REPOS UPDATE without specifying one of these values, Log Master obtains old objects information from the DB2 log and does not include compression dictionaries.

**NOSCAN**

Directs Log Master not to use the DB2 log as a source for the old objects information in the Repository.

**PRIME FROM DB2CATALOG**

Directs Log Master to update the old objects information in the Repository with object descriptions from the DB2 catalog. Log Master searches for old objects information in both the DB2 catalog and the DB2 log unless you specify the NOSCAN keyword.

**INCLUDE DICTIONARY**

Directs Log Master to store both old objects information and copies of compression dictionaries in the Old Objects Table of the Repository. When a valid compression dictionary is not available, Log Master can use a copy stored in the Repository to interpret log records of compressed table spaces. Log Master can obtain dictionaries from the following sources, listed in order of preference:

- current table space
- DB2 log
- Old Objects Table of the Repository
- image copies of the table space

For information on recommended processes for storing compression dictionaries, see page 170.

For DB2 Load or Reorg actions that specify the DB2 KEEPDICTIONARY keyword, Log Master uses a record in the Old Objects Table of the Repository as a placeholder to indicate that the corresponding compression dictionary was not rebuilt. The placeholder sets the values of REC_CNT and REC_SEQ to zero for the
DB2 Load or Reorg action. Using a placeholder in this way ensures that for every rebuilt copy of a compression dictionary, Log Master stores only one dictionary object in the Repository. In addition, when using the dictionaries for decompression, Log Master loads only one copy in memory.

Table 12 shows the action that Log Master takes depending on the value of REPOS UPDATE and whether you specify an old objects data set.

**Table 12  Product action determined by REPOS UPDATE value**

<table>
<thead>
<tr>
<th>Syntax specification</th>
<th>Old objects data set</th>
<th>Product actions</th>
</tr>
</thead>
</table>
| REPOS UPDATE (no parameters)          | Not specified        | ■ uses the WHERE clause to define the objects that are added to the Old Objects Table  
                                         |                      | ■ obtains object information from only the DB2 log                           |
|                                       |                      | ■ might not add a row for a defined object (if the scan range does not include a CREATE or DROP for that object) |
| REPOS UPDATE (no parameters)          | Specified            | ■ adds all objects specified in the old objects data set to the Old Objects Table |
|                                       |                      | ■ adds an additional row for any objects found in the DB2 log                    |
|                                       |                      | that meet the criteria of the WHERE clause (use NOSCAN to prevent scan of DB2 log) |
| REPOS UPDATE PRIME FROM DB2CATALOG    | Not specified        | ■ uses the WHERE clause to define the objects that are added to the Old Objects Table |
|                                       |                      | ■ obtains object information from both the DB2 log and the DB2 catalog           |
|                                       |                      | ■ adds a row for each unique CREATE and DROP RBA set which Log Master encounters for a defined object |
| REPOS UPDATE PRIME FROM DB2CATALOG    | Specified            | ■ adds all objects specified in the old objects data set to the Old Objects Table |
|                                       |                      | ■ adds an additional row for any objects found in either the DB2 log or the DB2 catalog that meet the WHERE clause (use NOSCAN to prevent scan of DB2 log) |
| REPOS UPDATE PRIME FROM DB2CATALOG NOSCAN | Not specified   | ■ uses the WHERE clause to define the objects that are added to the Old Objects Table |
|                                       |                      | ■ obtains object information from only the DB2 catalog                           |
| REPOS UPDATE PRIME FROM DB2CATALOG NOSCAN | Specified   | ■ adds all objects specified in the old objects data set to the Old Objects Table |
|                                       |                      | ■ adds an additional row for any objects found in the DB2 catalog that meet the criteria of the WHERE clause |

Be aware of the following points regarding the REPOS UPDATE keyword:

- Log Master cannot update the Repository if the value of the REPOS keyword on the OPTION statement is NO.
To update the Old Objects Table in the Repository, you must specify the following additional keywords:

— specify the DB2CATALOG keyword of the LOGSCAN statement as YES
— specify the EXECMODE keyword of the OPTION statement as OVERTIME or ensure that the value of the EXECMODE installation option is OVERTIME

When you specify REPOS UPDATE, Log Master can generate a DDL output file or a Catalog Activity report in the same log scan, but it cannot generate other forms of output, such as other reports or SQL. To generate other forms of output in the same job, include an additional, separate log scan step. You cannot include two REPOS UPDATE log scans in the same work ID.

Log Master only creates objects in the ALPOLDO table for compression dictionaries that are rebuilt by DB2, and adds only one copy of an unchanged compression dictionary to the table.

Schedule jobs that update the Repository to run before

— regular production processing. For example, if you run a set of jobs every week, you should run a job to update the Repository before you run the weekly processing jobs.

— DB2 Load or Reorg actions that update compression dictionaries or that might assign table rows to different record ID (RID) values

If you update the Repository regularly, BMC Software recommends that you also run regular jobs to delete old or unusable data from the Old Objects Table. Alternately, you can delete (or display) information from this table by using an option on the Main Menu of the Log Master online interface.

Log Master cannot update the Repository when the input source for your log scan is a logical log file generated by Log Master or another program (INPUT LLOG).

When you store compression dictionaries in the Repository, you should

— run periodic jobs that update the Repository with current dictionaries
  Be sure to update the Repository before you use a DB2 Modify utility to delete any information about selected objects from the SYSIBM.SYSCOPY table.

— delete any rows from the Repository that represent old compression dictionaries if you decide to stop storing dictionaries in the Repository

If you do not delete the old dictionary rows, and you use a Modify utility to delete information from the SYSCOPY table, Log Master can use an incorrect dictionary from the Repository. For information about deleting rows from the Repository, see “Reduce Repository size when required” on page 385.
You can update the old objects information in the Repository by using the Generate REPOS UPDATE JCL option on the Main Menu of the Log Master online interface.

For more information about the Old Objects Table, the Repository, and the associated batch syntax, see the chapter on processing objects over time in the Log Master for DB2 User Guide.

REPOS DELETE

Directs Log Master to delete rows from either or both of the following tables within the Repository:

- Old Objects Table (ALPOLD0, used for overtime processing)
- Open Unit of Recovery table (ALPURID, used for ongoing processing)

Depending on your environment and the settings that you use to update the Repository, the size of these two tables can increase more rapidly than the other tables. Use REPOS DELETE to reduce the size of only these tables. To reduce the overall size of the Repository, use the sample REXX exec provided with the product. For more information, see “Maintaining the Repository” on page 385.

Be aware of the following points regarding the REPOS DELETE keyword:

- When you specify REPOS DELETE, the scan range definition of the LOSCAN statement is not required because Log Master does not read the DB2 log to delete Repository rows.

- You can specify both the OVERTIME and the ONGOING keyword in the same log scan, but you cannot specify either keyword more than one time.

- You can generate JCL to delete rows from either or both of these Repository tables by using the Generate REPOS DELETE JCL option on the Main Menu of the Log Master online interface.

- You can also delete (or display) information from the same two Repository tables directly through the Log Master online interface, without using JCL. Use the Delete/Display Repository Tables option on the Main Menu.

Select from the following values:

OVERTIME

Directs Log Master to delete rows from the Old Objects Table of the Repository (ALPOLD0). Log Master uses this table during overtime processing. Rows in this table represent either compression dictionaries or instances of objects that might have been dropped or dropped and re-created. This is the default value.
**ONGOING**

Directs Log Master to delete rows from the Open Unit of Recovery table of the Repository (ALPURID). Log Master uses this table for ongoing processing. Rows in this table contain information about transactions (units of recovery) that were open at the end of a previous ongoing log scan and that Log Master stored for later processing.

The WORKID keyword directs Log Master to delete only the ALPURID rows that relate to a work ID (or set of work IDs) that you specify. To specify a set of work IDs, use wildcard characters to define a string pattern for either the userID, the workID, or both portions of the work ID name. Log Master deletes rows in the ALPURID table that contain work IDs that match your string pattern. Use the same wildcard characters that you use when selecting log records. For more information, see Table 23 on page 282.

To specify multiple sets of work IDs, include an additional log scan in your job or job step (with an additional REPOS DELETE keyword).

**ALL**

Directs Log Master to delete all rows from the Repository tables that you specify.

**AGE (numOfDays)**

Directs Log Master to delete rows from your specified Repository tables that are older than a specified number of days. Enter a number of days between 1 and 32,767.

**RBA | LRSN X 'byteString'**

Directs Log Master to delete rows from your specified Repository tables that were created before a specific RBA/LRSN. The byte string can be up to 12 characters long. Use the following format: RBA X'nnnnnnnnnnnn'.

**DATE (mm/dd/yy)**

Directs Log Master to delete rows from your specified Repository tables that were created before a specified date. Enter a date using one of the available formats, including:

— MM/DD/YYYY (USA)
— DD.MM.YYYY (EUR)
— YYYY-MM-DD (ISO, JIS)

**BEFORE OLDEST ARCHIVE**

Directs Log Master to delete rows from your specified Repository tables that were created before the starting timestamp value of the oldest archive log file defined in the bootstrap data set (BSDS) of the current DB2 subsystem.
If you specify REPOS DELETE without specifying either the OVERTIME or ONGOING keywords, the default value is OVERTIME. You must specify a value for the row deletion criteria (ALL, AGE, DATE, or BEFORE OLDEST ARCHIVE).

**MARKSCAN**

Associates a symbolic name (called a log mark name) with the end point of your Range definition (the TO value). Use log marks to refer to a point in the log by using a name instead of an RBA/LRSN or a date and time value. Log Master keeps information about log marks in the Repository (in a table named ALPMARK). By using the MARKSCAN keyword, you can associate a log mark name with a point in the DB2 log other than the current RBA/LRSN value when your job starts executing.

Values for MARKSCAN are as follows:

*logMarkName*

Designates a specific log mark. A log mark name created within a given job step cannot be referenced within that job step. For more information about log marks and log mark names, see “MARK logMarkName” on page 274.

**DESC “description”**

Use this keyword to provide a description of the log mark name. The text can be up to 65 characters long and must be enclosed in quotation marks ("."). Continue text by keying through column 72 and continuing in column 1 of the next line.

**QUIESCE**

Directs Log Master to use the DB2 Quiesce utility to generate a quiesce point for the table spaces included in the log scan range. When you specify QUIESCE, Log Master overrides the TO value of your Range definition and sets the log end point to the current time. The log mark RBA/LRSN matches the quiesce RBA/LRSN used in the DB2 command. Without the QUIESCE keyword, Log Master chooses a log point that corresponds closely to the TO value.

Log Master also inserts index quiesce records into SYSIBM.SYSCOPY for COPY YES indexes along with the table space. Log Master records the correct quiet point registration so all recoverable objects have the proper QUIESCE entry in SYSCOPY.

**TABLESPACESET**

Directs Log Master to establish a quiet point for the specified table spaces, all referentially-related table spaces, and any related large object (LOB) table spaces or base table spaces that are not part of the referential integrity set.
**LOGSCAN statement**

---

**NOTE**

When using MARKSCAN QUIESCE,

- Ensure that your WHERE clause or filter refers to at least one specific DB2 object (such as a table name or a column name). This action ensures that you define (either directly or indirectly) a set of table spaces that Log Master can pass to the DB2 Quiesce utility.

- If the end point of a time frame is set to a log mark previously created with MARKSCAN QUIESCE, the resulting log scan contains only completed transactions.

---

**Sort file size parameters**

Specifies the estimated size of the data that Log Master must sort during a given log scan. You can also specify the size of the data to be sorted during an entire job on the SORTOPTS statement. For more information, see “Sort file size parameters” on page 152).

These parameters specify either the estimated size of the data to be sorted or a technique that Log Master uses to calculate the estimate. For more information, see “Sort file size parameters” on page 154.

**DB2CATALOG**

Determines whether Log Master includes DB2 catalog information from log records in the generated output.

**NO**

Directs Log Master to exclude DB2 catalog information as it generates output.

**YES**

Directs Log Master to include DB2 catalog information as it generates output. You must set DB2CATALOG to YES to update the Repository (REPOS UPDATE), generate a Catalog Activity report, or to generate data definition language (DDL) statements.

The DB2CAT installation option affects the action of the DB2CATALOG keyword. The person installing Log Master can set DB2CAT to NEVER, which prevents Log Master from including any DB2 catalog information in the generated output.

If your job operates on objects stored in the DB2 catalog, Log Master must perform completion processing on DB2 catalog log records. Because of the large number of log records related to the catalog, this processing can cause your job to run longer and require more resources than a job that does not read the DB2 catalog. If you frequently operate on objects stored in the catalog, you can improve performance by creating more frequent image copies of the catalog or by defining tables in the DB2 catalog with Data Capture Changes (DCC). Table 3 on page 65 shows which tables should be defined with DCC or Data Capture None (DCN).
Scan range definition

Specifies the part of the DB2 log that Log Master scans for information. The scan range definition can also specify a start point where Log Master begins generating REDO information (often called a recovery point). The scan range definition is not required when you specify the REPOS DELETE keyword.

For more information about specifying the range of the log scan, see “LOGSCAN scan range definition” on page 269.

FILTER

Specifies an existing filter that Log Master uses to select log records from within the log scan range. Filters are normally created using the online interface and are stored in the Repository. If you use the task-oriented dialogs of the online interface, the filter name defaults to be the same as the work ID name.

The filterName can be up to 18 characters long. The first character cannot be numeric, and any additional characters must be alphanumeric characters or national characters ($, #, or @).

Use the FILTER keyword to refer to a stored filter that already exists. To define new selection criteria within the current job, use the WHERE clause of the LOGSCAN statement.

userID.

You can qualify the filterName with a user ID. If you do not qualify the filter, the default value of userID is the TSO prefix or TSO user ID that submitted the job. For more information about creating filters, see the chapter on defining the log scan step in the Log Master for DB2 User Guide.

WHERE Search condition definition

Defines the search conditions that Log Master uses to select log records from within the log scan range. The conditions specified in a WHERE clause relate to database structures or characteristics, as opposed to a scan range that relates to a part of the log.

Use the WHERE clause to define new selection criteria within the current job. To use a stored filter that already exists, use the FILTER keyword of the LOGSCAN statement. For more information about specifying a search condition, see “LOGSCAN search condition definition” on page 277.
**ONGOING HANDLE handleID**

Defines a unique handle ID that Log Master uses to identify this log scan as an *ongoing* process (a log scan that is run repeatedly, with each start point dependent on the end of the previous log scan). For more information, see “Ongoing log scans” on page 179.

Log Master associates the different runs of an ongoing log scan by using the handle ID (along with the combination of the work ID and the user ID specified in a job’s WORKID statement). Use the online interface to create a work ID that contains an ongoing log scan. When you do, the online interface ensures that the Repository contains all necessary information and generates a unique combination of handle ID, work ID and user ID.

To rerun a job that contains an ongoing log scan, use either the USE or RERUN keywords. Both keywords can use the run sequence number of a previous run of an ongoing job. Log Master displays the run sequence number for a job in message BMC097494.

**USE RUNSEQ runseq | LASTRUN**

USE scans the log from the *start point* of a previous run that you specify with a run sequence number and continues to the end point that you specify in your Range definition (such as TO CURRENT). Use either a run sequence number or the LASTRUN keyword to specify a previous run. Log Master includes any transactions that are open at the beginning of the log scan.

When you specify USE LASTRUN, Log Master scans log from the start point of the most recent run that completed successfully and continues to the end point or limit that you specify in your Range definition. The following diagram shows that by entering USE with the *runseq* value from RUN 2, the scan range includes the open transactions represented by Open Transactions (1) and Open Transactions (2).
RERUN RUNSEQ runSeq | LASTRUN

RERUN scans the log from the start point of a previous run that you specify with a run sequence number and continues to the end point of that run. Specify the previous run by using either a run sequence number or the LASTRUN keyword. Log Master includes any transactions open at the start of the range, but excludes transactions that are open at the end of the range.

The following diagram shows that by entering RERUN RUNSEQ with the runSeq value from RUN 2, the scan range includes the open transactions represented by Open Transactions (1), but not the open transactions represented by Open Transactions (2).

When you specify RERUN LASTRUN, Log Master scans the log from the start point of the most recent run that completed successfully and continues to the end point of that run.

Be aware of the following points regarding USE RUNSEQ or RERUN RUNSEQ:

- If you specify the RERUN keyword, do not specify the QUIESCE value of the MARKSCAN keyword. This value sets the end point to the current log location. With the RERUN keyword, your end point is normally not at the current time.

- If you specify the RERUN keyword, Log Master ignores the specified limit.

- If you specify the USE keyword to re-use the start point of a previous log scan, the end point defined by the OR LIMIT keyword must be after the end point of the previous log scan.

- To honor the USE and RERUN keywords, Log Master obtains the start point of your specified run from the Repository by reading the end point of the run before your specified run. Because of this behavior, use caution when you specify these keywords again for subsequent log scans. In rare situations, Log Master can use a range for the log scan that you do not expect, or encounter errors. Table 13 shows an example.
LOGSCAN statement

RESET

Directs Log Master to establish a new start point for an ongoing log scan. Normally, Log Master adjusts the start point of an ongoing run based on the end point of the previous run (for more information, see “Ongoing log scans” on page 179). Use this keyword to prevent Log Master from adjusting the start point (for example, if DB2 log files for the period between the previous run and the current time are corrupted or unavailable).

When you specify RESET:

- Log Master uses the start point that you specify in the Range Definition (the FROM keyword)
- Log Master ignores any information stored in the Repository about open transactions from previous runs of the specified ongoing log scan (information remains in the Repository unless you specify the PURGE keyword)
- Log Master does not reset the run sequence number (the run sequence number of the current log scan is one more than the number of the previous run)
- You cannot specify the USE or RERUN keywords for the same run

The PURGE keyword causes Log Master to delete information related to the current ongoing log scan from the Log Master Repository (based on the combination of the user ID, work ID, and ongoing handle ID). Log Master does not reset the run sequence number. You must specify PURGE if your new end point is before the end point of the previous ongoing run.

Table 13 Example of possible errors with USE RUNSEQ or RERUN RUNSEQ

<table>
<thead>
<tr>
<th>Run sequence number</th>
<th>Ongoing syntax</th>
<th>Starting RBA/LRSN</th>
<th>Adjusted scan range (RBA/LRSN)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>FROM 050 TO CURRENT</td>
<td>100</td>
<td>050 – 100</td>
<td>Normal</td>
</tr>
<tr>
<td>02</td>
<td>FROM 050 TO CURRENT</td>
<td>200</td>
<td>100 – 200</td>
<td>Normal</td>
</tr>
<tr>
<td>03</td>
<td>FROM 050 TO CURRENT</td>
<td>300</td>
<td>200 – 300</td>
<td>Normal</td>
</tr>
<tr>
<td>04</td>
<td>FROM 050 TO CURRENT USE RUNSEQ 02</td>
<td>400</td>
<td>100 – 400</td>
<td>Normal (re-scan of previous runs)</td>
</tr>
<tr>
<td>05</td>
<td>FROM 050 TO CURRENT RERUN RUNSEQ 03</td>
<td>500</td>
<td>200 – 300</td>
<td>Normal (re-scan of run 03)</td>
</tr>
<tr>
<td>06</td>
<td>FROM 050 TO CURRENT RERUN RUNSEQ 05</td>
<td>600</td>
<td>400 – 300</td>
<td>Error (invalid scan range)</td>
</tr>
</tbody>
</table>
Ongoing log scans

Log Master provides the ability to define an ongoing log scan (one that is run repeatedly, with each start point dependent on the end of the previous log scan). With this feature, you can scan the logs multiple times, changing the range of the log scan for each run, but not changing the SYSIN syntax of your job step. In many environments, ongoing log scans extract data for migration to other databases or platforms.

For an ongoing log scan, Log Master keeps track of each run. Log Master also keeps track of any transactions (units of recovery) that start within the current log scan, but are still open at the end point. At the start of each run, Log Master automatically changes the start point so that it begins at the lowest RBA/LRSN address in the set of recorded open transactions from the previous run of the log scan. Log Master issues message BMC097778 to display any transactions (URIDs) that were open at the end of the previous run.

For ongoing log scans, Log Master ensures that

- any transactions that are open at the end of the current log scan will be included in a subsequent run of the log scan
- any transactions that were completed within a previous log scan are not processed twice, even though Log Master might scan part of the same log range again

When you work with ongoing log scans, remember the following points:

- Use the online interface to create a work ID that contains an ongoing log scan. The online interface ensures that an ongoing job’s combination of handle ID, work ID name, and user ID is unique. If you define your own handle ID outside of the online interface, ensure that you define a unique combination of these three items. (Log Master associates the different runs of an ongoing log scan by using the handle ID, along with the combination of the work ID name and the user ID specified in a job’s WORKID statement).

- When possible, schedule ongoing log scans to run before running any utilities that can change the location of rows within a table space. (For example, a DB2 Reorg utility can change the location of a row, so that the row associated with a record ID (RID) value is different than the row associated with that RID value in previous log records). Events that change the location of rows limit the Log Master options for row completion processing. When the log scan does not include such an event, Log Master is more likely to select an option for row completion processing that results in better product performance.

- When possible, schedule ongoing log scans to run before executing any data definition language (DDL) statements that change the structure of a table (for example, an ALTER COLUMN SET DATA TYPE statement) and any subsequent database reorganizations. This action enables Log Master to avoid scanning log
records for multiple versions of a DB2 table. When a log scan includes multiple versions of a table, Log Master performs additional processing to keep track of the versions and convert log records from previous versions to the current version of the table. Avoiding this additional processing can improve product performance.

- You can reset the start point of an ongoing log scan (for more information, see “RESET” on page 178).

- Log Master does not support ongoing log scans with an input source of logical log files.

- Log Master performs row completion processing differently for a work ID that contains an ongoing log scan. When an ongoing log scan requires access to the current table space, but updates to the table space have not yet been written to DASD storage, Log Master can defer completing some log records during the current log scan so that they can be completed in a subsequent log scan.

- Log Master responds to return codes greater than 8 (error) differently for a work ID that contains an ongoing log scan. For more information, see “Log Master termination processing” on page 110.

- Consider using ongoing log scans to periodically update the Log Master Repository with compression dictionaries (to avoid mounting image copies), or with structure definitions for old objects (to enable processing in overtime mode).
LOGSCAN report definition

Figure 18 shows the Report definition syntax of the LOGSCAN statement. Use this syntax to specify the reports that Log Master generates from the log records selected by current log scan. You can specify multiple Report definitions in one LOGSCAN statement, but Log Master scans the DB2 log only once for the entire work ID.

Figure 18  Report definition syntax diagram (part 1 of 2)
Use the Report definition to specify a report type and an output location. You can include or exclude specific column data.

Log Master reports (like all other forms of product output) include log data from transactions that are completed within the scan range that you specify. In this context, completed means either committed or aborted. To obtain information about transactions that are not complete at the end of a scan range, use the Open Transaction report or the Object Activity Summary report. For more information, see “OPEN TRANS” on page 188 or “ALL ACTIVITY” on page 190.

The following pages contain information about report types, along with references to output and column include/exclude syntax. For more information on Unicode characters in reports, see “Considerations for output files and Unicode” on page 229.
TEMPLATE

Directs Log Master to format the generated report by using a report template that is stored in an external data set or in the Log Master Repository. To create a report template, use the Log Master online interface to define the template. Use options in the interface to export the template into a data set or store it in the Repository. Log Master stores the report template in the form of XML elements and attribute values.

The report type that you specify in your job or job step must match the report type defined in the template. A template reflects the report options (for example, order by fields) that were selected when the template was created. To change the order by fields or other options, modify the template. You cannot specify a template and an Order By definition for the same report.

Log Master also provides sample report templates. For more information, see “Sample files and other information” on page 87. You can import the templates in the Log Master online interface and modify them if needed, or you can use them via the REPORT TEMPLATE DATASET syntax.

DATASET

Defines a data set that contains a template that Log Master uses to format the generated report. The data set must contain a valid Log Master report template.

ID

Defines the identity of a template in the Log Master Repository. Log Master uses the template to format your generated report. To use a template from the Repository, the template must have been created in or imported into the Repository on the DB2 subsystem (SSID) where your job or job step runs. To use the same template on multiple subsystems, export the template to a data set.

TYPE

Specifies the type of report that Log Master generates. The following paragraphs list and describe the reports generated by each TYPE keyword.

For more information about different types of reports, such as sample reports or ordering defaults and restrictions, see the chapter on Log Master for DB2 reports in the Log Master for DB2 User Guide.

BACKOUT INTEGRITY

Provides information about changes in data that would complicate an UNDO or REDO action. For each change that is part of an UNDO or REDO action, the report shows information about the change and about subsequent changes that would be affected if the original change is executed. The report also supplies information about the set of affected objects and the volume of information affected by the changes. The report presents only data associated with committed transactions.
To generate a Backout Integrity report, Log Master scans the DB2 log to the current time, even if you do not specify CURRENT as your end point. Log Master takes this action because the log records that you select might be affected by subsequent transactions that occur after your time frame. Log Master defines the current time as the last relative byte address (RBA) or log record sequence number (LRSN) that DB2 has written to the log when the Log Master job begins executing.

Log Master can produce the Backout Integrity report in two formats: Detail (including field data and all URID information) or Summary (a more concise report, omitting field data and some URID information). Use the SUMMARY keyword to generate the concise version of the Backout Integrity report.

Log Master cannot produce either version of a Backout Integrity report when the input source of your log scan is individual DB2 log files (INPUT DB2LOG).

**QUIET POINT**

Provides information about ranges within the DB2 log when no activity is present for a given set of DB2 objects. The report lists only the quiet ranges common to all table spaces defined (either directly or indirectly) by your WHERE clause or filter. The optional MAXURIDS keyword enables you to list ranges when the amount of activity is below a level that you specify (“almost quiet” ranges).

To generate this report, ensure that your WHERE clause or filter refers to at least one specific DB2 object (such as a table name or a column name). This action ensures that you define (either directly or indirectly) a set of table spaces that Log Master can search for quiet ranges.

Log Master cannot produce a Quiet Point report when the input source for your work ID is individual DB2 log files (INPUT DB2LOG) or a logical log file generated by Log Master or another program (INPUT LLOG).

**DURATION  \[HH.MM.SS.mmmmmm\]**

By default, the Quiet Point report includes all quiet ranges for the specified DB2 objects (no minimum). Use this keyword to exclude shorter quiet ranges from the report. Log Master suppresses any quiet ranges that are shorter than the duration that you specify.

Enter a minimum duration, using the time format shown in the syntax diagram. Enter any value from one microsecond to 24 hours. You must enter numeric values for hours, minutes and seconds, but microsecond values are optional. For example, to specify five seconds, enter 00.00.05 in the syntax.
MAXURIDS  \textit{uridCount}

By default, the Quiet Point report includes only quiet ranges that are completely quiet (absolutely no activity for the table spaces directly or indirectly defined by your filter). By using this keyword, you can include “almost quiet” ranges in the report. These ranges are periods within the log when the number of open transactions (units of recovery) that affect your table spaces is less than or equal to the number that you specify. Use this keyword to report on periods of high activity for your specified objects, when completely quiet ranges might not exist.

The report includes both the almost quiet ranges and the completely quiet ranges. For each quiet range, the report lists the number of open transactions that are present during the range.

Enter a number of open transactions between zero and thirty-two. Zero indicates that the report includes only completely quiet ranges. The default value is zero.

QUIESCE

You can direct Log Master to insert a quiesce entry into the DB2 catalog table SYSIBM.SYSCOPY for each table space defined (directly or indirectly) by the WHERE clause or filter. Specify YES to insert a quiesce record in SYSIBM.SYSCOPY using the RBA/LRSN of the latest quiet range included in the report. Specify NO to avoid inserting a quiesce record. The default value is NO.

When you specify YES, Log Master also inserts index quiesce records into SYSIBM.SYSCOPY for any COPY YES indexes associated with the table spaces. Log Master records the correct quiet point registration so all recoverable objects have the proper QUIESCE entry in SYSCOPY.

Log Master does \textit{not} insert a quiesce record if

- Log Master finds no quiet ranges that meet your criteria
- the report includes only almost quiet ranges

If you use a report template to customize the Quiet Point report, you can direct Log Master to sort the quiet ranges based on duration. If you sort in ascending order, Log Master uses the longest quiet range in your report as the point to insert a quiesce record in SYSIBM.SYSCOPY.

AUDIT

Provides a record of changes to data, identifying who changed the data, when it was changed, and how much data was changed. This report also gives related index and record ID (RID) information.

The Audit report is similar to the report produced by the DETAIL keyword, except that the Audit report presents only \textit{changed} data from update log records, and it includes primary key information.
You can also use an Audit report to show information about changes to the DB2 catalog, including changes to DB2 security. The Catalog Activity report also provides this information.

The REDO keyword is valid for only Audit, Detail, or Summary reports. Use the REDO keyword only when you want a report to show you which log records would be selected for a REDO operation based on your current WHERE clause or filter. The keyword determines how Log Master uses your filter to select log records for the output report. For more information, see “REDO” on page 190.

**CATALOG ACTIVITY**

 Provides information about changes that affect the DB2 catalog, including

- the type of DB2 object that was changed
- the type of data definition language (DDL) statement that changed the object
- URID-related information about the process, program, or user that made the change (for example, authorization ID or correlation ID)

A Catalog Activity report can show all changes that affect a given object type (for example, indexes), or all changes that result from a given DDL statement type (for example, GRANT and REVOKE statements to show security changes). This report contains the same basic information that would be included if you generated a MIGRATE DDL output file with the VERBOSE keyword.

Be aware of the following points as you define a Catalog Activity report:

- To generate a Catalog Activity report, you must set the value of the DB2CATALOG keyword to YES.

  The Catalog Activity report uses a wide format for data. Specify a logical record length (LRECL) value of at least 132 bytes to support the data format.

- “Catalog activity definition” on page 292 and “Catalog object definition” on page 295 provide filter options that are helpful with the Catalog Activity report.

- “ORDER BY definition” on page 194 provides the ACTIVITY TYPE and OBJECT TYPE keywords to conveniently sort a Catalog Activity report.

- To obtain the information that you need, you might need to include some additional DB2 objects in your filter. For more information about filter optimization and usage, see the chapter on defining the log scan step in the Log Master for DB2 User Guide.

- When you generate a Catalog Activity report, Log Master can generate an output DDL file or update the Repository, but it cannot generate other forms of output in the same log scan (such as other reports or SQL). To generate other output in the same job, include an additional, separate log scan step.
**COMMAND**

Provides information about DB2 commands that were issued within a specified period on a DB2 subsystem.

Be aware of the following points as you define a Commands report:

- This report is available when you are running Log Master on DB2 Version 9 or later.

- “ORDER BY definition” on page 194 provides the COMMAND keyword to conveniently sort a Commands report.

- When you generate a Commands report, Log Master cannot generate other forms of output in the same log scan (such as other reports or SQL). To generate other output in the same job, include an additional, separate log scan step.

**COMMIT**

Provides information about commit frequency for application tuning and system performance purposes. Depending on how you order the report, counts are maintained for each unit of recovery to indicate the activity volume between commit points.

Use this report to determine if batch jobs are not committing changes according to standards or performance recommendations. You can also use it to determine if online transactions are performing too many operations in a given unit of work.

**DATA CAPTURE ANALYSIS**

Provides information about log records with either the data capture changes (DCC) or data capture none (DCN) attributes. This report provides information to help determine the impact of setting the data capture changes attribute on specific tables. Log Master estimates the number of bytes that would be logged if you set data capture changes on or off.

**DETAIL**

Provides information about log records. Use this report to display log data to help diagnose application problems, or to show log data as an audit trail for applications.

This report is similar to the report produced by the AUDIT keyword, except that the Detail report presents all data from update log records, and it does not include primary key information.
You can also use a Detail report to show information about changes to the DB2 catalog, including changes to DB2 security. (The Catalog Activity report offers a more concise version of the same basic information. For more information, see “CATALOG ACTIVITY” on page 186).

The REDO keyword is valid for only Audit, Detail, or Summary reports. Use the REDO keyword only when you want a report to show you which log records would be selected for a REDO operation based on your current WHERE clause or filter. The keyword determines how Log Master uses your filter to select log records for the output report. For more information, see “REDO” on page 190.

**IMAGE COPY**

Provides information about the volume of table space activity between image copies. For the table spaces selected (directly or indirectly) by your filter, the report shows data on all applicable image copies logged in the SYSIBM.SYSCOPY table of the DB2 catalog during your time frame. For each image copy listed, the report shows the activity (the number of insert, update, and delete actions) since the preceding image copy.

Use this report to determine if a table space has too many updates between copies (which could lead to excessive recovery times) or if a table space is being copied more often than necessary.

**OPEN TRANS**

Provides information about transactions (units of recovery) that are not committed or aborted at the end point of your log scan, including any transactions that begin before the start point of your log scan. Use this report to identify any work that might not be represented on other product reports. Be aware of the following points regarding this report:

- Log Master reads DB2 checkpoint records before the start point of your log scan to obtain information about the transactions that were open when the checkpoint record was created.

- The report can include open transactions that do not affect the DB2 objects selected (directly or indirectly) by your filter. Log Master includes these transactions because an open transaction *might* include activity affecting your selected objects past the end point of your log scan.
The transactions included in the report can vary depending on whether your filter selects log records based on unit of recovery criteria (for example, by plan name or authorization ID).

— If your filter selects based on only unit of recovery criteria, the report includes only the open transactions selected by your filter.

— If your filter selects based on either 1) only object-specific criteria (such as table name or column name), or 2) both unit of recovery and object-specific criteria, the report includes all open transactions on the current DB2 subsystem.

The activity counts in the report are based on the log records selected by your filter and the range of your log scan. Treat the counts as estimates of the activity associated with open transactions; do not use them as definitive data.

**ROLLBACK**

Provides information about all rollback actions performed over a specified period of time, including the frequency and number of associated transactions. Depending on how you order the report, it can present activity counts for each rolled-back transaction (in this context, transactions are considered to be the same as URIDs). The Rollback report contains only aborted transactions and cannot include committed transactions.

Use this report to determine the amount of activity associated with rollback actions or to pinpoint jobs or transactions that are frequently rolled back.

**LOG BYTES**

Provides information about the distribution of log records in the DB2 log. It focuses on the type, size and distribution of log records, and not on the actual data contained in the log records. The Log Bytes report displays the following information about log records:

- counts of many log record types
- total number of bytes used by the log records
- percentage of space the log records use

Use the Log Bytes report for tuning purposes, to help you reduce unnecessary overhead in the log.

When you generate a Log Bytes report, Log Master cannot generate other forms of output in the same log scan (such as other reports or SQL). To generate other output in the same job, include an additional, separate log scan step.
**SUMMARY**

Provides summaries of transaction count information (the number of insert, update, and delete actions). Use this report to determine how many transactions are processed by various sort keys, or to show how much SQL would be generated by a REDO or UNDO operation on a given table space.

**REDO**

This keyword is valid for only Audit, Detail, or Summary reports. Use this keyword only when you want a report to show you which log records would be selected for a REDO operation based on your current WHERE clause and Range definition. The keyword determines how Log Master selects the log records for the output report.

When REDO is not specified, Log Master selects log records for your report that are selected by your filter and Range definition (as it does when it generates MIGRATE or UNDO SQL statements.) This value is the default and is recommended for most uses.

When REDO is specified, Log Master selects log records for your report that are within the table spaces defined (directly or indirectly) by the filter, and within the period from the REDO recovery point to the current time. Then Log Master excludes the log records that are selected by your filter and Range definition. Log Master selects log records in this manner because it assumes that any REDO operation takes place after a recovery, and that you want to reapply all of the log records except the problem log records defined by your filter.

For more information about REDO operations, see “REDO” on page 215.

**ALL ACTIVITY**

Use the ALL ACTIVITY keyword to generate an Object Activity Summary report. This report presents database activity (insert, update, and delete actions) from all transactions, including transactions that relate to DB2 objects that are no longer in the DB2 catalog, and transactions that are not complete within your scan range (not yet committed or aborted). With few exceptions, this version of the report presents information from all selected data management (DM) records in the DB2 log.

You cannot generate an Object Activity Summary report if Log Master is reading log records from a logical log file previously created by Log Master or an application program, or if your filter selects log records based on a column’s value (for example, WHERE TABLE01.QUANTITY > 500). For more information, see the Object Activity Summary report in the Log Master for DB2 User Guide.
FORMAT

Use the FORMAT keyword to generate Summary report output that includes only data (no report headings, column headings, or blank lines). Log Master provides two different formats, both of which use character representation for all data to make it easy to move data to other platforms.

- Specify CSV to generate data in the Comma Separated Value (CSV) format. Individual items of data are separated by commas.

- Specify SDF to generate data in the Standard Definition Format (SDF) format. Individual columns of data are filled with spaces so that each column has the same position and length on all lines of the output.

For either format, ensure that the logical record length (LRECL) of your report output file is long enough to contain the longest line in the report. For more information about LRECL values, see Table 21 on page 263. For examples of report output in the CSV or SDF formats, see the Log Master for DB2 User Guide.

INCLUDE TOTAL

Use the INCLUDE TOTAL keyword to include subtotal and report total lines in the output. This keyword is valid only when the format of report is CSV or SDF. By default, subtotal and report total lines are not included.

ORDER BY definition

By default, Log Master sorts each report according to a set of sort criteria. To change the sort order of a report, use the ORDER BY definition. For more information about changing the order, see “ORDER BY definition” on page 194.

DATASET Output definition

Directs Log Master to place the generated report in a data set. When you specify the DATASET keyword, you also must specify the data set name and disposition. For more information about specifying a location for the data set, see “LOGSCAN output definition” on page 260.

SYSOUT SYSOUT definition

Directs Log Master to send the generated report to the Job Entry System (JES) spool queue (known as SYSOUT). For more information about specifying JES SYSOUT parameters, see “SYSOUT definition” on page 197.
LOGSCAN report definition

**DDNAME**

Directs Log Master to place the generated report in a DD name. When you specify the DDNAME keyword, you also must specify `ddname`.

**ddname**

Specifies the eight-byte-maximum DD name in which Log Master places the generated report. You can specify a partitioned data set, a sequential data set, or a generation data group (GDG). If you do not specify `ddname`, Log Master generates SYSOUT output.

For information about LRECL and RECFM requirements for report outputs, see Table 21 on page 263.

**INCLUDE ROLLBACK**

Determines whether the output report includes information from compensated log records selected by your filter. Use a value of NO for normal processing. In this context, compensated log records are the log records of database changes that DB2 subsequently "compensates for" or "reverses." DB2 reverses changes in response to several situations, including ROLLBACK statements, ROLLBACK TO SAVEPOINT statements, or some negative SQL codes.

- noncompensated log records reflect original changes that DB2 does not reverse
- compensated log records reflect original changes (that are subsequently reversed)
- compensation log records reflect actions taken by DB2 to reverse the original changes

The Include Rollback field affects how Log Master processes compensated log records, not compensation log records.

Depending on the value of this keyword, Log Master includes or excludes compensated log records.

**NO**

Directs Log Master to exclude compensated log records. The output report reflects only noncompensated log records (log records of database changes that are not reversed). Use this value for normal processing.

**YES**

Directs Log Master to include compensated log records. The output report reflects both compensated log records and noncompensated log records.
ONLY

Directs Log Master to include only compensated log records. The output report reflects compensated log records, but does not reflect noncompensated log records.

WARNING

Exercise caution before specifying a value of YES or ONLY. In most circumstances, when you include compensated log records, your report includes data that has not been applied to the original DB2 database. If you take actions or apply changes to your database based on a Log Master report that reflects compensated log records, you can corrupt the data in your database.

Table 14 shows the default values of the INCLUDE ROLLBACK keyword and how it interacts with the different report types.

Table 14  INCLUDE ROLLBACK values by report type

<table>
<thead>
<tr>
<th>Report type</th>
<th>INCLUDE ROLLBACK valid?</th>
<th>INCLUDE ROLLBACK default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Backout Integrity</td>
<td>NO (includes only committed transactions)</td>
<td>NO</td>
</tr>
<tr>
<td>Commit</td>
<td>NO (includes only committed transactions)</td>
<td>NO</td>
</tr>
<tr>
<td>Data Capture Analysis</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Detail</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Image Copy</td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>Object Activity Summary</td>
<td>No (includes all transactions, regardless of presence of compensated log records)</td>
<td>YES</td>
</tr>
<tr>
<td>Open Transaction</td>
<td>No (includes only open transactions, regardless of presence of compensated log records)</td>
<td>not applicable</td>
</tr>
<tr>
<td>Quiet Point</td>
<td>No (does not include database transactions)</td>
<td>not applicable</td>
</tr>
<tr>
<td>Rollback</td>
<td>No (includes only aborted transactions)</td>
<td>ONLY</td>
</tr>
<tr>
<td>Summary</td>
<td>Yes</td>
<td>NO</td>
</tr>
</tbody>
</table>

Column include/exclude

The Column include/exclude definition enables you to control the information in your report based on specific columns in a table. For more information about including or excluding column data, see “LOGSCAN column include/exclude definition” on page 267.
ORDER BY definition

Figure 19 shows the ORDER BY definition syntax of the LOGSCAN statement. By default, Log Master sorts reports based on certain criteria (for example, entries in the summary report are often sorted in order based on table name). To change the default sort criteria, use the ORDER BY definition. You can choose a maximum of three sort criteria.

The ORDER BY definition is not valid for a Backout Integrity report or a standard default Quiet Point report. Different types of reports have different sets of ordering defaults. Several reports have restrictions on the columns or fields that you can use for ordering. For more information about a particular report’s defaults or restrictions, see the chapter on reports in the Log Master for DB2 User Guide.

You can also specify sort criteria by using a report template. If you use a report template to generate a report, you cannot specify an ORDER BY definition for the same report. Log Master provides additional sort capabilities in report templates that are not available for the standard default reports.

Figure 19  ORDER BY definition syntax diagram

orderingField

Specifies a sort criteria used to sort Log Master reports. The available sort criteria are as follows:

**ACTIVITY DESC**

Sorts reports based on activity, in descending order. Activity in this context represents the number of database changes (insert, update, delete, or exchange actions) that have occurred against a given DB2 object.

**ACTIVITY TYPE**

Sorts reports based on the type of DB2 catalog activity. This criteria is valid only for a Catalog Activity report. You can also think of this syntax as sorting the report based on a type of data definition language (DDL) statement. For example, the report can group together all log records that reflect DROP statements, or ALTER statements.
**AUTH ID**

Sorts reports based on authorization ID. An authorization ID is a character string used to connect to DB2; it frequently represents an individual, a work group, or a function.

**COMMAND**

Sorts reports based on a DB2 command name or DB2-supported synonym, such as START DATABASE (-STA DB). This criteria is valid only for a Commands report.

**CONNECTION ID**

Sorts reports based on connection ID. A connection ID is a string identifier associated with a specific address space connection.

**CONNECTION TYPE**

Sorts reports based on connection type. A connection type is a string constant representing the manner in which a unit of recovery thread is attached to DB2. For a list of possible connection type values, see “CONNECTION TYPE” on page 288.

**CORRELATION ID**

Sorts reports based on correlation ID. A correlation ID is a unique string identifier for a specific DB2 thread.

**CREATE RBA**

Sorts reports based on the RBA/LRSN value when a DB2 object was created. Use this value to sort reports that Log Master produces when it runs in overtime mode. For example, in overtime mode two different instances of a table can have the same name (and possibly the same DBID or OBID). By using this value, you can separate the data from each instance of the table.

You must specify TABLE NAME or OBID as a sort criteria before you specify CREATE RBA.

**DATABASE NAME**

Sorts reports based on database names.

**DBID**

Sorts reports based on database ID (DBID). A DBID is an internal identifier assigned to a database by DB2.
**DSNUM**

Sorts reports based on data set number. The data set number is the ordinal number of a given data set within a table space (usually a partitioned table space).

**MEMBER ID**

Sorts reports based on the member ID of a given member of a data sharing group. Use this keyword only with reports generated in a data sharing environment.

**MEMBER NAME**

Sorts reports based on the name of a given member of a data sharing group. Use this keyword only with reports generated in a data sharing environment.

**OBID**

Sorts reports based on object ID (OBID). An OBID is the internal identifier that DB2 assigns to an object.

**OBJECT TYPE**

Sorts reports based on the type of DB2 object. This criteria is valid only for a Catalog Activity report. For example, the report can group together all log records that affect tables or all log records that affect indexes. Besides the commonly used objects, such as tables and columns, the object type includes other objects defined in the DB2 catalog, such as plans, packages, collections, and buffer pools.

**PLAN NAME**

Sorts reports based on plan name. A plan name is a character string representing an application plan, the control structure used by DB2 to process SQL statements.

**PRIMARY KEY**

Sorts reports based on the primary key of a table. The primary key of a table is a column or set of columns within a table, defined so that there is one unique primary key value for each row in the table. If you specify PRIMARY KEY, but a table has no primary key defined, Log Master selects a key to use. For more information, see “QUALIFY” on page 209.

**PSID**

Sorts reports based on page set ID (PSID). A PSID is an internal page set identifier assigned to a given table space by DB2.
SSID

Sorts reports based on the subsystem identifier (SSID) of a given DB2 subsystem. Use this keyword only with reports generated in a data sharing environment.

TABLE NAME

Sorts reports based on the names of tables selected in your WHERE clause or filter.

TABLESPACE NAME

Sorts reports based on the names of table spaces selected in your WHERE clause or filter.

UNIT OF RECOVERY

Sorts reports based on unit of recovery identifier (URID). A URID is a hexadecimal value assigned by DB2 to a recoverable sequence of operations within DB2.

WITH ALL FIELDS

Use this keyword to include all of the URID-related fields in a generated Summary report. Log Master obtains these fields from the “Begin UR” log record of each unit of recovery and they include values such as authorization ID or correlation ID. By default, Log Master includes only a subset of the URID-related fields.

This keyword is valid only for Summary reports and only when the format of the report is CSV (comma separated value) or SDF (standard definition format).

SYSOUT definition

Figure 20 on page 198 shows the SYSOUT definition syntax of the LOGSCAN statement. This syntax determines how Log Master writes output to the JES spool queue. When you generate a job using the online interface, Log Master includes the default keywords CLASS(*) and NOHOLD.

For more information about these keywords and their allowable values, consult your operating system JCL documentation.
SYSOUT definition

Figure 20  SYSOUT definition syntax diagram

CLASS (class)

Specifies the output data class for the SYSOUT data set. The attributes of the class are defined during JES initialization.

DEST (destination.userID)

Specifies a routing destination for the SYSOUT data set (for example, a local storage device or a node). You can qualify destination with a TSO user ID.

HOLD | NOHOLD

Indicates whether Log Master creates the SYSOUT data set in HOLD status (in HOLD status, an operator must act to print and purge the data set). The default value is NOHOLD.

COPIES (nnnn)

Specifies how many copies of the SYSOUT data set are printed.
**CHARS** *(tableName)*

Specifies the name of the character-arrangement table used to print the SYSOUT data set. You can enter up to four character-arrangement tables. Separate multiple table names with commas.

**SEGMENT** *(pageCount)*

Specifies the size of a segment in the SYSOUT data set. Segmented SYSOUT data sets enable you to print each segment of the output while the job is still running.

**MODIFY** *(moduleName)*

Specifies the copy-modification module that JES uses to print the SYSOUT data set. A copy-modification module describes printing attributes (such as legends or column headings) for some printing devices. The table reference character *(trc)* specifies one of the character-arrangement tables entered using the CHARS keyword. Log Master uses the specified table.

**FCB** *(imageID)*

Specifies the forms control buffer (FCB) image that JES uses to print the SYSOUT data set. An FCB image describes printing attributes (such as lines per inch and form length) for some printing devices.

**FORMS** *(formsName)*

Specifies the print forms on which JES prints the SYSOUT data set.

**UCS** *(universalCharacterSetName)*

Specifies the universal character set (UCS) image that JES uses to print the SYSOUT data set. A UCS image defines the characters used and (in some cases) additional printing attributes.

**WRITER** *(externalWriterName)*

Specifies the name of an external writer program to process the SYSOUT data set (instead of JES). A vendor can supply the external writer, or your environment can create its own external writer program.

**LRECL** *(nnnn)*

Specifies the logical record length of records in the output report. Enter the *nnnn* value in bytes. For reports in CSV or SDF format, ensure that the LRECL of SYSOUT is long enough to contain the longest line in the report. For more information about LRECL values, see Table 21 on page 263.
LOGSCAN logical log output definition

Figure 21 shows the Logical log output definition syntax of the LOGSCAN statement. A logical log is a readable version of the DB2 log that contains before and after images of database changes. This syntax specifies the location and format of the output logical log control file and the logical log data file that Log Master creates using the log records selected in your log scan.

For more information about using logical log files as input, see “INPUT statement” on page 161. For more information about Unicode characters in logical log files, see “Considerations for output files and Unicode” on page 229.

Figure 21  Logical log definition syntax diagram

1 See page 202 for default values of INCLUDE
DATASET Output definition

Specifies the characteristics of the data set that contains the logical log data file. For more information, see “LOGSCAN output definition” on page 260.

CONTROL Output definition

Specifies the characteristics of the data set that contains the logical log control file. For more information, see “LOGSCAN output definition” on page 260.

XMLSTRING Output definition

Specifies the characteristics of the data set where Log Master stores the XMLSTRING control file. This file contains the string IDs and string data that DB2 uses to encode the data in XML columns. By default, Log Master does not generate this data set.

Specify this file only if you intend to use the output logical log file to migrate data to a different DB2 subsystem. When you apply the changes in the logical log to a different target subsystem, Log Master or the High-speed Apply Engine need the string IDs and data to serialize your XML data correctly. Log Master obtains this data from the DB2 catalog table SYSIBM.SYSSXMLSTRINGS.

For more information on specifying the data set characteristics, see “LOGSCAN output definition” on page 260.

RECFM

Documents the record format of the output logical log files. Log Master forces the record format of logical log control and data files to be variable length, and blocked (VB). If Log Master detects that any records in the logical log data file exceed the length restrictions of the VB format, it automatically divides the records into multiple segments and writes those segments to a VB file. For more information, see “Normal and segmented logical log record formats” on page 376.

VB

Indicates that records are variable length and blocked.

VBS

This keyword is tolerated for historical reasons, but ignored by the current version of Log Master. If you specify VBS, Log Master writes VB output logical log files.
LOGSCAN logical log output definition

DATEFMT

Specifies the date and time format used in the output logical log.

DB2I

Directs Log Master to use DB2’s internal date and time format. The DB2I format is YYYY-MM-DD-hh.mm.ss.nnnnnn. This is the default value.

When running on DB2 Version 10 and later, Log Master supports precision timestamps up to 12 digits, and inclusion of a time zone in the timestamp (YYYY-MM-DD-HH:MM:SS.nnnnnnnnnnnn±HH:MM).

SASTIME

Directs Log Master to use a date and time format compatible with software developed by the SAS Institute Inc. (a maker of decision support and statistical analysis software). The SASTIME format is YYYY/MM/DD/hh:mm:ss.nnnnnn.

EXPAND VAR

Determines how Log Master writes variable length fields in the output logical log.

YES

Directs Log Master to expand (fill) variable length fields with spaces up to the declared maximum length.

NO

Directs Log Master not to expand (fill) variable length fields. This is the default value.

Log Master does not expand data definition language (DDL) information in the logical log data file, even when this keyword is YES.

INCLUDE ROLLBACK

Determines whether Log Master includes information from compensated log records in the output logical log file. Use the default value of NO for normal processing. In this context, compensated log records are the log records of database changes that DB2 subsequently “compensates for” or “reverses” (for example, log records of changes that are later reversed by a ROLLBACK statement). Depending on the value of this keyword, Log Master includes or excludes compensated log records. For more information, see “INCLUDE ROLLBACK” on page 192.
NO

Directs Log Master to exclude compensated log records. The output logical log file does not reflect any compensated log records. Use this default value for normal processing.

YES

Directs Log Master to include compensated log records. The output logical log file reflects both compensated log records and noncompensated log records.

ONLY

Directs Log Master to include only compensated log records. The output logical log file does not reflect any noncompensated log records.

--- WARNING ---

Exercise caution before specifying a value of YES or ONLY. In most circumstances, when you include compensated log records, your output logical log file includes data that has not been applied to the original DB2 database. For example, if you migrate data from one database to another and you use an output logical log file that reflects compensated log records, you can corrupt the data in your target database.

If you do not specify INCLUDE ROLLBACK, Log Master ignores compensated log records. If you specify INCLUDE ROLLBACK without specifying YES or NO, the default value is YES (Log Master includes both compensated and noncompensated log records).

INCLUDE SYNONYMS

Determines whether Log Master includes information about the synonyms and aliases of DB2 tables in the output logical log file. Use this keyword when later processing of the logical log file needs to use or select records based on synonym or alias names. By default, Log Master does not include this information.

To obtain synonym and alias names, Log Master reads the SYSIBM.SYSSYNONYMS table in the DB2 catalog. This processing can slow performance. If you include synonyms or aliases frequently, you can reduce the performance loss by creating an additional index on the SYSSYNONYMS table. The ALPIXSYN member in the HLQ.DBSAMP library contains a sample DDL statement that you can modify to create the additional index. The HLQ value represents a qualifier that your environment assigns during installation.
INCLUDE DDL

Determines whether Log Master includes data definition language (DDL) information in the output logical log file. When the value of this keyword is YES, Log Master writes objects representing DDL activity into the logical log data file.

The DDL objects that Log Master writes in an output logical log file are not the same as the DDL statements that it writes in an output DDL (or SQL) file. A DDL object can represent multiple DDL statements, and a DDL statement can represent multiple insert, update, or delete actions against several tables in the DB2 catalog.

A DDL object that Log Master writes can be interpreted as input by either Log Master itself (using the LLOG keyword of the INPUT statement) or by the High-speed Apply Engine that is distributed with Log Master. However, these DDL objects cannot be used to generate UNDO DDL, and they cannot be processed by the High-speed Apply Engine as UNDO. The DSNDB06 insert, update, and delete records must be included in the logical log file to generate UNDO DDL.

**YES**

Directs Log Master to include DDL information. The output logical log file includes both DML statements (for insert, update, and delete actions) and DDL objects (representing DDL activity on the subsystem).

**NO**

Directs Log Master to include only DML statements (for insert, update, delete, or exchange actions).

If you omit the INCLUDE DDL keyword, the default value is NO (Log Master does not include DDL statements). If you specify the INCLUDE DDL keyword, but omit the YES or NO keywords, the default value is YES (DDL objects are included). The DDL objects written in the output logical log file are not affected by the setting of the EXPANDVAR keyword.

**NOTE**

It is possible to track DDL activity by selecting insert, update, and delete actions against several DB2 catalog tables (including SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, and several others). You can use the INCLUDE DDL keyword to capture DDL activity, or you can select log records relating to the DB2 catalog tables, but Log Master does not perform both types of processing in the same log scan. If INCLUDE DDL is YES, Log Master ignores any DB2 catalog tables included in your filter. BMC Software recommends that you use the INCLUDE DDL keyword.

See also “DDLOBJ=NO” on page 73.
INCLUDE LOBS

Determines whether Log Master includes the data from large object (LOB) columns in the output logical log. When the value of this keyword is YES, Log Master writes LOB VSAM files and saves them permanently to disk.

Log Master generates one LOB VSAM file for each LOB column (or each partition of a LOB column) that occurs in the selected log records. Log Master allocates an additional LOB VSAM file only when it has filled the initial data set and all possible extents, but more LOB column data remains to be written. Depending on your environment and your LOB data, the additional disk I/O required for the LOB VSAM files can slow Log Master performance.

The names of the LOB VSAM files are determined by a prefix value that you define. For more information, see “PREFIX (clusterPrefix)” on page 146 or “LOBPREF=&&SYSUID..LOB” on page 71. If you run this log scan repeatedly and use the output files as input to another execution of Log Master, consider using symbolic values such as &DATE. and &TIME. in the prefix to avoid duplicate data set errors. For more information about how Log Master processes LOB columns and data, see the chapter on expert information in the Log Master for DB2 User Guide.

**YES**

Directs Log Master to include LOB data in the output logical log. Log Master writes permanent LOB VSAM files and includes information about those files in the logical log control file.

**NO**

Directs Log Master to avoid including LOB data in the output logical log.

**INLINE**

Directs Log Master to include only inline LOB data in the output logical log, ignoring the auxiliary LOB tables. Log Master writes permanent LOB VSAM files and includes information about those files in the logical log control file.

If you do not specify INCLUDE LOBS, Log Master does not include LOB data. If you specify INCLUDE LOBS without specifying YES, NO, or INLINE, the default value is YES (Log Master includes LOB data).
INCLUDE XML

Determines whether Log Master includes the data from XML columns in the output logical log. When the value of this keyword is YES, Log Master writes XML VSAM files and saves them permanently to disk.

Log Master generates one XML VSAM file for each XML column (or each partition of an XML column) that occurs in the selected log records. Log Master allocates an additional XML VSAM file only when it has filled the initial data set and all possible extents, but more XML column data remains to be written. Depending on your environment and your XML data, the additional disk I/O required for the XML VSAM files can slow Log Master performance.

The names of the XML VSAM files are determined by a prefix value that you define (for more information, see “PREFIX (clusterPrefix)” on page 149 or “XMLPREF=&amp;SYSUID..XML” on page 73). If you run this log scan repeatedly and use the output files as input to another execution of Log Master, consider using symbolic values such as &DATE. and &TIME. in the prefix to avoid duplicate data set errors.

YES

Directs Log Master to include XML data in the output logical log. Log Master writes permanent XML VSAM files and includes information about those files in the logical log control file.

NO

Directs Log Master to avoid including XML data in the output logical log.

If you do not specify INCLUDE XML, Log Master does not include XML data. If you specify INCLUDE XML without specifying YES or NO, the default value is YES (Log Master includes XML data).

INCLUDE COMMAND

Determines whether Log Master includes DB2 command records in the output logical log. When you specify YES or ONLY, Log Master includes all DB2 commands that were issued during the specified time range. Log Master does not filter on command name in the output logical log.

This keyword is valid when Log Master is running on a DB2 Version 9 or later subsystem.

NO

Directs Log Master to avoid including DB2 command record information in the output logical log. The default value is NO.
YES

Directs Log Master to include DB2 command record information in the output logical log.

ONLY

Directs Log Master to include only DB2 command record information in the output logical log.

If you do not specify INCLUDE COMMAND, Log Master does not include DB2 command record information. If you specify INCLUDE COMMAND without specifying YES or NO, the default value is YES (Log Master includes DB2 command record information).

Column include/exclude

The Column include/exclude definition enables you to control the information in your logical log file based on table columns. If you include columns in (or exclude columns from) your logical log, the same columns are included or excluded from all other forms of output that are defined in the log scan.

For example, if you request both logical log and SQL output files, and you exclude the ROWID column of a LOB column from the output logical log file, Log Master excludes the LOB column from the output SQL file as well. In this situation, Log Master cannot generate the SQL for the LOB column because the ROWID was removed from the logical log record that it was processing. This limitation also applies to the DOCID column of an XML column.

For more information about including or excluding column data, see “LOGSCAN column include/exclude definition” on page 267.
LOGSCAN SQL file definition

Figure 22 shows the SQL file definition syntax of the LOGSCAN statement. This syntax specifies overall characteristics of the SQL statements that Log Master creates using the log records selected in your log scan. For more information about Unicode characters in SQL output files, see “Considerations for output files and Unicode” on page 229.

Figure 22  SQL file definition syntax diagram

SQL type/output definition

The SQL type/output definition defines the type of SQL that Log Master generates and where it writes the generated SQL. For more information, see “SQL type/output definition” on page 214.
QUALIFY

Specifies the columns that Log Master includes in the WHERE clauses of generated SQL. By default, Log Master uses available index information to determine what columns to include. To explicitly control the columns used, enter one of the following choices:

ALL

Specifies that Log Master includes all columns in the generated WHERE clauses.

PRIMARY KEY

Specifies that Log Master includes the column or columns associated with a table’s primary key index in the generated WHERE clauses.

If no primary key index exists for the table, Log Master selects an index using DB2’s internal list of indexes for that table. Log Master uses the following hierarchy to select an index. (The order of DB2’s internal list can be different than the order you receive when you select records in the SYSIBM.SYSINDEXES table of the DB2 catalog.)

1. If no primary key index exists, Log Master selects the first unique key index that it encounters.

2. If no unique key index exists, Log Master selects the first clustering key index that it encounters.

3. If no clustering key index exists, Log Master selects the first index that it encounters.

If a primary key or a unique key index exists, Log Master includes the columns associated with that index in the generated WHERE clauses. If no primary or unique key index exists, Log Master issues message BMC097492 as a warning and includes the columns associated with the first non-unique index it encounters. If no indexes exist at all, Log Master issues the warning message and includes all columns in the generated WHERE clauses.

NOTE

In DB2 Version 10 and later, a hash overflow index can also be the primary key or a unique key index.

CHANGED COLUMNS

Specifies whether Log Master includes all changed columns in the generated WHERE clauses. If you specify the ALL keyword, Log Master ignores this keyword.
**USE OVERRIDES**

Directs Log Master to include a unique set of columns (called an alternate index) in the WHERE clause. The alternate indexes must have been defined for the current work ID through the Log Master online interface.

**TSOID tsoID**

Directs Log Master to include the columns of an alternate index in the WHERE clause. The keyword enables you to use alternate indexes that have been defined by a different user ID than the one running the product. The alternate indexes must have been defined through the Log Master online interface as defaults for the user ID specified by tsoID.

**VERBOSE**

Directs Log Master to embed ANSI-standard comments in the generated SQL. These comments describe unit of recovery identifier (URID) information and other information related to the database structures. This information is useful under some circumstances. This keyword is optional.

Regardless of whether you specify this keyword, Log Master can include comments in the generated SQL to provide information about conditions detected during SQL generation. (For examples, see page 267 and page 230.)

**COMMIT FREQUENCY**

Specifies how frequently Log Master includes COMMIT statements in the generated SQL. The value can represent a number of SQL statements or a number of transactions. In this context, transactions are considered to be the same as unit of recovery identifiers (URIDs).

**NOTE**

Log Master enables you to express commit frequency for SQL generation differently than for DDL generation. You can express the SQL frequency as a number of statements or a number of transactions. You must express the DDL frequency as a number of statements.

If you specify INCLUDE DDL in the SQL type/output definition, Log Master treats the commit frequency value as a number of statements for DDL and a number of transactions or statements for SQL, depending on what you specify.

**NONE | 0**

Specifies that all statements or transactions are committed at one time. If you set this value to zero, Log Master interprets it as equivalent to the NONE keyword (all statements or transactions at once).
Specifies the number of SQL statements or transactions (URIDs) between generated COMMIT statements. If you set the value to one and use the TRANSACTIONS keyword, Log Master inserts COMMIT statements in the output SQL at the same points where they were included in the original SQL.

**TRANSACTIONS**

Directs Log Master to interpret the commit frequency as a number of transactions (units of recovery).

**STATEMENTS**

Directs Log Master to interpret the commit frequency as a number of statements.

If you omit the COMMIT FREQUENCY keyword, the default value is equivalent to one transaction (insert COMMIT statements in output at the same points as in the original SQL). If you specify a value without specifying STATEMENTS or TRANSACTIONS, the default value is TRANSACTIONS.

**UPDATE ALL COLUMNS**

Directs Log Master to generate SQL that updates all columns in a table row (whether the column is changed or not). If you omit this keyword, Log Master generates SQL that updates only the changed columns in a row.

**DECIMAL POINT**

Determines which character Log Master uses to represent the decimal point in the generated SQL. Log Master extracts the default character for DECIMAL POINT from the DSNHDECP module of the subsystem for BSDS and DB2 log input.

**PERIOD**

Defines the decimal point character as a period (.)

**COMMA**

Defines the decimal point character as a comma (,).

When Log Master obtains input from a logical log, it uses the same decimal point character that was in use when the logical log was created.
**LOGSCAN SQL file definition**

---

**SQLID sqlid**

Directs Log Master to generate a SET CURRENT SQLID statement and insert it at the beginning of the generated SQL. When you execute the generated SQL, DB2 runs it under the authority granted to the authorization ID represented by sqlid.

Specify a valid authorization ID that has the appropriate authority in your environment.

---

**CONCAT OPERATOR**

Directs Log Master to use the specified characters as the SQL concatenation operator in output SQL statements. Log Master uses a concatenation operator to generate SQL statements that contain any string constants that are longer than the maximum length supported by the current version of DB2 (for example, 255 or 32,704 characters).

---

**CONCAT**

Defines the concatenation operator as the keyword CONCAT. This is the default value. For example:

| UPDATE owner.tableName SET column01='text01' CONCAT 'appendedText' WHERE column01 LIKE 'text01%'; |

---

**BARS**

Defines the concatenation operator as vertical bars (||). For example:

| UPDATE owner.tableName SET column01='text01' || 'appendedText' WHERE column01 LIKE 'text01%'; |

---

**INCLUDE or EXCLUDE INSERT COLUMN NAMES**

Directs Log Master to include or exclude column names in SQL INSERT statements. The default value is INCLUDE INSERT COLUMN NAMES.

By excluding column names and including only the values, SQL files are smaller.

---

**INCLUDE INSERT COLUMN NAMES**

Includes column names in SQL INSERT statements. This is the default value.
EXCLUDE INSERT COLUMN NAMES

Excludes column names from SQL INSERT statements.

**NOTE**

For ALTER-added columns, generated-always columns, or LOB auxiliary table spaces with LOG NO, excluding INSERT column names might result an error in SQL.

**Column include/exclude**

The Column include/exclude definition enables you to control the information in your SQL file based on table columns. For more information, see “LOGSCAN column include/exclude definition” on page 267.
Figure 23 shows the SQL type/output definition syntax of the LOGSCAN statement. This syntax specifies the type of SQL statements included in and the location of the output SQL file that Log Master creates using the log records selected in your log scan.

Figure 23 SQL type/output definition syntax diagram

1 See page 221 for default values of INCLUDE.
MIGRATE

Directs Log Master to generate SQL used for migration, duplicating the SQL that was originally executed. The generated SQL reflects the database changes contained in the log records specified by your WHERE clause or filter. For example, if the original SQL contains an INSERT statement, the MIGRATE SQL contains the same statement.

UNDO

Directs Log Master to generate UNDO SQL, reversing the statement type of the SQL that was originally executed. The generated SQL reverses the database changes contained in the log records specified by your WHERE clause or filter. For example, if the original SQL contains an INSERT statement, the UNDO SQL contains a corresponding DELETE statement.

Log Master cannot reverse the database changes that result from a mass delete action performed on a table defined with Data Capture None (DCN) if the table is stored in a segmented or universal table space. (For more information, see “Considerations for output files and SQL” on page 225.)

EXECUTE

Directs Log Master to execute the generated UNDO SQL immediately after it is generated, without any review or intervention. This keyword is most frequently used in test environments where any unintended consequences do not affect important production data.

REDO

Directs Log Master to generate REDO SQL, so that you can reapply a set of transactions after a recovery action, while omitting a set of “problem” transactions defined by your Range definition and WHERE clause. For example, assume some changes have been made to a set of table spaces in error. You can

1. define the problem changes with a Range definition and WHERE clause

2. generate REDO SQL

3. recover the set of table spaces to a point in time before the problem changes

4. execute the REDO SQL and reapply all of the changes after that point in time, except the problem changes.
The following diagram shows how Log Master uses the Range definition and WHERE clause to generate REDO SQL.

![Diagram showing Log Master's use of Range definition and WHERE clause to generate REDO SQL]

For REDO SQL, Log Master starts by selecting all of the log records that are

- within the table spaces defined (directly or indirectly) by your WHERE clause

To generate REDO SQL, your WHERE clause must refer to at least one specific DB2 object, such as a table space, table, or column.

- within the period between the REDO recovery point and the current time

To generate REDO SQL, you must define a REDO recovery point. For more information, see “REDO FROM” on page 271.

From this set of log records, Log Master excludes all of the log records

- defined by your WHERE clause
- defined in your Range definition (the FROM and TO values)

**NOTE**

You should understand the difference between MIGRATE and REDO SQL. Both types duplicate the original SQL. If the original SQL contains an INSERT statement, both MIGRATE and REDO SQL contain duplicate INSERT statements. The differences are as follows:

- MIGRATE SQL contains all of the changes within the applicable table spaces that satisfy the WHERE clause; REDO SQL contains all of the changes that do not satisfy the WHERE clause.
- MIGRATE SQL contains all changes within your Range definition; REDO SQL contains all of the changes from the REDO recovery point to the current time that are not within your Range definition.

BMC Software strongly recommends that you generate REDO SQL before you perform the recovery. You might be able to generate the SQL after the recovery by using the PROCESS PITS keyword of the OPTION statement, but this practice is not recommended. For more information, see “PROCESS PITS” on page 128.
Also remember that when you generate REDO SQL, Log Master extends the endpoint of the log scan to the current date and time. This action causes Log Master to process more log records than you might expect.

**DATASET Output definition**

Specifies the characteristics of the output data set that contains the generated SQL. For more information, see “LOGSCAN output definition” on page 260.

**TEMPLATE Output definition**

Specifies the characteristics for the output data set that contains the template information for the generated SQL. For more information, see “LOGSCAN output definition” on page 260.

The template file is optional, but BMC Software recommends that you create it if you intend to execute the generated SQL with Log Master (either by using an EXECSQL statement or by using a separate job or job step to run the High-speed Apply Engine). The template file

- improves performance when the High-speed Apply executes the generated SQL
- contains one entry for each distinct statement type in the generated SQL
- provides information that High-speed Apply uses to generate a static plan for the SQL statements
- can be used with a version of the High-speed Apply Engine against a DB2 Universal Database (UDB) target on distributed systems platforms (download both the SQL file and the template file, edit the first line of the SQL file, replace the previous name with the fully qualified path name of the template file, and execute the SQL file with High-speed Apply)

For more information about the High-speed Apply Engine that is distributed with Log Master, see the Log Master for DB2 User Guide.

**RECFM**

Determines the record format of the output SQL file.

**FB**

Indicates that records are fixed length and blocked. This is the default value.
**VB**

Indicates that records are variable length and blocked. This value allows individual SQL statements to be written on a single record, regardless of their length. This type of file can be easier to process on other platforms. When you specify this value, the following items apply:

— Log Master ignores the presence of the VERBOSE keyword.
— Dynamic SQL processors (SPUFI or DSNTEP2) might not process the file.
— Log Master sets the logical record length of the output file (LRECL) to the system-determined block size (SDB) of the storage device where the output file is written (minus 4 bytes).

**INCLUDE DDL**

Indicates whether Log Master generates data definition language (DDL) statements and includes them in the same output as the generated SQL. Log Master generates DDL to match the SQL type (MIGRATE or UNDO) that you requested. Log Master does not generate REDO DDL.

**YES**

Log Master includes DDL statements in the generated SQL.

**NO**

Log Master does not include DDL statements.

If you omit the INCLUDE DDL keyword, the default value is NO (Log Master does not include DDL statements). If you specify the INCLUDE DDL keyword, but omit the YES or NO keywords, the default value is YES (DDL statements are included).

**INCLUDE RI**

Indicates whether Log Master is to include log records that reflect changes resulting from referential integrity (RI) constraints when generating SQL output. This keyword does not cause Log Master to add tables to the filter; it only determines whether Log Master includes RI changes (in addition to normal changes) for the tables that are selected by the filter. To include a complete set of RI changes, you must specify all of the tables involved in the RI constraints in your WHERE clause or filter.

**NOTE**

Correctly setting this parameter can be complicated. For more information, see the discussion of working with changes resulting from referential integrity in the *Log Master for DB2 User Guide*. 
The default value for this keyword depends on the type of SQL that you are generating:

<table>
<thead>
<tr>
<th>SQL type</th>
<th>Default value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDO</td>
<td>NO</td>
<td>Normally, you do not need RI updates when you use REDO SQL to reapply transactions after a recovery action.</td>
</tr>
<tr>
<td>MIGRATE</td>
<td>NO</td>
<td>Whether to include RI updates depends on the definition on the target system.</td>
</tr>
<tr>
<td>UNDO</td>
<td>YES</td>
<td>Normally, you need RI updates when you use UNDO SQL to back out transactions.</td>
</tr>
</tbody>
</table>

**YES**

Log Master includes changes that result from referential integrity constraints.

**NO**

Log Master does not include referential integrity changes. Setting this keyword to NO causes Log Master to exclude referential integrity changes. If you have specified any additional tables in your log scan as mentioned in the preceding paragraphs, Log Master still includes any changes that do not result from referential integrity constraints.

**Considerations for migration:**

This keyword can be important when you migrate data from a source database environment to a target environment. During migration, your decision to include RI changes depends on how the RI constraints are defined in the target environment, as follows:

<table>
<thead>
<tr>
<th>Are the RI constraints the same in the target and source environments?</th>
<th>INCLUDE RI value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>NO</td>
<td>You probably do not want to include the additional RI changes because the constraints can cause duplicate changes.</td>
</tr>
<tr>
<td>No</td>
<td>YES</td>
<td>You might want to include the RI changes. If so, include all of the affected tables in the WHERE clause or filter.</td>
</tr>
</tbody>
</table>
INCLUDE TRIGGER

Indicates whether Log Master includes any log records in the generated SQL that reflect changes that result from activity defined within a trigger. The activity can be performed directly within the trigger or by other processes nested beneath the trigger (for example, in a stored procedure called from within a trigger).

**YES**

Log Master includes changes that result from trigger activity. For consistency with earlier versions of the product, this is the default value.

**NO**

Log Master does not include trigger changes. Setting this keyword to NO causes Log Master to exclude trigger changes. If you have specified any additional tables in your log scan, Log Master still includes any changes to those tables that do not result from trigger activity.

This keyword does not cause Log Master to search additional tables for changes resulting from triggers, it only determines whether Log Master includes the trigger changes that exist in the log records selected by your log scan. To include a complete set of trigger changes, you must specify all of the tables affected by trigger activity in your WHERE clause or filter.

Use this keyword when a target database environment defines the same triggers and relationships as the source environment. In this situation, specify NO. Log Master excludes any trigger records from the output file. When the generated output is processed in the target environment, the triggers are activated as they were in the source environment, and you avoid any duplication of database changes. Before you generate and execute MIGRATE SQL, consider the triggers and other constraints that are defined in the target environment carefully.

To select log records resulting from trigger activity, a log scan’s input files must be either

- generated by a DB2 Version of 9.1 or later (for DB2 log files, from the BSDS or specified by INPUT DB2LOG)
- generated by a Log Master version of 7.3.00 or later running against a DB2 Version of 9.1 or later (for input logical log files)
INCLUDE ROLLBACK

Determines whether Log Master includes information from compensated log records in the output SQL file. Use the default value of NO for normal processing. In this context, compensated log records are the log records of database changes that DB2 subsequently “compensates for” or “reverses” (for example, log records of changes that are later reversed by a ROLLBACK statement). Depending on the value of this keyword, Log Master includes or excludes compensated log records. For more information, see “INCLUDE ROLLBACK” on page 192.

NO

Directs Log Master to exclude compensated log records. The output SQL file does not reflect any compensated log records. Use this default value for normal processing.

YES

Directs Log Master to include compensated log records. The output SQL file reflects both compensated log records and noncompensated log records.

ONLY

Directs Log Master to include only compensated log records. The output SQL file does not reflect any noncompensated log records.

WARNING

Exercise caution before specifying a value of YES or ONLY. In most circumstances, when you include compensated log records, your output SQL file includes data that has not been applied to the original DB2 database. For example, if you migrate data from one database to another and you use an output SQL file that reflects compensated log records, you can corrupt the data in your target database.

If you do not specify INCLUDE ROLLBACK, Log Master ignores compensated log records. If you specify INCLUDE ROLLBACK without specifying YES or NO, the default value is YES (Log Master includes both compensated and noncompensated log records).
**INCLUDE LOBS**

Determines whether Log Master includes the data from large object (LOB) columns in the output SQL file. When the value of this keyword is **YES**, Log Master writes temporary LOB VSAM files during processing, but does not save them permanently to disk unless the log scan includes an output logical log file.

Log Master generates one LOB VSAM file for each LOB column (or each partition of a LOB column) that occurs in the selected log records. Log Master allocates an additional LOB VSAM file only when it has filled the initial data set and all possible extents, but more LOB column data remains to be written. Depending on your environment and your LOB data, the additional disk I/O required to process LOB VSAM files can slow Log Master performance.

Log Master includes the LOB data as part of the generated SQL statements (by using character representation of hexadecimal values, for example `x'407AC2'`). If the length of a statement exceeds the limit imposed by your version of DB2, Log Master issues a warning message, omits the statement, and continues processing.

Log Master cannot include LOB column data in the UNDO SQL that is generated for delete and some insert actions. For more information, see “Large volume columns and SQL” on page 227. For more information about how Log Master processes LOB columns and data, see the *Log Master for DB2 User Guide*.

**YES**

Directs Log Master to include LOB data in the output SQL file. Log Master writes temporary LOB VSAM files and includes LOB data in the generated SQL statements.

**NO**

Directs Log Master to avoid including LOB data in the output SQL file.

**INLINE**

Directs Log Master to include only inline LOB data in the output SQL file, ignoring the auxiliary LOB tables. Log Master writes permanent LOB VSAM files and includes LOB data in the generated SQL statements.

If you do not specify INCLUDE LOBS, Log Master does not include LOB data. If you specify INCLUDE LOBS without specifying YES, NO, or INLINE, the default value is YES (Log Master includes LOB data).
INCLUDE XML

Determines whether Log Master includes the data from XML columns in the output SQL file. When the value of this keyword is YES, Log Master writes temporary XML VSAM files during processing, but does not save them permanently to disk unless the log scan includes an output logical log file.

Log Master generates one XML VSAM file for each XML column (or each partition of an XML column) that occurs in the selected log records. Log Master allocates an additional XML VSAM file only when it has filled the initial data set and all possible extents, but more XML column data remains to be written. Depending on your environment and your XML data, the additional disk I/O required to process XML VSAM files can slow Log Master performance.

Log Master includes the XML data as part of the generated SQL statements. If the length of a statement exceeds the limit imposed by your version of DB2, Log Master issues a warning message, omits the statement, and continues processing.

**YES**

Directs Log Master to include XML data in the output SQL file. Log Master writes temporary XML VSAM files and includes XML data in the generated SQL statements.

**NO**

Directs Log Master to avoid including XML data in the output SQL file.

If you do not specify INCLUDE XML, Log Master does not include XML data. If you specify INCLUDE XML without specifying YES or NO, the default value is YES (Log Master includes XML data).

INCLUDE HISTORY

Indicates whether Log Master includes changes to history tables that are associated with system-maintained temporal tables in the generated SQL. This keyword does not cause Log Master to search additional tables for changes; it only determines whether Log Master includes the history changes that exist in the log records that are selected by your log scan.

DB2 does not have a flag in the history log record to indicate whether the record was logged as a result of activity performed on the associated system-maintained table, or as a result of DML activities performed on the history table by a user. Therefore, Log Master includes all history records in the generated SQL when the value of this keyword is YES, and the history tables are explicitly or implicitly specified in the WHERE clause or filter.
When you generate MIGRATE or REDO SQL, the default value for this keyword is NO. When you generate UNDO SQL, the default value is YES.

**YES**

Log Master includes all changes to the history tables in the output SQL file. You must also specify the history tables in the WHERE clause or filter.

**NO**

Log Master does not include changes to history tables in the output SQL file, even if you specify the history tables in the WHERE clause or filter.

**Considerations for including history changes**

When you work with system-maintained temporal tables, be aware of the following points:

- Because the SYSTEM_TIME start and end columns are defined as GENERATED ALWAYS, any SQL output that includes activities that were performed on the system-maintained table might not be executable. To exclude the GENERATED ALWAYS columns from the SQL output, use the EXCLUDE COLUMNS clause in the log scan statement. For more information, see “LOGSCAN column include/exclude definition” on page 267.

- When you generate MIGRATE SQL, if you include history changes, the SQL might not be executable, or might not keep the system-maintained and the history tables in sync. For instance, when applying the generated SQL, the versioning relationship might result in duplicate INSERTS to the history table. In addition, DB2 generates new timestamp values for the GENERATED ALWAYS columns when the migrated SQL is applied.

- When you perform an UNDO or drop recovery, unpredictable results can occur if you attempt to recover the base table and its history table as a set, and also attempt to maintain history information.
The INCLUDE HISTORY keyword can be important when you migrate data from a source database environment to a target environment. During migration, your decision to include history table changes depends on how the tables are defined in the target environment:

— If the history tables are the same in the target environment as in the source environment, you probably do not want to include the additional history changes because the constraints can cause duplicate changes. Set INCLUDE HISTORY to NO.

— If the history tables in the target environment are not set up the same as the source environment, you might want to include the history tables. If so, include all of the affected tables in the WHERE clause or filter, and set INCLUDE HISTORY to YES.

**NOTE**

You might need to disable the versioning relationship between the target system-maintained and history tables. Doing so ensures that, when updates are applied to the base table, the history table does not record the wrong history information. After the migration is complete, enable versioning again.

The INCLUDE HISTORY keyword is not valid for report and logical log outputs; however, if the filter includes a history table, Log Master includes all history records in the output.

### Considerations for output files and SQL

This section describes some unusual situations that can occur in the DB2 log. These situations are infrequent and they normally result from the operation and properties of DB2. Some of these situations can cause inconvenient results when you execute the SQL statements that Log Master generates based on the DB2 log records. On rare occasions, these situations might require you to edit the SQL statements or change your DB2 environment.

#### Logging of LOAD RESUME LOG YES and LOAD REPLACE LOG YES actions

A LOAD RESUME YES LOG YES action causes your DB2 Load utility to add new rows to an existing DB2 table. Under normal circumstances, the DB2 log contains a log record for each row that is added to the table. For a LOAD RESUME LOG YES action, the DB2 log contains a log record for each page of rows that are added to the table.
The log record for at least one page might contain both rows that were loaded in the LOAD RESUME action, and rows that existed in the table before the LOAD RESUME action (if the pre-existing rows were on the same page as the rows that were loaded).

The DB2 log does not distinguish the loaded rows from the pre-existing rows. If you generate SQL based on the LOAD RESUME log records, Log Master processes both the loaded rows and the pre-existing rows.

This scenario can also occur with log records that represent a LOAD REPLACE LOG YES action. In some cases, when a load action includes a very large number of rows, DB2 internally divides the load action into separate units of recovery. If the log record for a page of rows is not complete in one unit of recovery, the next unit of recovery includes the complete page, starting from the beginning. Any rows that were written in the previous unit of recovery are duplicated in the next unit.

For both LOAD RESUME LOG YES and LOAD REPLACE LOG YES actions, the following results can occur:

- For UNDO SQL, the generated SQL can contain extra DELETE statements.
- For REDO or MIGRATE SQL, the generated SQL can contain extra INSERT statements. When you execute the extra INSERT statements against a table with a unique key, they fail with SQL code -803.

To resolve this situation, edit the generated SQL to remove the extra INSERT or DELETE statements.

### Duplicate rows in input LOAD data

When you load data into a table that is defined with a unique key, and the load action results in multiple records with the same value for the unique key (duplicate rows), Log Master generates SQL statements that can encounter errors or warnings when they are executed.

- For a LOAD REPLACE LOG YES or a LOAD RESUME LOG YES on a table that does not have a row with the same unique key on the table before the load action, DB2 logs the insert records for the initial load action and the DELETE statements to delete duplicate rows. Log Master generates INSERT statements that correspond to the initial load action, but does not generate the DELETE statements for DB2’s subsequent actions to delete the duplicate keys. The second INSERT statement fails with a -803 SQL code.

The results of the load action and the executed SQL statements are the same.

- For a LOAD REPLACE LOG NO on a table that does not have a row with the same unique key on the table before the load action, Log Master generates only one MASS DELETE statement that is associated with the LOAD REPLACE (because the initial load action is not logged).
The results of the load action and the executed SQL statements are the same.

- For a LOAD RESUME LOG YES on a table that has a row with the same unique key on the table before the load action, Log Master generates an INSERT statement (initial load action), but does not generate the DELETE statement to delete the duplicate key. The INSERT statement fails with a -803 SQL code.

In this case, it is important that Log Master does not generate DELETE statements. Doing so would delete the pre-existing row from the table, and could result in data loss.

The results of the load action and the executed SQL statements are not the same.

- For a LOAD RESUME LOG NO, regardless of whether the table has a row with the same unique key on the table before the load action, Log Master does not generate any SQL statements associated with the load action.

**Large volume columns and SQL**

DB2 logs changes to LOB objects differently than changes to non-LOB objects. The log records of a delete action that affects a LOB column do not include the LOB column data. Similarly, the log records of an update to a LOB column do not contain the “before image” of the LOB column.

Despite these differences, Log Master can generate UNDO SQL to undo the effect of a delete action on a LOB column. Log Master can generate UNDO SQL to undo an update action on a LOB column (depending on the presence of image copies or other log records).

This restriction does not apply to

- Inline LOB columns when the LOB is contained entirely in the inline section, and you use Data Capture Changes (DCC).

- XML columns.

The XML table space or LOB table space that corresponds to a large volume column (or a partition of the column) can be logged or not logged at the discretion of database administration personnel. If a large volume table space is not logged, Log Master cannot generate SQL for changes to the corresponding large volume column. In this situation, Log Master generates SQL statements for changes to all other columns, but does not generate changes for either the unlogged column or the corresponding table space.
Mass delete actions and SQL

Log Master cannot generate UNDO SQL to reverse the database changes that result from a mass delete action (such as a TRUNCATE statement or a DELETE statement with no WHERE clause) when all of the following conditions exist:

- The mass delete affects a table that is defined with Data Capture None (DCN).
- The table is stored in a segmented or a universal table space.

Because of the way DB2 logs the mass delete, Log Master cannot generate UNDO SQL. To prevent this situation, define the table with Data Capture Changes (DCC). This definition causes DB2 to log individual delete actions for each record in the table, and enables Log Master to generate UNDO SQL to reverse the delete actions.

Similarly, the log records that DB2 creates for a LOAD REPLACE LOG NO action also prevent Log Master from generating UNDO SQL. You can also use the GENERATE MASSDELETE keyword of the OPTION statement to direct Log Master to ignore the log records that result from a LOAD REPLACE LOG NO action, but depending on your situation, the generated SQL might not meet your needs.

Log Master can generate MIGRATE or REDO SQL to represent a mass delete action. Under version 9 and later versions of DB2, Log Master generates a TRUNCATE statement for the selected table.

GENERATE ALWAYS with ROWID / AS IDENTITY columns

The following points relate to a column that is defined with a data type of ROWID or as an identity column (with the AS IDENTITY syntax):

- When you define a table with either a ROWID or an identity column and use the syntax GENERATE ALWAYS for that column, DB2 creates and maintains a unique value in the column. DB2 does not permit an SQL statement to insert or update a value in this type of column.

- When DB2 logs an INSERT or DELETE action on the table, the log record contains the current value stored in the ROWID or identity column. If Log Master generates an SQL INSERT statement based on the log record, it includes the current value.
Considerations for output files and Unicode

Because the generated SQL statement attempts to insert a value in a column defined as GENERATE ALWAYS, it fails with a negative SQL code. To resolve this situation, perform one of the following actions based on your environment:

- Exclude the ROWID or identity column from the generated SQL statements. For more information about excluding columns from generated output, see “LOGSCAN column include/exclude definition” on page 267. When you execute the generated SQL statement, the INSERT is successful, but DB2 creates a unique value in the ROWID or identity column that is different than the value that was originally associated with the table row.

- Change the definition of the column to use the GENERATE BY DEFAULT syntax instead of GENERATE ALWAYS. When you execute the generated SQL statement, the INSERT is successful (unless the value happens to be the same as a value that already exists in the table).

Considerations for output files and Unicode

When data from the DB2 log contains Unicode characters, Log Master takes different actions depending on the context where the characters occur:

- when Unicode characters occur in DB2 object names, Log Master translates Unicode to EBCDIC

- when Unicode characters occur in column data, Log Master translates Unicode to EBCDIC except for the following types of programmatically readable output
  - data portion (not record header) of records in logical log data files
  - data portion (not record header) of records in load data files

When Log Master cannot successfully translate one or more characters from Unicode to EBCDIC and must write output that contains those characters, it takes the following actions:

- issues a warning message to provide hexadecimal representation of the original Unicode characters (message BMC097411 for DB2 object names or message BMC097231 for column data)

- includes substitute characters in the output as defined by the applicable CCSID conversion information (for example, x’3F’)

  (In SQL output, Log Master uses hexadecimal representation to include the original Unicode characters.)

- continues processing
Table 15 shows the additional actions that Log Master can take depending on the form of output.

### Table 15  Additional actions for untranslatable Unicode characters

<table>
<thead>
<tr>
<th>Context of untranslatable characters</th>
<th>Type of output file</th>
<th>Product action</th>
</tr>
</thead>
</table>
| DB2 object names                     | SQL statements      | Embeds additional comments in SQL output file with hexadecimal representation of the original Unicode characters of the object name, and includes substitute characters (for example, x’3F’) in object name.  
  
  **Note**: If you execute the generated SQL file without modification, you can encounter -204 or -206 SQL codes, indicating unidentified table or column names. To avoid these conditions, generate logical log output and use High-speed Apply to apply the updates.  
  
  reports                               | Includes substitute characters in object names. |
| logical log files                     | In the control file:  
  * Updates several control file records by setting “flag” fields and writing original Unicode object names into designated fields (XTYP page 333, DTSI page 346, DTBI page 347, DCLI page 352, DCL2 page 354, DLCI page 355, or DIXI page 362).  
  * Other logical log control file fields include DB2 object name with substitute characters  
  In the data file, includes substitute characters in object names within header of data records.  
| load files                            | Includes substitute characters in object names. |
| DB2 object names                      | DDL statements      | Issues message BMC097426 as a warning, embeds additional comments in output data definition language (DDL) file, and includes substitute characters in object names.  
  
  **Note**: If you execute the generated DDL file without modification, you can encounter negative SQL codes or create DB2 objects with names that contain the substitute characters. |
| log record data                       | SQL statements      | Uses x’n’nnnnnnnn’ notation to include Unicode characters in data portion of SQL statements (UX ‘xxxxxxxx’ for graphic Unicode characters).  
  
  **Note**: Use caution when migrating data from a Unicode table to an EBCDIC table by using migrate SQL.  
  
  reports                               | Writes additional lines in the report to note the presence of untranslated characters in data, including hexadecimal representation of the original Unicode characters. |
| logical log files                     | Logical log data file includes original Unicode data (no error or warning). |
| load files                            | Load data file includes original Unicode data (no error or warning). |
| DDL statements                        | Issues message BMC097426 as a warning, embeds additional comments in output data definition language (DDL) file, and includes substitute characters in object names. |
Figure 24 shows the Load file definition syntax of the LOGSCAN statement. This syntax specifies the location and format of the load data file and load control file that Log Master creates. When Log Master creates load output, the load data file contains information from the log records that you select, but the records are formatted for a DB2 Load utility (or a file import utility on another platform).

Figure 24  Load file definition syntax diagram (part 1 of 2)
Figure 24  Load file definition syntax diagram (part 2 of 2)
Although Log Master writes a load data file in the DB2 Load utility format, the content is not the same as a load file created by a DB2 Unload utility. The differences are as follows:

- The load data file created by Log Master is basically a set of log records, formatted as a load file. Like most other forms of Log Master output, it reflects DB2 activity over a period of time. Because of this fact, the load data file might contain many load records that affect the same row in a table space. A load file created by a DB2 Unload utility reflects a table space at a given point in time, and therefore contains only one load record for each row in the table space.

- Log Master adds several additional columns (called informational columns) to the specific log record information in the load data file and load control file output. The informational columns give context to the selected log records in the file. For more information, see “INFO COLUMNS” on page 236.

Log Master sorts the records in an output load file (unless you use the NOSORT keyword). It uses the following columns as sort keys (in order): database name, table space name, table creator, table name, and RBA/LRSN. For more information about Unicode characters in output load files, see “Considerations for output files and Unicode” on page 229.

**DATASET Output definition**

Specifies the characteristics of the output load data file. For more information, see “LOGSCAN output definition” on page 260.

**RECFM**

Determines the record format of the output load file.

**VB**

- Indicates that records are variable length and blocked. This is the default value.

**VBS**

- Indicates that records are variable length, blocked, and spanned. You must specify VBS to create an output load file when all of the following conditions exist:
  - The load file includes tables with a page size of 8 KB, 16 KB or 32 KB.
  - The total length of any update record in the load file (including the load record header, the before image, and the after image) is greater than the system-determined block size in your environment.
CONTROL Output definition

Specifies the characteristics of the output load control file. This keyword is optional, reflecting the optional status of the load control file itself. For more information, see “LOGSCAN output definition” on page 260.

The load control file contains a set of control statements for a DB2 Load utility. These control statements define the data contained in the load data file, to

- identify the tables into which data is loaded
- describe fields in the input records (columns in the tables)
- describe the overall format of the load data file

If the load data file contains information from log records of updates, the field specifications in the load control file describe only the columns in the after image of the log records (for more information, see “UPDATES” on page 235).

FORMAT

Specifies the overall format for the load data and load control files. The default value is LOG MASTER. The CSV and SDF formats use character representation for all data to make it easier to move data to other platforms.

LOG MASTER

Indicates that the load data and load control files are formatted using an internal Log Master format. The LOADPLUS for DB2 product available from BMC Software can also read a load file in this format.

UNLOAD PLUS

Indicates that the load data and load control files use the native format of UNLOAD PLUS. This format is similar to the format generated by Log Master, with small differences in the position and content of the null byte used in load records.

- In the Log Master format, the null byte precedes column data, contains $\text{x'FF'}$ to indicate a null column, and contains $\text{x'00'}$ to indicate column data that is not null.

- In the UNLOAD PLUS format, the null byte follows column data, contains $\text{x'6F'}$ (question mark) to indicate a null column, and contains $\text{x'40'}$ (space) to indicate column data that is not null.
**CSV | COMMA SEPARATED VALUE**

Indicates that the load data and load control files are formatted using the Comma Separated Value (CSV) format available in some database software. Individual fields in the load records are separated by commas.

**SDF | STANDARD DEFINITION FORMAT**

Indicates that the load data and load control files are formatted using the Standard Definition Format (SDF) format available in some database software. Individual fields in the load records are filled with spaces so that each field has the same position and length in all load records.

Use the EXCLUDE NULL INDICATOR keyword to prevent Log Master from adding the null byte at the end of each field that corresponds to a nullable column. The byte contains \x'6F' (question mark) for a null column, and contains \x'40' (space) when the column is not null. This keyword applies only to output load files in SDF format; it affects both the load data file and the load control file. The default value causes Log Master to include the null byte (also called a null indicator).

See also “LOADFMT=LOGMAST” on page 69.

**UPDATES**

Specifies which parts of a selected update log record appear in the output load data file. The default value is BOTH SEPARATE.

**BEFORE**

Include only the before image of an update in the load output.

**AFTER**

Include only the after image of an update in the load output.

**BOTH | SEPARATE | TOGETHER**

BOTH includes both the before and after images in the load output. If you specify BOTH, you can choose whether the images appear as SEPARATE load records (the default value) or TOGETHER as a single load record. If you specify this keyword, the records in the load data file contain both before and after images of update log records, but the load control file contains field specifications for only the after image of the update.

The value of this keyword can determine the overall layout of the records in your output load file. For more information, see “Layout of records in load files” on page 249.
INFO COLUMNS

Log Master adds several informational columns to the specific log record information in the load data file and load control file output. Either the INFO COLUMNS keyword or the URID FIELDS keyword control which informational columns are added to the records in the output load file. BMC Software recommends using the INFO COLUMNS keyword. You cannot specify values for both the INFO COLUMNS keyword and the URID FIELDS keyword in the same log scan. Remember the following points when you specify informational columns for an output load file:

- Log Master always adds three informational columns to the load data and load control files to give context to the load records. The load records always contain the LOG RBA, CHANGE TYPE, and UPDATE TYPE columns.

- Log Master includes the TABLE NAME and TABLE CREATOR columns when the value of the SEPARATE DATASETS keyword is YES.

- Log Master includes the TABLE SEQNUM column when any of the following conditions exist:
  - The value of the EXECUTION MODE keyword is OVERTIME.
  - The input source is a logical log file created by Log Master in overtime mode.
  - The INFO COLUMNS (or URID FIELDS ONLY) keyword specifies TABLE SEQNUM as an informational column.

- When you use a load file to move data from one table to another table, you must define additional columns in the target table to match the informational columns included in the load file. If you do not want the informational columns in the receiving table, edit the load control file to filter out the unwanted columns.

- If you specify the value of URID TABLE as YES, Log Master always includes a standard set of informational columns in the separate URID-related information (regardless of the value of either the INFO COLUMNS or URID FIELDS keywords). For a list of the standard informational columns, see “Additional Informational Columns in Load Files” on page 250.

The `infoColName` can be any of the following values:

**AUTH ID**

A character string used to connect to DB2. (It frequently represents an individual, a work group, or a function.)

**CONNECTION ID**

A string identifier associated with a specific address space connection.
**CONNECTION TYPE**

A string constant representing the manner in which a unit of recovery thread is attached to DB2 (for example, BATCH, TSO, UTILITY).

**CORRELATION ID**

A unique string identifier for a specific DB2 thread.

**DATABASE NAME**

The name of the database containing the load records.

**LONG DATABASE NAME**

Reserved for use in future versions of DB2.

**LONG TABLE CREATOR**

The complete authorization ID that was in use when the table represented in the load records was created.

The complete creator name is 128 characters long. If the actual name is shorter than this length, Log Master adds spaces to the end of the name until it is 128 characters long.

**LONG TABLE NAME**

The complete name of the table containing the load records (length and space filling similar to “LONG TABLE CREATOR”).

**LONG TABLESPACE NAME**

Reserved for use in future versions of DB2.

**PARTITION**

The number of a partition (within a partitioned table space) that contains the load records.

**PLAN NAME**

A character string representing an application plan, the control structure used by DB2 to process SQL statements.
**RECORD TIMESTAMP**

The timestamp value associated with the log record that is represented in the individual load record.

**RID**

The record ID (RID) associated with the log record that is represented in the individual load record. The RID contains a page number value and a row ID value.

**TABLE CREATOR**

A shortened version of the authorization ID that was in use when the table represented in the load records was created. When SEPARATE DATASETS is set to the default value of NO, Log Master includes this column even if you do not specify this keyword. Log Master provides the shortened creator name for compatibility with earlier versions of DB2 and earlier versions of logical log records. For the complete table name, see “LONG TABLE CREATOR” on page 237.

The shortened table creator is no longer than 8 characters. If the actual name is longer than this limit, Log Master truncates the complete name and supplies the first 8 bytes in this column.

**TABLE NAME**

A shortened version of the name of the table represented in the load records. When SEPARATE DATASETS is set to the default value of NO, Log Master includes this column even if you do not specify this keyword. Log Master provides the shortened name for compatibility with earlier versions of DB2 and earlier versions of logical log records. For the complete table name, see “LONG TABLE NAME” on page 237.

The shortened name is no longer than 18 characters. If the actual name is longer than this limit, Log Master generates a unique 18-character name and supplies it in this column.

To generate the shortened name, Log Master uses a tilde (~) character and characters that represent the table’s 4-digit hexadecimal database ID (DBID) and object ID (OBID). Log Master truncates the initial portion of the complete name and concatenates it with the generated string so that the combination is no longer than 18 characters. For example, if a table has a DBID of 051C, an OBID of 0007, and a name of A_VERY_DESCRIPTIVE_AND_PRECISE_TABLE_NAME, then the shortened name would be A_VERY_DE~051C0007.
TABLE SEQNUM

A number representing the *instance* of a table during the time frame of your log scan.

If you drop and re-create a table using the same fully qualified name, log records can relate to either the original or the re-created instance of the table. The table sequence number is greater than 1 *only* if the log scan includes log records relating to more than one instance of the same table.

This column becomes important when your job runs in overtime mode or when your input is combined from multiple logical log files (for example, when weekly logical log files are combined to produce a single monthly output). In any situation where log records relate to *multiple* instances of the *same* table name, you can use the table sequence number to select the log records that you require.

TABLESPACE NAME

The name of the table space containing the load records.

URID COMMIT LRSN

The LRSN of the log record where the unit of recovery was committed.

URID COMMIT POINT

The RBA of the log record where the unit of recovery was committed.

URID DISP

The disposition of the unit of recovery (for example, committed or aborted).

URID BEGIN POINT

The RBA of the first log record in a unit of recovery.

URID LRSN

The LRSN of the first log record in a unit of recovery.

URID MEMBER ID

The numeric identifier of the data sharing member associated with the unit of recovery.

URID TIMESTAMP

The timestamp value associated with the start of the unit of recovery.
URID FIELDS

Log Master adds several informational columns to the specific log record information in the load data file and load control file output. Either the URID FIELDS keyword or the INFO COLUMNS keyword control which informational columns Log Master adds to the records in the output load file. BMC Software recommends using the INFO COLUMNS keyword. You cannot specify values for both the URID FIELDS keyword and the INFO COLUMNS keyword in the same log scan.

Some of the informational columns relate to the unit of recovery identifier (URID). The URID FIELDS keyword lets you select certain predefined sets of URID-related informational columns.

**ALL**

Indicates that Log Master includes all of the URID-related informational columns in each load record. For a list of the informational columns that are added, see Table 17 on page 251.

**URID**

Indicates that Log Master includes a specific subset of the URID-related informational columns in each load record. For a list of the informational columns that are added, see Table 18 on page 252.

**NONE**

Indicates that Log Master does not include any of the URID-related informational columns in each load record. Only a default set of informational columns are included. This is the default value. For a list of the informational columns that are added, see Table 19 on page 253.

**ONLY infoColName**

This keyword enables you to control both the URID-related and the other informational columns that are included in each load record. To use the ONLY keyword, you must specify at least one informational column for inclusion. Separate multiple columns with commas.

The *infoColName* can be any value listed in “INFO COLUMNS” on page 236.
**URID TABLE**

Specifies whether Log Master includes a separate record in the output load data file for each unit of recovery. The records contain URID-related information (for example, activity counts). This keyword also determines whether the load control file includes additional control cards to create a separate table for the URID-related information. The name of the separate URID table in the control cards is always #ALPURID.#ALPURIDTABLE.

Log Master always includes the same set of informational columns with the URID-related information, regardless of the value of either the INFO COLUMNS or the URID FIELDS keywords. For more information about the columns included with the URID-related information, see Table 20 on page 253.

**YES**

Directs Log Master to include URID-related information in the load data file and additional control cards in the load control file.

**NO**

Directs Log Master to exclude URID-related information from output load files. This is the default value.

Log Master produces the URID-related information differently, depending on the value of the SEPARATE DATASETS keyword. For more information about this keyword, see “SEPARATE DATASETS” on page 247.

- When SEPARATE DATASETS is YES, Log Master writes the URID-related information into a separate load data file and writes the additional control cards in a separate load control file.

The data set names of the load data file and load control file follow the specifications that you enter, except that Log Master automatically sets the value of the &TABOWN symbolic substitution to #ALPURID and the value of the &TABNAME symbolic to #ALPURIDTABLE. (For more information about symbolics, see “Symbolic substitutions” on page 323.) Do not concatenate any additional characters with the &TABxxxx symbolics; you might generate a node in the data set name that is longer than the operating system allows.

- When SEPARATE DATASETS is NO, Log Master includes the URID-related information in the load data file and includes the additional control cards in the load control file.
EXPAND VAR

Determines how Log Master writes variable length fields in the output load data files and load control files.

**YES**

Directs Log Master to expand (fill) variable length fields with spaces up to the declared maximum length. This is the default value.

**NO**

Directs Log Master not to expand (fill) variable length fields.

Log Master ignores the EXPAND VAR keyword for an output load file in either the SDF or CSV formats. To conform to the requirements of the SDF format, Log Master creates output as if EXPAND VAR is set to YES. To conform to the CSV format, Log Master creates delimited output as if EXPAND VAR is set to NO.

INCLUDE ROLLBACK

Determines whether Log Master includes information from compensated log records in the output load file. Use the default value of NO for normal processing. In this context, compensated log records are the log records of database changes that DB2 subsequently “compensates for” or “reverses” (for example, log records of changes that are later reversed by a ROLLBACK statement). Depending on the value of this keyword, Log Master includes or excludes compensated log records. For more information, see “INCLUDE ROLLBACK” on page 192.

**NO**

Directs Log Master to exclude compensated log records from the output load file. Use this default value for normal processing.

**YES**

Directs Log Master to include compensated log records. The output load file reflects both compensated log records and noncompensated log records.

**ONLY**

Directs Log Master to include only compensated log records. The output load file does not reflect any noncompensated log records.
**WARNING**

*Exercise caution* before specifying a value of YES or ONLY. In most circumstances, when you include compensated log records, your output load file includes *data that has not been applied* to the original DB2 database. For example, if you migrate data from one database to another and you use an output load file that reflects compensated log records, you can *corrupt the data* in your target database.

If you do not specify INCLUDE ROLLBACK, Log Master ignores compensated log records. If you specify INCLUDE ROLLBACK without specifying YES or NO, the default value is YES (include both compensated and noncompensated log records).

**INCLUDE LOBS**

Determines whether Log Master includes the data from large object (LOB) columns in the output load file. When the value of this keyword is YES, Log Master writes temporary LOB VSAM files during processing, *but does not* harden them permanently to disk unless the log scan includes an output logical log file.

Log Master generates one LOB VSAM file for each LOB column (or each partition of a LOB column) that occurs in the selected log records. Log Master allocates an additional LOB VSAM file only when it has filled the initial data set and all possible extents, but more LOB column data remains to be written. Depending on your environment and your LOB data, the additional disk I/O required to process LOB VSAM files can slow Log Master performance considerably.

For more information about how Log Master processes LOB columns and data, see the *Log Master for DB2 User Guide*.

If you do not specify INCLUDE LOBS, Log Master does not include LOB data. If you specify INCLUDE LOBS without specifying YES, NO, or INLINE, the default value is YES (Log Master includes LOB data).

**YES**

Directs Log Master to include LOB data in the output load file. Log Master writes temporary LOB VSAM files and includes LOB data in the output files in one of two ways: within the load data file or in external files. For more information about defining external files, see “TEMPLATE” on page 245.

**NO**

Directs Log Master to avoid including LOB data in the output load file.
INLINE

Directs Log Master to include only inline LOB data in the output load file, ignoring the auxiliary LOB tables. Log Master writes permanent LOB VSAM files and includes LOB data in the output files in one of two ways: within the load data file or in external files. For more information about defining external files, see “TEMPLATE” on page 245.

INCLUDE XML

Determines whether Log Master includes the data from XML columns in the output load file. When the value of this keyword is YES, Log Master writes temporary XML VSAM files during processing, but does not harden them permanently to disk unless the log scan includes an output logical log file.

Log Master generates one XML VSAM file for each XML column (or each partition of an XML column) that occurs in the selected log records. Log Master allocates an additional XML VSAM file only when it has filled the initial data set and all possible extents, but more XML column data remains to be written. Depending on your environment and your XML data, the additional disk I/O required to process XML VSAM files can slow Log Master performance considerably.

YES

Directs Log Master to include XML data in the output load file. Log Master writes temporary XML VSAM files and includes XML data in the output files in one of two ways: within the load data file or in external files. For more information about defining external files, see “TEMPLATE” on page 245.

NO

Directs Log Master to avoid including XML data in the output load file.

If you do not specify INCLUDE XML, Log Master does not include XML data. If you specify INCLUDE XML without specifying YES or NO, the default value is YES (Log Master includes XML data).

IGNOREFIELDS

Determines whether Log Master generates the IGNOREFIELDS clause in the load control file. With the value of YES for this keyword, DB2 will ignore the values for the GENERATED ALWAYS columns that define the PERIOD for a system-maintained temporal table. To retain PERIOD values, use the PERIODOVERRIDE keyword (page 245).

YES

Directs Log Master to generate the IGNOREFIELDS clause in the output load file.
NO

Directs Log Master to avoid generating the IGNOREFIELDS clause in the output load file.

PERIODOVERRIDE

Directs Log Master to generate the PERIODOVERRIDE clause in the output load file so that, when the data is reloaded, DB2 maintains the values of the GENERATED ALWAYS columns that define the PERIOD for a system-maintained temporal table.

YES

Directs Log Master to generate the PERIODOVERRIDE clause in the output load file.

NO

Directs Log Master to avoid generating the PERIODOVERRIDE clause in the output load file.

TEMPLATE

Specifies the large volume columns (such as LOB columns or XML columns) that might appear in selected log records, associates each column with an external file, and defines the characteristics of each external file. Log Master provides two ways to include large volume column data in an output load file.

- inside load data file

  Log Master can include the values of a large volume column within the output load data file. (Log Master writes LOB data as character representation of the hexadecimal values; for example x'401AC2'.) The length of the resulting load record must not exceed the limit imposed by your version of DB2. If the length of a generated load record exceeds the limit, Log Master issues an error message and terminates processing.

- in external files

  Log Master can generate separate, external files that contain the values of a large volume column. The format of the external files conforms to that used by an IBM Load utility.

To embed large volume column data within the load data file, omit the TEMPLATE keyword. To define external files, use the TEMPLATE keyword and associated syntax.
You must define the same output method for all occurrences of the same type of large volume column in the same table. However, you can define different methods for different types of large volume columns in the same table. Consider the following example:

If a table has two LOB columns and two XML columns, you can write the LOB columns into the load data file and the XML columns to external files. You cannot write one LOB column into the load data file and write the other LOB column to an external file.

Large volume column definition

Specifies a large volume column (such as a LOB column or an XML column) that might appear within selected log records and includes a definition of the external file that will contain the values for that column. Define a LOB or XML column as it appears in the table that contains the column.

LOB CREATOR

Defines the creator of the table that contains a LOB column.

XML CREATOR

Defines the creator of the table that contains an XML column.

TABLE NAME

Defines the name the table that contains a LOB or XML column.

COLUMN NAME

Defines the name of a large volume column (such as a LOB column or an XML column) that is associated with or contained in the table defined by the xxx CREATOR and TABLE NAME keywords.

DATASET Output definition

Specifies the characteristics of the external file that is associated with the large volume column defined by the xxx CREATOR, TABLE NAME, and COLUMN NAME keywords. The data set that contains the external file must be a partitioned data set (PDS) or a partitioned data set extended (PDSE, sometimes referred to as a library). For more information about the characteristics that you can define, see “LOGSCAN output definition” on page 260.
SEPARATE DATASETS

Specifies whether Log Master creates separate output files for each *individual table* selected in the log scan. This keyword applies to both the load data file and the load control file.

**NO**

Directs Log Master not to create separate output files. This is the default value.

**YES**

Directs Log Master to create separate output files, resulting in multiple load data files and multiple load control files. If you specify YES, you must mask the names of any data sets that you specify for either the output load data file or the output load control file. You must use at least the &TABNAME symbolic in the file names. Log Master resolves the symbolic substitutions for each table in the load data or load control file. For more information about symbolics, see “Symbolic substitutions” on page 323.

GENERATE EMPTY FILES

Extends the effect of the SEPARATE DATASETS keyword. This keyword directs Log Master to create separate output files for *all* individual tables selected in your log scan regardless of whether Log Master finds log records related to a table. If your WHERE clause or filter selects a table, but the log contains no records related to that table, Log Master creates

- a separate (empty) output load data file that contains no load records (some environments can use the empty load data file to avoid JCL errors)
- a separate output load control file that contains the syntax required for a DB2 Load utility to load data into that table

To use this keyword, your WHERE clause or filter can contain *only* the common filter predicates listed in Table 16 on page 247. In addition, the filter predicates in the WHERE clause or filter can be combined *only* with the OR logical operator. If you use other filter predicates or combine them with the AND or NOT operator, Log Master does not create empty files for tables with no log records.

Table 16  Valid filter predicates, GENERATE EMPTY FILES keyword (part 1 of 2)

<table>
<thead>
<tr>
<th>Selectable field</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE NAME</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td>LIKE</td>
</tr>
<tr>
<td>TABLESPACE NAME</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td>LIKE</td>
</tr>
<tr>
<td></td>
<td>IN</td>
</tr>
</tbody>
</table>
You cannot use this keyword if the value of the FILTER METHOD keyword (or the FLTRMTHD installation option) is DYNAMIC.

**NOSORT**

Directs Log Master to *not* sort the records in the output load data file. When you specify NOSORT, the load records can appear in any order.

If you do not specify the NOSORT keyword, Log Master sorts the records by using the following columns as sort keys (in order):

- database name
- table space name
- table creator
- table name
- RBA/LRSN

BMC recommends caution when specifying NOSORT. Use of this keyword could result in duplicate records being created in successive ongoing log scans (due to the dependent RBA/LRSN).

To avoid duplicate records when using NOSORT

- define tables being extracted with the Data Capture Changes (DCC) attribute
- ensure that application programs in your environment ignore records in the load data file if the records are dependent on a transaction that is incomplete at the end of an ongoing log scan

To determine whether programs should ignore a record, review the dependent RBA/LSRN value stored in the ALPURID Repository table for the run sequence number of the job that created the load data file.

— If the dependent RBA/LSRN value is `x'FFFFFFFFFFFF'`, the load data file does not contain any records that should be ignored.

— If the dependent RBA/LSRN value is not `x'FFFFFFFFFFFF'`, compare it to the informational column LOGRBA in the output load file.

<table>
<thead>
<tr>
<th>Selectable field</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE NAME</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td>LIKE</td>
</tr>
<tr>
<td></td>
<td>IN</td>
</tr>
<tr>
<td>tableName.columnName</td>
<td>=</td>
</tr>
</tbody>
</table>

**Table 16**  Valid filter predicates, GENERATE EMPTY FILES keyword (part 2 of 2)
If the LOGRBA is greater than or equal to the dependent RBA/LSRN value, ignore the record. The same record will appear in the load data file produced by a subsequent run of the ongoing log scan.

If the LOGRBA is less than the dependent RBA/LSRN value, process the record.

For more information about ongoing processing, see page 378 (the section discusses logical log files, however, the ongoing processing information also applies to load data files). For more information about dependent RBA/LRSN values, see page 379. For more information about informational columns, see page 250.

**Column include/exclude**

The Column include/exclude definition enables you to control the information in your load file based on table columns. For more information about including or excluding column data, see “LOGSCAN column include/exclude definition” on page 267.

**Layout of records in load files**

Each record in an output load file represents a log record selected by your filter and time frame. In addition to the row image (or images) of the log record, Log Master includes additional informational columns to give context to the record. The following paragraphs summarize the layout of an output load file record.

**load record header**

The header contains all of the additional informational columns that Log Master includes with the log record. The length and content of the header varies depending on the values of the INFO COLUMNS or URID FIELDS keywords. For more information, see “INFO COLUMNS” on page 236.

**row image**

Depending on the type of database action that the load record represents, and the value of the UPDATES keyword, the content of the row image is as follows:

- insert action: an image of the inserted row
- delete action: an image of the deleted row
- update action: an image of the row after the update is completed
- mass delete action: an image with all default column values
- exchange action: an image with the same load column information leading to the record type, followed by the clone table creator and the clone table name
additional row image (when required)

If the load record represents an update action, the record includes another row image that contains the before image of the row.

The UPDATES keyword determines the position and content of the row image (or images) in an output load file. Figure 25 shows the layout of row images for each value of the UPDATES keyword.

**Figure 25 Load file record layout for different UPDATES values**

- **UPDATES BEFORE**
  - Load Record Header (variable length *)
  - Row Image
  - INSERT Image, DELETE Image, or UPDATE BEFORE Image

- **UPDATES AFTER**
  - Load Record Header (variable length *)
  - Row Image
  - INSERT Image, DELETE Image, or UPDATE AFTER Image

- **UPDATES BOTH TOGETHER**
  - Load Record Header (variable length *)
  - Row Image
  - Row Image
  - INSERT Image, DELETE Image, or UPDATE BEFORE Image
  - UPDATE BEFORE Image (when required)

- **UPDATES BOTH SEPARATE**
  - Load Record Header (variable length *)
  - Row Image
  - INSERT Image, DELETE Image, or UPDATE BEFORE Image
  - UPDATE AFTER Image (when required)

* Length and content of Load Record Header vary depending on the value of URID FIELDS or INFO COLUMNS keywords

---

**Additional Informational Columns in Load Files**

Log Master includes informational columns at the beginning of all records in the output load file, in addition to the columns that you specify in the WHERE clause or filter. For more information about the additional columns and when they are included, see “INFO COLUMNS” on page 236.
If you move data from one table to another by using a load file, you must define additional columns in the target table to match these additional informational columns. The tables on the following pages show the layout of the additional columns and additional records. The tables show the informational columns that Log Master includes when you specify:

- URID FIELDS as ALL (Table 17)
- URID FIELDS as URID (Table 18 on page 252)
- URID FIELDS as NONE (Table 19 on page 253)
- URID TABLE as YES (the additional table that you must define, Table 20 on page 253)

The tables on these pages show the layout of the additional columns or records when the value of the FORMAT keyword is LOG MASTER. If the value is SDF or CSV, the content and order of the columns is the same, but the type and length of the data in a column might be different.

- To find the sequence and length of columns in an SDF load file, look at the output load control file.
- To find the sequence of columns in a CSV load file, look at the output load control file (the length of columns in an output CSV load file varies for each record).

Table 17 shows the informational columns that Log Master includes if you specify URID FIELDS as ALL.

### Table 17  Additional informational columns: URID FIELDS ALL  (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#CONNECTION_TYPE</td>
<td>CHAR(8)</td>
<td>Connection type</td>
</tr>
<tr>
<td>#CONNECTION_ID</td>
<td>CHAR(8)</td>
<td>Connection ID</td>
</tr>
<tr>
<td>#CORRELATION_ID</td>
<td>CHAR(12)</td>
<td>Correlation ID</td>
</tr>
<tr>
<td>#AUTH_ID</td>
<td>CHAR(8)</td>
<td>Authorization ID</td>
</tr>
<tr>
<td>#PLAN_NAME</td>
<td>CHAR(8)</td>
<td>Plan name</td>
</tr>
<tr>
<td>#URIDLRSN</td>
<td>CHAR(6)</td>
<td>Unit of Recovery LRSN (the same as #URIDBEGINPOINT if not data sharing)</td>
</tr>
<tr>
<td>#URIDBEGINPOINT</td>
<td>CHAR(6)</td>
<td>Unit of Recovery ID (RBA)</td>
</tr>
<tr>
<td>#URIDMEMBERID</td>
<td>SMALLINT</td>
<td>Data sharing member ID (0 if not data sharing)</td>
</tr>
<tr>
<td>#URIDTIMESTAMP</td>
<td>CHAR(26)</td>
<td>Unit of recovery date and time</td>
</tr>
<tr>
<td>#URIDDISP</td>
<td>CHAR(1)</td>
<td>Unit of recovery disposition C = Committed A= Aborted</td>
</tr>
<tr>
<td>#URIDCOMMITLRSN</td>
<td>CHAR(6)</td>
<td>Unit of recovery COMMIT LRSN (the same as #UORCOMMITPOINT if not data sharing)</td>
</tr>
<tr>
<td>#URIDCOMMITPOINT</td>
<td>CHAR(6)</td>
<td>Unit of recovery COMMIT (RBA)</td>
</tr>
</tbody>
</table>
Table 17  Additional informational columns: URID FIELDS ALL (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DB_NAME</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>#TS_NAME</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>#TB_CREATOR</td>
<td>CHAR(8)</td>
<td>Table creator</td>
</tr>
<tr>
<td>#TB_NAME</td>
<td>CHAR(18)</td>
<td>Table name</td>
</tr>
<tr>
<td>#TB_SEQNUM</td>
<td>CHAR(3)</td>
<td>Table sequence number (This number can be greater than 1 only if the load file contains records from more than one “instance” of the same table.)</td>
</tr>
<tr>
<td>#LLRECTIMESTAMP</td>
<td>CHAR(26)</td>
<td>Date and time of log record represented in current load record</td>
</tr>
<tr>
<td>#LLPARTNUM</td>
<td>CHAR(2)</td>
<td>Number of partition (within partitioned table space) of log record represented in current load record</td>
</tr>
<tr>
<td>#LLRID</td>
<td>CHAR(5)</td>
<td>Record ID (RID) of log record represented in current load record</td>
</tr>
<tr>
<td>#LOGRBA</td>
<td>CHAR(6)</td>
<td>RBA/LRSN of log record represented in current load record</td>
</tr>
<tr>
<td>#CHANGE_TYPE</td>
<td>CHAR(1)</td>
<td>Type of modification: I = INSERT, D = DELETE, U = UPDATE</td>
</tr>
<tr>
<td>#UPDATE_TYPE</td>
<td>CHAR(1)</td>
<td>Field is valid only when #CHANGE_TYPE is U: A = After Image, B = Before Image, ‘ ’ (blank) = Both images</td>
</tr>
</tbody>
</table>

Table 18 shows the informational columns that Log Master includes if you specify URID FIELDS as URID.

Table 18  Additional informational columns: URID FIELDS URID (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#URIDLRSN</td>
<td>CHAR(6)</td>
<td>Unit of Recovery LRSN (the same as #URIDBEGINPOINT if not data sharing)</td>
</tr>
<tr>
<td>#URIDBEGINPOINT</td>
<td>CHAR(6)</td>
<td>Unit of Recovery ID (RBA)</td>
</tr>
<tr>
<td>#URIDMEMBERID</td>
<td>SMALLINT</td>
<td>Data sharing member ID (0 if not data sharing)</td>
</tr>
<tr>
<td>#DB_NAME</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>#TS_NAME</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>#TB_CREATOR</td>
<td>CHAR(8)</td>
<td>Table creator</td>
</tr>
<tr>
<td>#TB_NAME</td>
<td>CHAR(18)</td>
<td>Table name</td>
</tr>
<tr>
<td>#TB_SEQNUM</td>
<td>CHAR(3)</td>
<td>Table sequence number (This number can be greater than 1 only if the load file contains records from more than one “instance” of the same table.)</td>
</tr>
<tr>
<td>#LOGRBA</td>
<td>CHAR(6)</td>
<td>RBA/LRSN of log record represented in current load record</td>
</tr>
</tbody>
</table>
### Table 19: Additional informational columns: URID FIELDS NONE

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DB_NAME</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>#TS_NAME</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>#TB_CREATOR</td>
<td>CHAR(8)</td>
<td>Table creator</td>
</tr>
<tr>
<td>#TB_NAME</td>
<td>CHAR(18)</td>
<td>Table name</td>
</tr>
<tr>
<td>#TB_SEQNUM</td>
<td>CHAR(3)</td>
<td>Table sequence number (This number can be greater than 1 only if the load file contains records from more than one “instance” of the same table.)</td>
</tr>
<tr>
<td>#LOGRBA</td>
<td>CHAR(6)</td>
<td>RBA/LRSN of the log record represented in current load record</td>
</tr>
<tr>
<td>#CHANGE_TYPE</td>
<td>CHAR(1)</td>
<td>Type of modification:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I = INSERT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D = DELETE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U = UPDATE</td>
</tr>
<tr>
<td>#UPDATE_TYPE</td>
<td>CHAR(1)</td>
<td>Field is valid only when #CHANGE_TYPE is U:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A = After Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B = Before Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’ ’ (blank) = Both images</td>
</tr>
</tbody>
</table>

Table 20 on page 253 shows the format of the additional table that you must define when you specify URID TABLE as YES. The default name of the URID table in the control cards of the load control file is `#ALPURID.#ALPURIDTABLE`. To change the default table name, edit the load control file.

### Table 20: Additional table format: URID TABLE YES (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DB_NAME</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>#TS_NAME</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>#TB_CREATOR</td>
<td>CHAR(8)</td>
<td>Table creator</td>
</tr>
<tr>
<td>#TB_NAME</td>
<td>CHAR(18)</td>
<td>Table name</td>
</tr>
<tr>
<td>#TB_SEQNUM</td>
<td>INT</td>
<td>Table sequence number (This number can be greater than 1 only if the load file contains records from more than one “instance” of the same table.)</td>
</tr>
<tr>
<td>#LOGRBA</td>
<td>CHAR(6)</td>
<td>RBA/LRSN of the log record represented in current load record</td>
</tr>
<tr>
<td>#CHANGE_TYPE</td>
<td>CHAR(1)</td>
<td>Type of modification:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I = INSERT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D = DELETE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U = UPDATE</td>
</tr>
<tr>
<td>#UPDATE_TYPE</td>
<td>CHAR(1)</td>
<td>Field is valid only when #CHANGE_TYPE is U:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A = After Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B = Before Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’ ’ (blank) = Both images</td>
</tr>
</tbody>
</table>
Table 20  Additional table format: URID TABLE YES (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#LOGRBA</td>
<td>CHAR(6)</td>
<td>RBA/LRSN of log record represented in current load record</td>
</tr>
<tr>
<td>#CONNECTION_TYPE</td>
<td>CHAR(8)</td>
<td>Connection type</td>
</tr>
<tr>
<td>#CONNECTION_ID</td>
<td>CHAR(8)</td>
<td>Connection ID</td>
</tr>
<tr>
<td>#CORRELATION_ID</td>
<td>CHAR(12)</td>
<td>Correlation ID</td>
</tr>
<tr>
<td>#AUTH_ID</td>
<td>CHAR(8)</td>
<td>Authorization ID</td>
</tr>
<tr>
<td>#PLAN_NAME</td>
<td>CHAR(8)</td>
<td>Plan name</td>
</tr>
<tr>
<td>#URIDLRSN</td>
<td>CHAR(6)</td>
<td>Unit of Recovery LRSN (the same as #URIDBEGINPOINT if not data sharing)</td>
</tr>
<tr>
<td>#URIDBEGINPOINT</td>
<td>CHAR(6)</td>
<td>Unit of recovery ID (RBA)</td>
</tr>
<tr>
<td>#URIDMEMBERID</td>
<td>SMALLINT</td>
<td>Data sharing member ID (0 if not data sharing)</td>
</tr>
<tr>
<td>#URIDTIMESTAMP</td>
<td>CHAR(26)</td>
<td>Unit of recovery date and time</td>
</tr>
<tr>
<td>#URIDDISP</td>
<td>CHAR(1)</td>
<td>Unit of recovery disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C = Committed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A= Aborted</td>
</tr>
<tr>
<td>#URIDCOMMITLRSN</td>
<td>CHAR(6)</td>
<td>Unit of recovery COMMIT LRSN (the same as #UORCOMMITPOINT if not data sharing)</td>
</tr>
<tr>
<td>#URIDCOMMITPOINT</td>
<td>CHAR(6)</td>
<td>Unit of recovery COMMIT (RBA)</td>
</tr>
<tr>
<td>#INSERTS</td>
<td>INTEGER</td>
<td>Number of INSERTs for the unit of recovery</td>
</tr>
<tr>
<td>#DELETES</td>
<td>INTEGER</td>
<td>Number of DELETEs for the unit of recovery</td>
</tr>
<tr>
<td>#UPDATES</td>
<td>INTEGER</td>
<td>Number of UPDATEs for the unit of recovery</td>
</tr>
<tr>
<td>#DELETERIS</td>
<td>INTEGER</td>
<td>Number of DELETEs due to referential integrity for the unit of recovery</td>
</tr>
<tr>
<td>#UPDATERIS</td>
<td>INTEGER</td>
<td>Number of UPDATEs due to referential integrity for the unit of recovery</td>
</tr>
</tbody>
</table>
LOGSCAN DDL file definition

Figure 26 shows the DDL file definition syntax of the LOGSCAN statement. This syntax specifies the type of data definition language (DDL) statements in (and the location of) the DDL file that Log Master creates using the log records selected in your log scan.

Figure 26  DDL file definition syntax diagram

Be aware of the following points regarding DDL output files:

- When you generate a DDL file, Log Master can generate a Catalog Activity report or update the Repository, but it cannot generate other forms of output in the same log scan (such as other reports or SQL). To generate other output in the same job, include an additional, separate log scan step.

- To generate DDL statements, Log Master must perform completion processing on DB2 catalog log records. Because of the large number of log records related to the catalog, this processing can cause your job to run longer and require more resources than a job that does not read the DB2 catalog. If you frequently operate on objects stored in the catalog, you can improve performance by creating more-frequent image copies of the catalog, or by defining tables in the DB2 catalog with Data Capture Changes (DCC). Table 3 on page 65 shows which tables should be defined with DCC or Data Capture None (DCN).
DB2 can implicitly create objects based on the DDL that you execute (for example, when you create a table without specifying an IN clause, DB2 can create a database and table space for you). When Log Master encounters log records that reflect the implicit creation of an object, Log Master includes comments within the generated output file that contain the corresponding CREATE statement.

Because of the way that DB2 logs the creation of an index, Log Master cannot distinguish an implicitly created index from an explicitly created one. Log Master includes a corresponding CREATE INDEX statement in the output file without comments.

When DB2 implicitly creates a database, it logs the implicit create action in a different unit of recovery from the original (explicit) create action. The URID of the implicit create action has different URID-related attributes (such as authorization ID and plan name) than the URID of the explicit create action. Be aware of this difference if you use URID-related criteria to select log records for an output DDL file.

For more information about Unicode characters in DDL output files, see “Considerations for output files and Unicode” on page 229.

**MIGRATE**

Directs Log Master to create DDL used for migration, replicating the DDL that was originally executed. For example, if the original DDL contains a CREATE TABLE statement, the MIGRATE DDL contains the same statement.

**UNDO**

Directs Log Master to create UNDO DDL, reversing the statement type of the DDL that was originally executed. For example, if the original DDL contains a DROP TABLE statement, the UNDO DDL contains a corresponding CREATE TABLE statement.

**EXECUTE**

Directs Log Master to execute the generated UNDO DDL immediately after it is generated, without any review or intervention. This keyword is most frequently used in test environments where any unintended consequences do not affect important production data.

Be aware of the following points as you generate UNDO DDL:

Because of DB2 constraints, Log Master cannot generate UNDO DDL to reverse the effects of an ALTER TABLE ADD COLUMN or an ALTER TABLE ALTER COLUMN statement. If Log Master encounters these statements as it generates UNDO DDL, it issues a warning message, ignores the ALTER statement, and continues generating other DDL statements.
If the original DDL includes a DROP statement, and you generate UNDO DDL, Log Master generates a CREATE statement to re-create the dropped object. However, Log Master does not generate statements to reinsert the data that was removed by the drop action. To recover the data, use the Log Master automated drop recovery feature (see “DROPRECOVERY statement” on page 308).

The DDL objects that Log Master writes in an output logical log file cannot be used to generate UNDO DDL, and they cannot be processed by the High-speed Apply Engine as UNDO. The DSNDDB06 insert, update, and delete records must be included in the logical log file to generate UNDO DDL.

When Log Master creates UNDO DDL, it attempts to reverse the effect and the order of the original DDL statements. In some cases, the dependencies or constraints on existing DB2 objects prevent the reversed DDL statements from executing correctly.

For example, consider the following actions:

— Create partitioned table space
— Create a distinct type
— Create table (in partitioned table space) using the distinct type

The generated UNDO DDL reverses the order of these actions. When you execute the DDL, the following actions occur:

— Drop table: DB2 requires that you drop a partitioned table space, not a partitioned table. Because of this requirement, Log Master inserts a comment in the generated DDL file to indicate that it cannot drop the partitioned table.

— Drop distinct type: This action fails because the type is still used in the table, which was not dropped by the preceding action.

— Drop table space: This action succeeds (dropping both the table and the table space), but the distinct type remains in existence.

Other similar situations result from DB2 dependencies and constraints. To avoid these situations, examine and edit your generated UNDO DDL statements before executing them.

**DATASET Output definition**

Specifies the characteristics of the output data set that contains the MIGRATE or UNDO DDL. For more information, see “LOGSCAN output definition” on page 260.

**RECFM**

Determines the record format of the output DDL file.
**FB**

Indicates that records are fixed length and blocked. This is the default value.

**VB**

Indicates that records are variable length and blocked. This value allows individual DDL statements to be written on a single record, regardless of their length. This type of file can be easier to process on other platforms.

When you specify this value, the following items apply:

— Log Master ignores the presence of the VERBOSE keyword.
— Dynamic DDL processors (SPUFI or DSNTEP2) might not process the file.
— Log Master sets the logical record length of the output file (LRECL) to the system-determined block size (SDB) of the storage device where the output file is written (minus 4 bytes).

**VERBOSE**

Directs Log Master to embed ANSI-standard comments in the generated DDL file. These comments describe unit of recovery identifier (URID) information and other information related to the database structures. The URID information can be useful. For example, you can use it to determine which user issued GRANT or REVOKE statements to change DB2 authorizations. This keyword is optional.

By generating a verbose DDL file with a filter that includes catalog activity or catalog objects, you can select information about changes to the DB2 catalog, including changes to DB2 security. The Catalog Activity report offers a more concise version of the same basic information. For more information, see “CATALOG ACTIVITY” on page 186.

**COMMIT FREQUENCY**

Specifies how frequently Log Master includes COMMIT statements in the output DDL file.

**NONE | 0**

Causes Log Master to commit all DDL statements at one time (at the end of processing). If you set this value to zero, Log Master interprets it as equivalent to the NONE keyword.

**nnnn**

Specifies the number of DDL statements between COMMIT statements.
The maximum limit for this value is 9999. The default value is 1, which causes Log Master to generate a COMMIT statement after each generated DDL statement.

To prevent errors as it generates UNDO DDL, Log Master includes COMMIT statements at different places than in the original DDL.

**NOTE**

Log Master enables you to express commit frequency for DDL generation differently than for SQL generation. Where you must express DDL frequency as a number of statements, you can express SQL frequency as a number of transactions or a number of statements. In this context, “transactions” are considered to be the same as URIDs.

If you specify INCLUDE DDL in the SQL type/output definition, Log Master treats the commit frequency value as a number of statements for DDL, and a number of transactions or statements for SQL, depending on what you specify.

**SQLID sqlid**

Directs Log Master to generate a SET CURRENT SQLID statement and insert it at the beginning of the generated DDL. When you execute the generated DDL, DB2 runs it under the authority granted to the authorization ID represented by sqlid.

Specify a valid authorization ID that has the appropriate authority in your environment.
Figure 27 shows the Output definition syntax of the LOGSCAN statement. Use this syntax to control the location and type of data sets that Log Master creates as output.

**dataSetName**

Specifies a data set where Log Master creates the output. The `dataSetName` follows operating system guidelines for data set names. Depending on the keyword that you use (SHR, MOD, OLD, or NEW), you can specify a partitioned data set, a sequential data set, or a member of a generation data group (GDG). If the `dataSetName` is eight bytes or less, Log Master treats the name as a DD name.

You can also use the normal product symbolic substitutions in data set names. For more information, see “Symbolic substitutions” on page 323.

**(-nn)**

Defines the data set as an existing entry in the GDG group specified by `dataSetName`. The `-nn` value refers to the entry using a relative numeric value. Refer to the most recent entry with the suffix of zero (0). Refer to the preceding generation with the suffix of (-1), and so on.

**( + nn )**

Defines the data set as a new entry to the GDG group specified by `dataSetName`. The `+nn` value refers to the entry using a relative numeric value. To create a new entry, use a suffix of +1.
(member)

Defines a member of a partitioned data set. Use this keyword only for SQL, DDL, logical log control, load control, and report data sets.

SHR

Defines the disposition of the data set. SHR indicates that dataSetName already exists and that it can be shared.

OLD

Defines the disposition of the data set. OLD indicates that dataSetName already exists, but it cannot be shared.

MOD

Defines the disposition of the data set. MOD indicates the following behavior:

- If dataSetName already exists, Log Master appends any output to the end of the data set.
- If dataSetName does not exist, Log Master creates a new sequential data set.

For more information about what Log Master does when the data set does not exist and you do not specify allocation parameters, see “Allocation parameters” in succeeding paragraphs.

In either case, access to the data set cannot be shared.

NEW

Directs Log Master to create a new sequential data set. You cannot specify a partitioned data set as NEW. For more information about what Log Master does if you do not specify allocation parameters for a new data set, see “Allocation parameters” in succeeding paragraphs.

Allocation parameters

If a data set is specified as NEW (or if it is specified as MOD, but the data set does not exist) you can specify parameters to allocate the data set. For more information about how to specify the parameters, see “Allocation parameters” on page 262.

BMC Software recommends that you specify allocation parameters for either a NEW data set or a MOD data set that does not exist. If you do not specify allocation parameters, Log Master creates a new data set, but the data set’s attributes depend on the storage management policies in your environment, and might not be optimal for the output that Log Master creates.
Allocation parameters

Figure 28 shows the Allocation parameters syntax used in various elements of the LOGSCAN statement. These parameters determine the keywords and values that Log Master uses when it allocates space for an output data set.

**Figure 28  Allocation parameters syntax diagram**

- **LRECL (nnnn)**
  Specifies the logical record length (in bytes) of records in an output data set. When Log Master writes
  - SQL output files or DDL output files to a fixed blocked (FB) file, it forces the LRECL value of the output data set to be 80 bytes
  - load control files, it forces the LRECL value of the output data set to be 80 bytes (and the RECFM value to be FB)
  - printed output reports to a fixed blocked (FB) file, it forces the LRECL value of the output data set to be a minimum of 81 bytes.
The required LRECL value depends on the output type. For reports that you generate using a template, you must define the LRECL of your output to be large enough to accommodate the longest line length defined by your template.

Table 21 shows the required and default values of this keyword, depending on the type of output that Log Master writes to the output data set.

<table>
<thead>
<tr>
<th>Type of output</th>
<th>Required RECFM value</th>
<th>Default LRECL value</th>
<th>Required LRECL value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports (standard default reports)</td>
<td>FBA</td>
<td>81</td>
<td>81 to 32760</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Catalog Activity report requires a value greater than 132)</td>
</tr>
<tr>
<td>Reports (standard default in CSV or SDF format)</td>
<td>VB</td>
<td>270</td>
<td>81 to 32760</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Catalog Activity report requires a value greater than 132)</td>
</tr>
<tr>
<td>Reports (generated from report template)</td>
<td>FBA</td>
<td>81</td>
<td>81 to 32760</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Define the LRECL of your output to be large enough to accommodate the longest line length defined by your template.)</td>
</tr>
<tr>
<td>SQL files</td>
<td>FB or VB</td>
<td>FB 80</td>
<td>FB 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VB</td>
<td>VB a value greater than or equal to the maximum length of the SQL statement plus 4</td>
</tr>
<tr>
<td>SQL template files</td>
<td>same as SQL file</td>
<td>same as corresponding SQL file</td>
<td></td>
</tr>
<tr>
<td>DDL files</td>
<td>FB or VB</td>
<td>FB 80</td>
<td>FB 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VB</td>
<td>VB a value greater than or equal to the maximum length of the DDL statement plus 4</td>
</tr>
<tr>
<td>Logical log data files</td>
<td>VB</td>
<td>SDB minus 4 bytes</td>
<td>Required: a value greater than or equal to the length of the logical log header</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Recommended: a value equal to the length of the logical log header plus the length of the table rows included in the output</td>
</tr>
<tr>
<td>Logical log control files</td>
<td>VB</td>
<td>SDB minus 4 bytes</td>
<td>Required: a value greater than or equal to the length of the longest logical log control record (see Chapter 4, &quot;Logical log files&quot;)</td>
</tr>
</tbody>
</table>
LIKE (modelDscb)

Specifies the allocation attributes of an output data set by copying the attributes of the model data set defined by modelDscb. Use an existing cataloged data set on a direct access volume as the model data set. This keyword is required if you are cataloging generation data group (GDG) data sets.

CYLINDERS | TRACKS

Specifies how the allocates space for an output data set. CYLINDERS indicates that Log Master uses DASD cylinders, TRACKS indication that Log Master uses DASD tracks.

SPACE

Determines how Log Master requests space for an output data set on a direct access volume.

priSpace

Specifies the number of units of storage (either cylinders or tracks) that Log Master requests when it creates the output data set.

secSpace

Specifies the number of units of storage that Log Master requests when it finds that either the primary allocation is full or a previous secondary allocation is full. If you specify secSpace, separate it from priSpace with a comma.
DIRBLKS

Defines the number of directory blocks that Log Master requests the operating system to allocate to an output data set. This keyword is valid only when you define an external file to hold data from a large volume column (such as an XML column or a LOB column) as part of an output load file, and only for a partitioned data set (PDS). For all other data sets, Log Master ignores this keyword. For more information, see “TEMPLATE” on page 245.

DSNTYPE

Defines the type of output data set that Log Master requests the operating system to allocate. The supported data set types for this context are PDS (partitioned data set) and PDSE (partitioned data set extended, sometimes referred to as a library). If you define a PDSE, you must use DFSMS parameters such as STORCLAS or DATAACLAS to define the remaining characteristics of the data set.

This keyword is valid only when you define a PDSE as an external file to hold LOB column values as part of an output load file. For all other data sets, Log Master ignores this keyword. For more information, see “TEMPLATE” on page 245.

UNIT (unitName)

Specifies the storage device (or group of devices) where Log Master requests space for an output data set. Enter the unit name of a device or group of devices. Do not use this keyword with DFSMS.

RELEASE

Directs Log Master to request that the operating system releases any unused space when it closes a given output data set.

MAXVOL (volumeCount)

Directs Log Master to limit the number of volumes that it uses to store output data sets. Enter a maximum number of volumes as volumeCount.

EXPDT (expDate)

Specifies the expiration date of an output data set. In the initial request for space, Log Master includes the date specified by expDate. On or after that date, the data set can be deleted or written over by another data set. You cannot specify both an expiration date and a retention period for the same data set.

Enter an expiration date using the format yyyy/ddd, where yyyy represents a four-digit year and ddd represents the number of days from the beginning of the year (the julian date). For example, you would enter January 30, 2013 as 2013/030.
Do not enter any two-digit year values, except for the dates 99365, 99366, 98000, and 99000 (these dates have special meanings for the operating system and certain tape management systems).

**RETPD (retPeriod)**

Specifies the retention period of an output data set. Enter a retention period using a number from 0 to 9999. In the initial request for space, Log Master includes the retention period. After the specified number of days have passed, the data set can be deleted or written over by another data set. You cannot specify both a retention period and an expiration date for the same data set.

**COMPRESS**

Indicates whether Log Master instructs the operating system to compress the data in an output data set.

**DATACLAS (dataClass)**

Specifies the data class of an output data set. Use this keyword only with DFSMS. A data class defines allocation attributes of a data set (such as organization or record length).

**MGMTCLAS (mgmtClass)**

Specifies the management class of an output data set. Use this keyword only with DFSMS. A management class defines how DFSMS manages a data set (when DFSMS migrates the data set, when it archives the data set, and so on).

**STORCLAS (storClass)**

Specifies the storage class of an output data set. Use this keyword only with DFSMS. A storage class defines storage attributes of a data set (similar to those defined with the UNIT or MAXVOL keywords).

**VOLUME**

Specifies the volume or volumes where Log Master stores output data sets. You can specify up to six VOLSERs (regardless of the value of the MAXVOL keyword). Separate multiple VOLSERs with commas. If you specify a magnetic tape volume, Log Master overwrites any existing information on the tape.
LOGSCAN column include/exclude definition

Figure 29 shows the Column include/exclude definition syntax of the LOGSCAN statement. This syntax enables you to explicitly control which table columns appear in the generated output. For example, you can create a report that does not print the information in a sensitive column, or you can generate MIGRATE SQL that duplicates only certain columns into another database.

Figure 29  Column include/exclude definition syntax diagram

Be aware of the following points regarding included or excluded columns:

- If you include columns in (or exclude columns from) a logical log file, the same columns are included in or excluded from all other forms of output that are defined for the log scan.

For example, if you request both logical log and SQL output files, and you exclude the ROWID column of a LOB column from the output logical log file, Log Master excludes the LOB column from the output SQL file as well. In this situation, Log Master cannot generate the SQL for the LOB column because the ROWID was removed from the logical log record that it was processing. This limitation also applies to the DOCID column of an XML column.

- If you exclude columns from an output SQL file, and the DB2 log for your selected objects contains an update action that affects only the excluded columns, Log Master
  — does not generate an UPDATE statement
  — increments the “updates eliminated” value in message BMC097226
  — includes an ANSI-standard comment in the SQL file to note the omitted update action
Do not confuse the Column include/exclude definition with a filter predicate that selects records based on a specific column. The differences are as follows:

— A Column include/exclude definition affects the entire column in the output (regardless of the value in the column). If you exclude a column, the output does not contain the specified column at all.

— A filter predicate based on a column value affects log records in the output (based on the column’s value in the log record). If a filter excludes records based on a column’s value, the output still contains the specified column, only certain log records are missing. For more information, see “Filter predicate” on page 279.

**INCLUDE | EXCLUDE**

Directs Log Master to generate output that includes or excludes certain columns. You cannot specify an include list and an exclude list for the same table.

**INCLUDE**

Directs Log Master to include only the column or columns specified by the COLUMNS keyword.

**EXCLUDE**

Directs Log Master to exclude the column or columns specified by the COLUMNS keyword. The output contains all of the original table columns except those in the exclude list.

**TABLE owner.tableName**

Specifies the table that contains the column or columns specified by the COLUMNS keyword. All of the columns in a single column list must be part of the same table. To specify columns from more than one table, you must specify multiple TABLE keywords.

To fully qualify a table, enter the owner of the table. If you do not specify an owner ID, Log Master uses a default value. The default value is the user ID of the batch job that runs Log Master. Separate the owner ID from the table name with a period.

**COLUMNS (columnName)**

Specifies the column or columns in a given table that Log Master includes in (or excludes from) the generated output. You can specify one or more columns in the table, separated by commas.
**LOGSCAN scan range definition**

Figure 30 shows the Scan range definition syntax of the LOGSCAN statement. This syntax determines the part of the DB2 log that Log Master scans for log records. It also enables you to specify the start point for REDO information (often called a recovery point).

If you specify the REPOS DELETE keyword of the LOGSCAN statement, the Scan range definition is not required.

**Figure 30  Scan range definition syntax diagram**

**FROM Range definition TO Range definition**

Specifies the portion of the DB2 log that Log Master scans (also called the time frame). FROM indicates the start point and TO indicates the end point of the log scan. Each Range definition defines one point in the DB2 log. For more information, see “Range definition” on page 272.
OR LIMIT

Defines an alternate limit on the range of the log scan. When you specify this keyword, the end point of your log scan is the point defined by the TO keyword in the Range Definition or the point defined by OR LIMIT, whichever comes first. The default value of this keyword does not define an alternate limit.

This keyword is frequently used with ongoing jobs. For example, assume that during a period of high logging activity, an ongoing job does not complete during its assigned processing window. The operating system terminates the job. No valid output is produced. If you run the job again, the same problem exists. In this situation, define an alternate limit that includes a range of log short enough to be processed by Log Master during the window. You can then run the same job repeatedly (producing valid output for each run without changing SYSIN syntax for each run), until you work your way through the period of high activity and “catch up” to normal operation.

**NOTE**

The limit defined by this keyword does not define how long Log Master executes at runtime; it defines only how much log Log Master scans. For example, if you specify OR LIMIT 01:00:00, Log Master does not stop running after 1 hour, it stops after it has scanned all log files covering a 1-hour period from your start point. The same principle applies to a limit specified as a number of log files.

**nnn LOG FILES**

Defines the alternate limit as a number of log files. Log Master determines which log file contains the start point specified by the FROM keyword of the Range Definition, adds the specified number of log files, and sets the alternate limit equal to the end point of the selected log file. In a data sharing environment, Log Master performs the same process to determine the end points of the selected log files on each member, and then sets the alternate limit equal to the earliest end point.

**nnn DAYS**

Defines the alternate limit as a number of days. Specify this keyword either by itself or with a \texttt{hh:mm:ss} value that is added to the number of days. Log Master determines the date and time of the start point specified in the Range Definition, adds the specified number of days, and sets the alternate limit equal to the computed date and time value.
Log Master computes the date and time based on your specified start point. When all of the following conditions exist, Log Master adjusts the start point before it computes the alternate limit:

- Log Master runs in a non-data-sharing environment.
- The start point in the Range Definition is not specified as a date and time value.
- The current log scan is either a non-ongoing log scan or the initial run of an ongoing log scan.

In this situation, Log Master sets the start point equal to the starting timestamp value of the log file that contains your specified start point.

\textit{hh:mm:ss}

Defines the alternate limit as a number of hours, minutes, or seconds. Specify this value either after the DAYS keyword, or by itself. Log Master determines the date and time of the start point specified in the Range Definition, adds the specified period of time, and sets the alternate limit equal to the computed date and time value. For more information about a situation where Log Master adjusts your start point, see page 270.

To specify a period less than one hour, specify the hours value as 00. To specify a period of seconds, specify both the hours and minutes value as 00.

Be aware of the following points when specifying an alternate limit:

- If you specify the RERUN RUNSEQ keyword to execute a previous log scan again, Log Master ignores the specified OR LIMIT when processing a rerun. If you specify USE RUNSEQ to use the start point of a previous log scan, be sure that the end point defined by OR LIMIT is after the end point of the previous log scan.

- The implementation of OR LIMIT depends on Repository changes introduced in Log Master version 7.3.00. Do not specify OR LIMIT for the first execution of a migrated work ID after you install a later version of Log Master than 7.3.00 if that work ID was migrated from an earlier version than 7.3.00, and it contains an ongoing log scan. Subsequent executions can support an OR LIMIT specification.

\textbf{REDO FROM}

Determines the start point that Log Master uses to generate REDO information. This point is also referred to as the recovery point or point of recovery. Log Master can write REDO information to a logical log file, an SQL output file (in the form of REDO SQL), or a report (in one of three report formats).

Use REDO information to re-apply a set of transactions \textit{after a recovery action}, while \textit{omitting} a set of “problem” transactions defined by your Range definition and WHERE clause. For more information, see “REDO” on page 215.
To generate REDO information, define a recovery point and a WHERE clause that refers to at least one specific DB2 object (such as a table space, table, or column). BMC Software strongly recommends that you generate REDO SQL before you recover any table spaces. You might be able to generate the SQL statements after the recovery by using the PROCESS PITS keyword of the OPTION statement, but BMC Software does not recommend doing so. For more information, see “PROCESS PITS” on page 128.

LASTQUIESCE

Directs Log Master to generate REDO information starting with the most recent quiesce point available in the SYSIBM.SYSCOPY table of the DB2 Catalog. The quiesce point must include all table spaces explicitly or implicitly specified in the WHERE clause or filter. You cannot specify this type of recovery point if the value of the FILTER METHOD keyword (or the FLTRMTHD installation option) is DYNAMIC.

RBA | LRSN X 'byteString'

Directs Log Master to generate REDO information starting with a specific RBA/LRSN. The byte string can be up to 12 characters long. Use the following format: RBA X'nnnnnnnnnnnn'.

MARK logMarkName

Directs Log Master to generate REDO information starting at the log mark represented by logMarkName. A log mark is a symbolic name that represents a given point in the log. Remember that a log mark name created within a given job step cannot be referenced within that job step.

For more information about log marks and log mark names, see “MARK logMarkName” on page 274.

Range definition

Figure 31 on page 273 shows the Range definition syntax that occurs in both the LOGSCAN and DROPRECOVERY statements. This syntax enables you to define a start point or an end point in the DB2 log. You can define the point as a relative byte address (RBA) value, a log record sequence number (LRSN) value, an existing log mark, or a date and time value.
Range definition syntax diagram

**RBA | LRSN X 'byteString'**

Describes a start or end point as a specific RBA or LRSN value. Enter a 1 to 12-character hexadecimal value in the following format: RBA X'nnnnnnnnnnnn'. Enter LRSN values in a data sharing environment or RBA values in a non-data-sharing environment.

Example:

```
FROM RBA X'0005A4693F4'
TO RBA X'0005A488902'
```
MARK logMarkName

Describes a start or end point as a specific log mark. A log mark is a symbolic name that represents a given point in the log. A log mark name:

- can be up to 40 characters long
- can be any combination of alphanumeric or national characters
- can use periods to separate portions of the name
- is not qualified by Log Master with a user ID or a TSO prefix
- can use symbolic substitutions (for more information, see “Symbolic substitutions” on page 323)

Within the Range definition, use existing (not newly created) log marks. Remember that a log mark created within a job step (one run of Log Master) cannot be referenced within that step.

You can create a log mark by using the LOGMARK statement or the MARKSCAN keyword of the LOGSCAN statement. For more information see “LOGMARK statement” on page 305 or “MARKSCAN” on page 173. Log Master maintains log mark information in the Repository (in a table named ALPMARK).

You can reference log marks using a numeric value appended to the log mark name. This technique enables you to reuse the same log mark name and refer to specific log points by the numeric suffix. The suffix can be a relative or absolute generation.

(relativeGeneration)

Refers to a specific log point by using a relative value for the numeric suffix. Refer to the most recent log mark with the generation suffix of zero (0). Refer to the previous generation of the log mark with the generation suffix of (-1), and so on. When you work with ongoing processing, it can be convenient to use relative generations with your log mark names.

Example:

```
FROM MARK MSTRUPDATESTART(-1)
TO MARK MSTRUPDATESTART(0)
```

(absoluteGeneration)

Refers to a specific log point by using the absolute value of the numeric suffix. Separate the suffix from the log mark name with a period.
Range definition

Example:

| FROM MARK MSTRUPDTESTART.12 |
| TO MARK MSTRUPDTESTART(0) |

**CURRENT**

Use this keyword only in the TO portion of the Scan range definition. This keyword defines the end point of the log scan as the most recent RBA/LRSN value that DB2 has written to the log when your Log Master job begins executing.

If the current job or job step includes the MARKSCAN keyword of the LOGSCAN statement to establish a new log mark, the new log mark will correspond to the RBA/LRSN selected for CURRENT.

Example:

| FROM MARK BENCHMARK(0) |
| TO CURRENT |

**LAST ARCHIVE**

Use this keyword only in the TO portion of the Scan range definition. This keyword defines the end point of the log scan as the RBA/LRSN value of the end point of the most recently-written archive log file (when your Log Master job begins executing). Use this keyword to reduce the chance of reading active log files on the current DB2 subsystem.

In a data sharing environment, Log Master determines the end points of the most recent archive log files of all members and then uses the *earliest* of those end points. If the current job or job step uses the MARKSCAN keyword of the LOGSCAN statement to establish a new log mark, the new log mark will correspond to the RBA/LRSN selected for LAST ARCHIVE.

Example:

| FROM MARK BENCHMARK(0) |
| TO LAST ARCHIVE |
**DATE (mm/dd/yyyy) / TIME (hh:mm:ss)**

Specify a start point or end point according to date and time.

Example:

```
FROM DATE(12/18/2012) TIME(08:15:30.000000)
TO DATE(12/20/2012) TIME(12:15:30.000000)
```

Use any of the available date and time formats in the Range definition, but remember that to control the date and time displayed in reports, you must use the OPTION statement (for more information, see “DATEFMT” on page 121). Log Master supports the following date and time formats:

- MM/DD/YYYY/HH:MM:SS.nnnnnn (USA)
- DD.MM.YYYY.HH.MM.SS.nnnnnn (EUR)
- YYYY-MM-DD-HH.MM.SS.nnnnnn (ISO)
- YYYY-MM-DD-HH:MM:SS.nnnnnn (JIS)

The default format is ISO.

***(Current Date)***

Specifies the current date when the log scan step is executed. If you do not specify a TIME value, Log Master uses a default value of the current date and time. Because of this behavior, you cannot enter DATE(*) without a time value in the FROM portion of the Scan range definition.

Example:

```
TO DATE(*)
```

**-nn (relativeDate)**

Specifies a relative date value. When *nn* is preceded by a minus sign (-), Log Master subtracts the specified number of days from the current date.

Example:

```
FROM DATE(-2) TIME(00:00:00)
TO DATE(-1) TIME(00:00:00)
```

This example scans a range from midnight two days ago until midnight one day ago.
**- hh:mm:ss (relativeTime)**

Specifies a relative time value. When a time value is preceded by a minus sign (-), Log Master subtracts the number of hours, minutes and so on from the current time.

Example:

```
FROM DATE(*) TIME (-12:00:00) TO CURRENT
```

This example scans a range from 12 hours before run time up until the current date and time.

**TODAY**

Specifies the current date value when the log scan step runs.

Example:

```
FROM DATE(2012-12-18) TIME (00:01:00)
TO DATE (TODAY) TIME (00:01:00)
```

**YESTERDAY**

Specifies a date value one day before the current date value when the log scan step runs.

Example:

```
FROM DATE(-30) TIME (00:01:00)
TO DATE (YESTERDAY) TIME (00:01:00)
```

---

**LOGSCAN search condition definition**

Figure 32 on page 278 shows the Search condition definition syntax used in the WHERE clause of the LOGSCAN statement. Use this syntax to define the data that Log Master includes in the output of your LOGSCAN statement. A search condition can contain a simple filter predicate, multiple filter predicates, or a combination of filter predicates and additional search conditions. Use standard SQL syntax to combine multiple filter predicates and search conditions.
Filter predicate

A simple filter predicate defines one condition that will control log scan output. For more information, see “Filter predicate” on page 279.

Search condition definition

Search conditions can include additional, nested search conditions. All nested search conditions must conform to the syntax rules defined in this section.

AND | OR | NOT

Use the AND keyword, the OR keyword, or the NOT keyword to join multiple search conditions or filter predicates. These keywords work the same as logical operators in standard SQL syntax.
Figure 33 shows the filter predicate syntax used in the WHERE clause of the LOGSCAN statement. A filter predicate defines one condition that Log Master uses to select log records. You can combine a filter predicate with additional filter predicates to form a Search condition definition. For more information, see “LOGSCAN search condition definition” on page 277.

Figure 33  Filter predicate syntax diagram

1 See page 281 for limits on the use of the PART keyword.
Filter predicate

Selectable field definition

Specifies the field of a log record that Log Master compares against the `value` variable to select log records during the log scan. For more information about the fields that you can select, see “Selectable field definition” on page 285. For more information about abbreviating items, see “Synonyms” on page 298.

operator

Defines the operator that Log Master uses to compare a selectable field against a specified `value`. Match the type of the operator to the type of the field that you select. The available operators are as follows:

<table>
<thead>
<tr>
<th>Operator symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td>equal</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>not equal</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>not equal</td>
</tr>
<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>not greater than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>not less than</td>
</tr>
<tr>
<td><code>*&lt;</code></td>
<td>not less than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal to</td>
</tr>
</tbody>
</table>

value

For each log record, Log Master compares the value of the selectable field against the constant entered as `value` to determine whether to select the log record. For more information about the selectable field, see “Selectable field definition” on page 285. For `value`, you can enter:

- character constants (EBCDIC or Unicode)
  (for Unicode, use hexadecimal representation of UTF-8 values)
- graphic (double-byte) constants (EBCDIC)
- decimal numeric constants
- hexadecimal numeric constants
For each filter predicate, match the type of the constant in value to the type of the selectable field. As you enter constant values, be aware of the following points:

- Enter EBCDIC character constants and decimal numeric constants using standard EBCDIC characters. If required, Log Master supports delimited DB2 object names. For more information about delimited names, see the chapter on defining the log scan step in the Log Master for DB2 User Guide.

- Enter hexadecimal numeric constants as you do in SQL statements (for example, X'D3D6C1C4C7C5D5').

- Enter graphic constants (double-byte character set) as you do in SQL statements.

- For Unicode character constants (such as DB2 object names or column values)
  - If the constant contains only translatable characters, enter it using standard EBCDIC characters.
  - If the constant contains any untranslatable characters, use hexadecimal representation of the Unicode UTF-8 characters for the whole constant (for example, for a Unicode string of 'A B C', enter X'4120422043').

Log Master distinguishes between hexadecimal numeric constants and hexadecimal representation of Unicode characters based on the data type of the selectable field.

PART nnnn

Selects log records that reflect activity in a given partition of a table space. The nnnn represents the partition number within the table space. To use this keyword, the following conditions must be true:

- The value of the Selectable field definition must be either TABLESPACE NAME or PSID. For more information about the selectable field, see “Selectable field definition” on page 285.

- The operator must be either equal or not equal (=, <>, ¬=, or *=).

For example, to select only the log records in one partition, use a search condition such as TABLESPACE NAME = ACCOUNTDB.TS01 PART 1. To select all of the log records in a table space except for the third partition, use a condition like PSID = 505.007 AND PSID <> 505.007 PART 3.

You can also use the PART keyword as you define a set of multiple table spaces (using the IN or NOT IN keywords).
LIKE stringPattern

Log Master can compare a string item (for example, a table name or the content of a column) against a string pattern to determine whether to select log records during the log scan. This operation is similar to the LIKE predicate in SQL syntax. If the string item meets the requirements specified by the string pattern, Log Master selects the log record.

To specify a string pattern for selecting log records, use the wildcard characters listed in Table 23. The characters can be included at the beginning of, in the middle of, or at the end of the string pattern. You can include more than one occurrence or more than one type of wildcard character in the string pattern.

Table 23  Wildcard characters for comparisons

<table>
<thead>
<tr>
<th>Character</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>any string of zero or more characters (Unicode UTF-8 value: x'2A')</td>
</tr>
<tr>
<td>?</td>
<td>any single character (Unicode UTF-8 value: x'3F')</td>
</tr>
<tr>
<td>%</td>
<td>any string of zero or more characters (Unicode UTF-8 value: x'25')</td>
</tr>
<tr>
<td>_</td>
<td>any single character (Unicode UTF-8 value: x'5F')</td>
</tr>
</tbody>
</table>

Exercise caution using this wildcard character because the underscore (_) is valid in many DB2 contexts, such as a table name.

For graphic or vargraphic data: Using GX’...’ or UX’...', the following wildcard characters can be embedded in the string patterns:

<table>
<thead>
<tr>
<th>Character</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>any string of zero or more characters (EBCDIC X'425C')</td>
</tr>
<tr>
<td>*</td>
<td>any string of zero or more characters (Unicode X'002A')</td>
</tr>
<tr>
<td>%</td>
<td>any string of zero or more characters (EBCDIC X'426C')</td>
</tr>
<tr>
<td>%</td>
<td>any string of zero or more characters (Unicode X'0025')</td>
</tr>
<tr>
<td>?</td>
<td>any single character (EBCDIC X'426F')</td>
</tr>
<tr>
<td>?</td>
<td>any single character (Unicode X'003F')</td>
</tr>
<tr>
<td>–</td>
<td>any single character (EBCDIC X'426D')</td>
</tr>
<tr>
<td>–</td>
<td>any single character (Unicode X'005F')</td>
</tr>
</tbody>
</table>
The following examples illustrate some common uses of wildcard characters in a string pattern:

<table>
<thead>
<tr>
<th>String pattern</th>
<th>Selects log records if a string</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIKE 'ACCOUNT*'</td>
<td>begins with the characters ACCOUNT</td>
</tr>
<tr>
<td>LIKE &quot;%MANAGER&quot;</td>
<td>ends with the characters MANAGER</td>
</tr>
<tr>
<td>LIKE '<em>X</em>'</td>
<td>includes the character X in any position within the string</td>
</tr>
<tr>
<td>LIKE '25C39C25'</td>
<td>includes the Unicode character X'C39C' in any position within the string</td>
</tr>
<tr>
<td>LIKE &quot;<em>ITEM_</em>&quot;</td>
<td>ends with the characters ITEM, followed by any single character</td>
</tr>
<tr>
<td>LIKE 'NUM=???NAME=A*'</td>
<td>begins with the characters NUM=, followed by any three characters, followed by the characters NAME=A, followed by any number of characters</td>
</tr>
</tbody>
</table>

**IN (value, value)**

Log Master can compare data from log records against a single value or a set of values separated by commas. The IN keyword indicates a set of multiple values.

**NOT**

Specifies that log records are selected when the value of a field does not meet the requirements that you specify. This keyword can select log records when

- a column value does not match a string pattern value
- a column value does not match any member of a set of multiple values
- a column value is not changed
- a column value is not null
- a record’s update type does not match any member of a set of multiple update types

**owner.tableName.columnName CHANGED | NULL**

Selects log records based on the content of a specific column in a specific table. The CHANGED keyword enables you to select log records if the content of the column is changed in any way. The NULL keyword enables you to select log records if the content of the column is a null value.

Log Master can also compare the value in a specific table and column against a constant value or a set of constant values. For more information, see “owner.tableName.columnName” on page 290.
To fully qualify a specific column in a specific table, enter the user ID of the owner of the table. If you do not specify a value, Log Master uses a default value. The default value of **owner** is the user ID of the batch job that runs Log Master.

Do not confuse this type of filter predicate with the Column include/exclude definition that selects table columns in certain types of output. The differences are as follows:

- A filter predicate based on a column’s status affects **log records** in the output. If a filter excludes records based on a column’s status, the output still contains the specified column, only certain log records are missing.

- A Column include/exclude definition affects **the entire column** in the output (regardless of the column’s status). If a column is excluded, the output does not contain the specified column at all. For more information, see “**LOGSCAN column include/exclude definition**” on page 267.

**UPDATE TYPE**

Selects log records based on what type of action was performed on a table row (the update type). The valid values are **INSERT**, UPDATE, DELETE, and EXCHANGE.

**IN (INSERT | UPDATE | DELETE | EXCHANGE)**

Log Master can compare the update type of log records against a single update type or a set of update types separated by commas. The IN keyword indicates a set of multiple update types. The valid update types are **INSERT**, UPDATE, DELETE, and EXCHANGE.

**Catalog activity definition**

Selects log records based on the type of DB2 catalog activity that they reflect (what type of data definition language (DDL) statement resulted in the log record). Use the Catalog activity definition when you are generating a DDL output file; it is not valid if your job specifies any other type of output. For more information about the types of DB2 catalog activity that you can select, see “**Catalog activity definition**” on page 292.

**Catalog object definition**

Selects log records based on the type of DB2 catalog object that the log record relates to (for example, records relating to indexes or records relating to views). Use the Catalog object definition when you are generating a DDL output file; it is not valid if your job specifies any other type of output. For more information about the types of DB2 catalog objects that you can select, see “**Catalog object definition**” on page 295.
**DB2 command definition**

Selects log records based on the DB2 command that the log record relates to (for example, records relating to a TRACE). The DB2 Command definition is not valid if your job specifies any other type of output. For more information about the DB2 commands that you can select, see “DB2 command definition” on page 299.

**Selectable field definition**

Figure 34 on page 286 shows the Selectable field definition syntax used in the WHERE clause of the LOGSCAN statement. The Selectable field definition is one part of the Search condition definition syntax used to select log records during a log scan. Other parts of the syntax enable you to compare one item or “field” within a log record against a constant value. The Selectable field definition enables you to specify which field Log Master compares against the constant value.
You can use synonyms to specify the selectable field of a filter predicate. For more information about synonyms, see “Synonyms” on page 298.
UNIT OF RECOVERY

Select log records based on a unit of recovery. A unit of recovery is a recoverable sequence of operations within DB2, identified by a 12-character hexadecimal unit of recovery identifier (URID). The identifier represents the log address of the first log record in the unit of recovery.

- In a non-data-sharing environment, specify a relative byte address (RBA) value.
- In a data sharing environment, specify a log record sequence number (LRSN) value.

BMC Software strongly recommends that you specify LRSN values in a data sharing environment. If you need to use RBA values (for example, when you have RBA values from DSN1LOGP output), you can specify an RBA value as long as your filter operator is equal (=), not equal (<>), IN, or NOT IN. For all other operators, specify an LRSN value.

If you use unit of recovery-related criteria (for example, URID, authorization ID, plan name, or correlation ID) to select log records for an output DDL file, be aware of whether your selected objects were created explicitly or implicitly. For more information, see page 256.

CORRELATION ID

Selects log records based on a correlation ID. A correlation ID is a unique string identifier for a specific DB2 thread.

PLAN NAME

Selects log records based on a plan name. A plan name is a character string representing an application plan, the control structure used by DB2 to process SQL statements.

AUTH ID

Selects log records based on an authorization ID. An authorization ID is a character string used to connect to DB2 (it frequently represents an individual, a work group, or a function).

CONNECTION ID

Selects log records based on a connection ID. A connection ID is a string identifier associated with a specific address space connection.
**CONNECTION TYPE**

Selects log records based on a connection type. A connection type is a string constant representing the manner in which a unit of recovery thread is attached to DB2. The following paragraphs contain a partial list:

- BATCH
- DIST (for DDF: Distributed Data Facility)
- DB2CALL
- MASS (for IMS)
- RRSAF (Recoverable Resource Manager Services Attachment Facility)
- SASS (for CICS)
- SYSTEM (for Real Time Statistics)
- TSO
- UTILITY

**DATABASE NAME**

Selects log records in a specific database.

**TABLESPACE NAME**

Selects log records in a specific table space.

**TABLE NAME**

Selects log records in specific table.

**DBID**

Selects log records based on a database ID. A database ID is the internal identifier that DB2 assigns to a database. Enter the DBID as a decimal number (for example, 505) or a hexadecimal number (for example, x'01F9').

**PSID**

Selects log records based on a page set ID. A page set ID is the internal page set identifier that DB2 assigns to a given table space. Enter the PSID and the accompanying DBID as decimal numbers (for example 505.028) or as hexadecimal numbers (for example, x'01F9'.x'001C').
OBID

Selects log records based on an object ID. An object ID is the internal identifier that DB2 assigns to an object. Enter the OBID and the accompanying DBID as decimal numbers (for example, 505.007) or as hexadecimal numbers (for example, x'01F9'.x'0007').

MEMBER NAME

Selects log records based on the name of a given member of a data sharing group. Select this field only in a data sharing environment.

MEMBER ID

Selects log records based on the member ID of a given member of a data sharing group. Select this field only in a data sharing environment. Enter the member ID as a decimal number (for example, 14) or a hexadecimal number (for example, x'0D').

Filtering on a member ID or a member name determines what log records Log Master selects; a filter does not influence whether Log Master reads log files from other members in a data sharing group. To influence which member's log files Log Master reads, use the USELGRNG keyword of the OPTION statement (“USELGRNG” on page 127).

SUBSYSTEM NAME

Selects log records based on the subsystem identifier (SSID) of a given DB2 subsystem. Select this field only in a data sharing environment. (Log Master also accepts the synonym SSID, see “Synonyms” on page 298).

LUW NETWORK ID

Selects log records based on the logical unit of work (LUW) network ID. An LUW identifies a thread within a network and is used to correlate local and remote activity for a single distributed transaction. The LUW network ID is an eight byte character field.

LUW NAME

Selects log records based on the logical unit of work (LUW) name. The LUW name is an eight byte character field.

LUW UNIQUE VALUE

Selects log records based on the logical unit of work (LUW) unique value. The LUW six byte numeric field that can contain an LRSN value.
ALIAS

Selects log records based on an alias that represents a specific table. For information about this keyword and the input source for your log scan, see the “SYNONYM” keyword.

SYNONYM

Selects log records based on a synonym that represents a specific table.

- If your input source is a logical log file, an option to include synonym and alias information must have been specified when the logical log file was created. For more information, see “INCLUDE SYNONYMS” on page 203.

- If your input source is the bootstrap data set (INPUT BSDS) or specific DB2 log files (INPUT DB2LOG), specifying this keyword can slow performance. For information on reducing the performance impact, see “INCLUDE SYNONYMS” on page 203.

OBJECT SET

Selects log records based on the table space names in a RECOVERY MANAGER for DB2 object set.

COMMAND

Selects log records based on a DB2 command, such as START TRACE.

owner.tableName.columnName

Selects log records based on the value of a specific column in a specific table. Log Master can compare the value of a column against a constant value or a set of constant values.

To fully qualify a specific column in a specific table, enter the user ID of the owner of the table. If you do not specify a value, Log Master uses the user ID of the batch job that runs Log Master.

Log Master can also select records based on whether the value in a specific table column has changed or is null. (For more information, see the material provided in “owner.tableName.columnName  CHANGED | NULL” on page 283.)
Do not confuse this type of filter predicate with the Column include/exclude definition that selects table columns in certain types of output. The differences are as follows:

- A filter predicate based on a column value affects log records in the output (based on the column’s value). If a filter excludes records based on a column’s value, the output still contains the specified column, only certain log records are missing.

- A Column include/exclude definition affects the entire column in the output (regardless of the value in the column). If a column is excluded, the output does not contain the specified column at all. For more information, see “LOGSCAN column include/exclude definition” on page 267.

**BEFORE | AFTER**

Selects update log records based on the value of a specific column in your specified table either before or after an update occurs. This keyword directs Log Master to examine either the before row image or the after row image contained in update log records.

**BEFORE**

Select update log records based on the content of a column’s before image.

**AFTER**

Select update log records based on the content of a column’s after image.

If you do not specify a value, Log Master selects log records when the content of either the column’s before image or the column’s after image satisfies the conditions of your WHERE clause or filter.

This keyword does not change how Log Master selects insert or delete log records; Log Master selects them based solely on the column’s value. For example, in a WHERE clause or filter you request Log Master to select log records if the column QUANTITY contains a value greater than 500, using the BEFORE keyword. Log Master selects

- all insert log records where QUANTITY is greater than 500
- all delete log records where QUANTITY is greater than 500
- update log records where QUANTITY is greater than 500 before the update was applied

To select update records before a change was applied along with delete log records, use the following WHERE clause syntax:
To select update records after a change was applied along with insert log records, use the following WHERE clause syntax:

```
WHERE
  (USERNAME.SOMETABLE.QUANTITY BEFORE = 500
   AND UPDATE TYPE IN(DELETE,UPDATE) )
```

```
WHERE
  (USERNAME.SOMETABLE.QUANTITY AFTER = 500
   AND UPDATE TYPE IN(INSERT,UPDATE) )
```

**Catalog activity definition**

Figure 35 on page 293 shows the catalog activity definition syntax used in the WHERE clause of the LOGSCAN statement. Use this syntax to select log records based on the type of DB2 catalog activity they reflect. You can also think of this syntax as selecting a type of data definition language (DDL) statement. For example, you can select all log records that resulted from DROP statements, or all log records that record privilege distribution (GRANT statements).

Use the catalog activity definition when you are generating a DDL output file or Catalog Activity report; it is not valid if your log scan specifies any other type of output. By defining a filter that includes catalog activity and generating a Catalog Activity report, you can select information about changes to the DB2 catalog, including changes to DB2 security.
To select log records based on a type of DB2 catalog activity, Log Master must perform completion processing on DB2 catalog log records. Because of the large number of log records related to the catalog, this processing can cause your job to run longer and require more resources than a job that does not read the DB2 catalog. If you frequently operate on objects that are stored in the catalog, you can improve performance by creating more frequent image copies of the catalog or by defining tables in the DB2 catalog with Data Capture Changes (DCC). Table 3 on page 65 shows which tables should be defined with DCC or Data Capture None (DCN).

**ALTER**

Selects log records based on whether they resulted from an ALTER statement.

**COMMENT ON**

Selects log records based on whether they resulted from a COMMENT ON statement.
**CREATE**

Selects log records based on whether they resulted from a CREATE statement.

**DROP**

Selects log records based on whether they resulted from a DROP statement.

**GRANT**

Selects log records based on whether they resulted from a GRANT statement.

**RENAME**

Selects log records based on whether they resulted from a RENAME statement.

**LABEL ON**

Selects log records based on whether they resulted from a LABEL ON statement.

**REVOKE**

Selects log records based on whether they resulted from a REVOKE statement.

**IN**

Log Master can select log records based on whether they resulted from a single type of DDL statement or a set of DDL statement types. The **IN** keyword indicates a set of multiple DDL statement types. Enclose the set of DDL statement types within parentheses and separate multiple types with commas. For example:

```
IN (DROP, REVOKE)
```

**NOT**

Specifies that log records are selected when the type of DDL statement reflected in them does not match any member of a set of multiple DDL statement types. For example:

```
NOT IN (CREATE, GRANT)
```
Catalog object definition

Figure 36 shows the Catalog object definition syntax used in the WHERE clause of the LOGSCAN statement. Use this syntax to select log records based on the type of DB2 catalog object that is affected by the activity reflected in the log record.

For example, you can select all log records that affect storage groups or all log records that affect indexes. Besides the commonly used objects, such as tables and columns, you can also select log records that affect other objects defined in the DB2 catalog, such as plans, packages, collections, and buffer pools.

Use the catalog object definition when you are generating a data definition language (DDL) output file or Catalog Activity report; it is not valid if your log scan specifies any other type of output. By defining a filter that includes catalog activity and generating a Catalog Activity report, you can select information about changes to the DB2 catalog, including changes to DB2 security.

Figure 36  Catalog object definition syntax diagram

To select log records based on a type of DB2 catalog object, Log Master must perform completion processing on DB2 catalog log records. Because of the large number of log records related to the catalog, this processing can cause your job to run longer and require more resources than a job that does not read the DB2 catalog. If you frequently operate on objects stored in the catalog, you can improve performance by creating more frequent image copies of the catalog, or by defining tables in the DB2 catalog with Data Capture Changes (DCC). Table 3 on page 65 shows which tables should be defined with DCC or Data Capture None (DCN).

**catalogObject**

Specifies the type of object in the DB2 catalog. The available objects are as follows:

**ALIAS**

Selects log records based on whether they reflect changes that affect aliases.


**BUFFERPOOL**

Selects log records based on whether they reflect changes that affect buffer pools.

**COLLECTION**

Selects log records based on whether they reflect changes that affect collections.

**COLUMN**

Selects log records based on whether they reflect changes that affect columns.

**CONTEXT**

Selects log records based on whether they reflect changes that affect contexts.

**DATABASE**

Selects log records based on whether they reflect changes that affect databases.

**DISTINCT TYPE**

Selects log records based on whether they reflect changes that affect distinct types.

**FUNCTION**

Selects log records based on whether they reflect changes that affect functions.

**GLOBAL TABLE**

Selects log records based on whether they reflect changes that affect global tables.

**INDEX**

Selects log records based on whether they reflect changes that affect indexes.

**MASK**

Selects log records based on whether they reflect changes that affect column masks.

**PACKAGE**

Selects log records based on whether they reflect changes that affect packages.
PERMISSION
Selects log records based on whether they reflect changes that affect row permissions.

PLAN
Selects log records based on whether they reflect changes that affect application plans.

PROCEDURE
Selects log records based on whether they reflect changes that affect procedures.

ROLE
Selects log records based on whether they reflect changes that affect roles.

SCHEMA
Selects log records based on whether they reflect changes that affect schemas.

SEQUENCE
Selects log records based on whether they reflect changes that affect sequences.

STORAGE GROUP
Selects log records based on whether they reflect changes that affect storage groups.

SYNONYM
Selects log records based on whether they reflect changes that affect synonyms.

SYSPRIV
Selects log records based on whether they reflect changes in system privileges.

TABLE
Selects log records based on whether they reflect changes that affect tables.

TABLESPACE
Selects log records based on whether they reflect changes that affect table spaces.

TRIGGER
Selects log records based on whether they reflect changes that affect triggers.
VIEW

Selects log records based on whether they reflect changes that affect views.

IN (catalogObject)

Log Master can select log records based on whether they reflect changes that affect a single type of DB2 object or a set of DB2 objects. The IN keyword indicates a set of objects. Enclose the set of DB2 objects within parentheses, and separate multiple objects with commas.

NOT

Specifies that log records are selected when the DB2 object that they affect does not match any member of a set of DB2 objects.

Synonyms

You can use the synonyms contained in the following table in the syntax of a filter predicate or the ORDER BY definition. For more information, see “Filter predicate” on page 279 or “ORDER BY definition” on page 194.

Table 24 Synonyms (part 1 of 2)

<table>
<thead>
<tr>
<th>Syntax term</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTH ID</td>
<td>AUTH_ID, AUTHID</td>
</tr>
<tr>
<td>CONNECTION ID</td>
<td>CONNECTION_ID, CONNID</td>
</tr>
<tr>
<td>CONNECTION TYPE</td>
<td>CONNECTION_TYPE, CONNTYPE</td>
</tr>
<tr>
<td>CORRELATION ID</td>
<td>CORRELATION_ID, CORRID, JOB_NAME, JOB_NAME</td>
</tr>
<tr>
<td>DATABASE NAME</td>
<td>DATABASE_NAME, DBNAME</td>
</tr>
<tr>
<td>PLAN NAME</td>
<td>PLAN_NAME, PLNAME</td>
</tr>
<tr>
<td>SUBSYSTEM NAME</td>
<td>SSID</td>
</tr>
<tr>
<td>TABLE NAME</td>
<td>TABLE_NAME, TBNAME</td>
</tr>
<tr>
<td>TABLESPACE NAME</td>
<td>TABLESPACE_NAME, TSNAME</td>
</tr>
</tbody>
</table>
**DB2 command definition**

Figure 37 shows the DB2 command definition syntax used in the WHERE clause of the LOGSCAN statement. Use this syntax to select log records based on the DB2 command they reflect.

The DB2 command definition is not valid if your log scan specifies any other type of output. By defining a filter that includes DB2 commands and generating a Commands report, you can select information about changes resulting from DB2 commands.

**Figure 37  DB2 Command definition syntax diagram**

<table>
<thead>
<tr>
<th>Syntax term</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT OF RECOVERY</td>
<td>UNIT_OF_RECOVERY, URID</td>
</tr>
<tr>
<td>UPDATE TYPE</td>
<td>UPDATE_TYPE</td>
</tr>
</tbody>
</table>

**DB2Command**

Specifies the DB2 command. A brief description of the available commands follows. You can use the full command name or a DB2-supported synonym. For more information about the DB2 commands, see the IBM documentation.

**ACCESS DATABASE**

Selects log records based on the ACCESS DATABASE command, which forces a physical open of or removes the group buffer pool (GBP) dependent status for a table space, index space, or partition.
**ALTER BUFFERPOOL**
Selects log records based on the ALTER BUFFERPOOL command, which changes attributes for an active or inactive buffer pool.

**ALTER GROUPBUFFERPOOL**
Selects log records based on the ALTER GROUPBUFFERPOOL command, which changes attributes of group buffer pools.

**ALTER UTILITY**
Selects log records based on the ALTER UTILITY command, which changes an execution of the REORG or REBUILD utilities.

**ARCHIVE LOG**
Selects log records based on the ARCHIVE LOG command, which closes a current active log and opens the next available log data set.

**CANCEL THREAD**
Selects log records based on the CANCEL THREAD command, which cancels processing for specific local or distributed threads.

**DISPLAY ARCHIVE**
Selects log records based on the DISPLAY ARCHIVE command, which displays input archive log information.

**DISPLAY BUFFERPOOL**
Selects log records based on the DISPLAY BUFFERPOOL command, which displays the current status of active or inactive buffer pools.

**DISPLAY DATABASE**
Selects log records based on the DISPLAY DATABASE command, which displays status information about DB2 databases and database objects.

**DISPLAY DDF**
Selects log records based on the DISPLAY DDF command, which displays status and configuration information for the distributed data facility (DDF) and connections or threads controlled by DDF.

**DISPLAY FUNCTION SPECIFIC**
Selects log records based on the DISPLAY FUNCTION SPECIFIC command, which displays statistics about external user-defined functions accessed by DB2 applications.

**DISPLAY GROUP**
Selects log records based on the DISPLAY GROUP command, which displays information about the data sharing group to which a DB2 subsystem belongs.
**DISPLAY GROUPBUFFERPOOL**
Selects log records based on the DISPLAY GROUPBUFFERPOOL command, which displays status information about DB2 group buffer pools and related statistics.

**DISPLAY LOCATION**
Selects log records based on the DISPLAY LOCATION command, which displays dynamic status information about a specified remote location.

**DISPLAY LOG**
Selects log records based on the DISPLAY LOG command, which displays log information about the offload task and the status of the task.

**DISPLAY PROCEDURE**
Selects log records based on the DISPLAY PROCEDURE command, which displays statistics about stored procedures that DB2 applications access.

**DISPLAY PROFILE**
Selects log records based on the DISPLAY PROFILE command, which displays whether profiling is active or inactive.

**DISPLAY RLIMIT**
Selects log records based on the DISPLAY RLIMIT command, which displays the current status of the resource limit facility.

**DISPLAY THREAD**
Selects log records based on the DISPLAY THREAD command, which displays current status information about DB2 threads.

**DISPLAY TRACE**
Selects log records based on the DISPLAY TRACE command, which displays a list of active traces.

**DISPLAY UTILITY**
Selects log records based on the DISPLAY UTILITY command, which displays the status of utility jobs, including jobs that are in a data sharing group.

**MODIFY TRACE**
Selects log records based on the MODIFY TRACE command, which changes the trace events (IFCID) associated with an active trace.

**RECOVER BSDS**
Selects log records based on the RECOVER BSDS command, which reestablishes dual bootstrap data sets (BSDS) after one BSDS has been disabled by a data set error.

**RECOVER INDOUBT**
Selects log records based on the RECOVER INDOUBT command, which recovers threads that are in an indoubt state because DB2 or a transaction manager could not automatically resolve the indoubt status with the commit coordinator.
RECOVER POSTPONED
Selects log records based on the RECOVER POSTPONED command, which completes back-out processing for units of recovery that are left incomplete during an earlier restart.

RESET GENERICLU
Selects log records based on the RESET GENERICLU command, which enables you to purge information VTAM stores in the coupling facility for partners of a DB2 subsystem.

RESET INDOUBT
Selects log records based on the RESET INDOUBT command, which purges the information that the indoubt thread report (generated by the DISPLAY THREAD command) displays.

SET ARCHIVE
Selects log records based on the SET ARCHIVE command, which sets the following maximums for the archive log:
- number of tape units
- deallocation time of tape units

SET LOG
Selects log records based on the SET LOG command, which modifies the checkpoint frequency that is specified during installation.

SET SYSPARM
Selects log records based on the SET SYSPARM command, which enables you to change subsystem parameters while DB2 is up.

START DATABASE
Selects log records based on the START DATABASE command, which makes a specified database available for processing.

START DB2
Selects log records based on the START DB2 command, which initializes the DB2 subsystem and makes it available to TSO applications and to other subsystems.

START DDF
Selects log records based on the START DDF command, which starts the distributed data facility (DDF).

START FUNCTION SPECIFIC
Selects log records based on the START FUNCTION SPECIFIC command, which starts an eligible external function that is stopped.
**START PROCEDURE**
Selects log records based on the START PROCEDURE command, which
- activates the definition of a stored procedure that is stopped
- refreshes a stored procedure that is stored in the cache

**START PROFILE**
Selects log records based on the START PROFILE command, which loads or reloads
the profile table into a data structure in memory.

**START RLIMIT**
Selects log records based on the START RLIMIT command, which starts the resource
limit facility and specifies a resource limit specification table for the facility to use.

**START TRACE**
Selects log records based on the START TRACE command, which starts DB2 traces.

**STOP DATABASE**
Selects log records based on the STOP DATABASE command, which makes specified
objects unavailable for applications and closes their data sets.

**STOP DB2**
Selects log records based on the STOP DB2 command, which stops the DB2
subsystem.

**STOP DDF**
Selects log records based on the STOP DDF command, which stops the distributed
data facility (DDF) if it has been started and terminates the DDF interface to VTAM or
TCP/IP.

**STOP FUNCTION SPECIFIC**
Selects log records based on the STOP FUNCTION SPECIFIC command, which
prevents DB2 from accepting SQL statements with invocations of the specified
functions.

**STOP PROCEDURE**
Selects log records based on the STOP PROCEDURE command, which prevents DB2
from accepting SQL CALL statements for stored procedures.

**STOP PROFILE**
Selects log records based on the STOP PROFILE command, which stops or disables
the profile function.

**STOP RLIMIT**
Selects log records based on the STOP RLIMIT command, which
- stops the resource limit facility
- resets all previously set limits to infinity
- resets the accumulated time to zero
**STOP TRACE**
Selects log records based on the STOP TRACE command, which stops tracing.

**TERM UTILITY**
Selects log records based on the TERM UTILITY command, which terminates the execution of a DB2 utility job step and releases resources associated with that step.

**IN (DB2Command)**
Log Master can select log records based on whether they resulted from a single DB2 command or a set of commands. The IN keyword indicates a set of multiple DB2 commands. Enclose the set of commands within parentheses and separate multiple commands with commas.

Example:

```
IN (STOP DB2, START DB2)
```

**NOT**
Specifies that log records are selected when the DB2 command reflected in them does not match any member of a set of multiple DB2 commands.

Example:

```
NOT IN (STOP DATABASE, START DATABASE)
```
LOGMARK statement

Figure 38 shows the LOGMARK statement syntax. Use the LOGMARK statement to associate a symbolic name (called a log mark name) with the current RBA/LRSN value of the DB2 log. You can use the log mark name in subsequent product syntax in place of the RBA/LSRN value. Log Master stores information about log marks in the ALPMARK table in the Repository.

LOGMARK logMarkName

Specifies a name that Log Master associates with the RBA/LRSN. For more information about log marks and log mark names, see “MARK logMarkName” on page 274.

DESC "description"

Defines an optional, 65-character text string. Use the text description to store any comments that you want to associate with the log mark name.

QUIESCE (databaseName.tableSpaceName)

Directs Log Master to use the DB2 Quiesce utility to generate a quiesce point for the table spaces defined by databaseName.tableSpaceName. Log Master associates the log mark name with the RBA/LRSN of the quiesce point. Separate multiple table spaces with commas.

The TABLESPACESET keyword directs Log Master to establish a quiesce point for the specified table spaces, all referentially related table spaces, and any related large object (LOB) table spaces or base table spaces that are not part of the referential integrity set.
**EXECSQL statement**

Figure 39 shows the EXECSQL statement syntax. The EXECSQL statement directs Log Master to execute a set of SQL statements against a DB2 database. Log Master uses the High-speed Apply Engine to execute SQL statements.

**Figure 39  EXECSQL statement syntax diagram**

![EXECSQL syntax diagram](image)

**SQLDataSetName**

Specifies the name of a data set containing valid ANSI SQL statements.

**BINDOWNER** *(authid)*

Specifies an authorization ID that has authority within DB2 security to bind programs and generate plans. Log Master uses this authorization ID to bind an application plan whenever Log Master executes generated SQL statements.

The default value of this keyword depends on the value of the BINDOWN installation option (for more information, see “BINDOWN=” on page 43). If your environment does not specify a value for BINDOWN, Log Master uses the user ID of the user who submitted the job.

**DECIMAL POINT**

Defines the character that was used to delineate fractional values in the SQL generated by the product. The default value for this keyword is the character specified in the DSNHDECP module of the target DB2 subsystem.

**PERIOD**

Defines the decimal point character as a period (.)
COMMA

Defines the decimal point character as a comma (,).

PLAN NAME (planName)

Specifies the name of an existing plan that Log Master uses to access data during execution of the generated SQL. Log Master creates or updates this plan during processing and uses it to access the required data. The default value for planName is the character constant ALP, followed by the job number (for example, 'ALP00302').

See also “PLANESQL=ALP” on page 42.

FREE | NOFREE

The FREE and NOFREE keywords determine the persistence of information about the plan defined by planName.

FREE

Indicates that Log Master deletes plan information from the DB2 catalog after processing. This is the default value.

NOFREE

Indicates that Log Master retains plan information in the catalog for a subsequent run of the job.

SQLPRINT

Determines the content of the SQLPRINT data set. Log Master writes the results of SQL execution (including SQL codes and statement identifiers) into the SQLPRINT data set as it executes generated SQL statements.

WARN

Causes the SQLPRINT data set to include only the results of SQL statements that receive nonzero SQL codes or SQL warnings. This is the default value.

ALL

Causes the SQLPRINT data set to include the results of all executed SQL statements.

See also “SQLPRINT” on page 101.
DROPRECOVERY statement

Figure 40 shows the DROPRECOVERY statement syntax. The DROPRECOVERY statement enables Log Master to work with the BMC Software product RECOVER PLUS (or DSN1COPY) to recover DB2 objects that have been dropped from the DB2 catalog.

Figure 40  DROPRECOVERY statement syntax diagram
During a drop recovery job step, Log Master obtains information from the DB2 log and (depending on the type of recovery requested) creates the following forms of output:

- data definition language (DDL) statements
- either
  - RECOVER PLUS SYSIN syntax
  - DSN1COPY syntax (and other associated files)
- SQL statements
- DB2 Command

The Log Master online interface also creates JCL to run the different types of output in the correct order to accomplish the recovery. The generated JCL contains multiple job steps. If necessary, you can edit the JCL to adapt it to your environment.

The DROPRECOVERY statement determines what actions Log Master takes to obtain the log information, what types of output it produces, and what syntax Log Master generates for RECOVER PLUS or DSN1COPY.

Be aware of the following points as you specify an automated drop recovery:

- To get meaningful results, you must select a minimum of
  - one type of object to recover (using the DATABASE NAME, TABLESPACE NAME, or TABLE NAME keywords)
  - one form of output that processes recovery information (using the RECOVER DATASET keyword or the REPORT keyword)

- Log Master supports the use of RECOVER PLUS to recover table spaces and tables. Log Master supports the use of DSN1COPY only to recover a single dropped table space.

- The DB2 Recover utilities that Log Master works with do not support the recovery of “versioned” dropped objects. A DB2 object is versioned during the period after its structure is changed by a DDL statement (for example, an ALTER COLUMN SET DATA TYPE statement) and before a DB2 Reorg utility is run to reconcile the multiple versions.

- To perform a drop recovery, Log Master must perform row completion processing on DB2 catalog log records. Because of the large number of log records related to the catalog, this processing can cause your job to run longer and require more resources than a job that does not read the DB2 catalog. If you frequently operate on objects stored in the catalog, you can improve performance by creating more frequent image copies of the catalog or by defining tables in the DB2 catalog with Data Capture Changes (DCC). Table 3 on page 65 shows which tables should be defined with DCC or Data Capture None (DCN).
Log Master provides automated drop recovery for LOB and XML table space or table objects. When you specify a LOB or XML base table or table space in the DROPRECOVERY statement, Log Master extends drop recovery discovery to include all of its LOB and XML auxiliary table spaces and tables.

Log Master supports the use of copies registered in BMCXCOPY to recover table spaces, with the exception of cabinet copies.

**DATABASE NAME **`databaseName`

Directs Log Master to obtain information about and generate output to recover a complete database that has been dropped from the DB2 catalog. The `databaseName` defines the name of the database as it was known to DB2 before it was dropped.

**TABLESPACE NAME DSNDB04 | databaseName.tablespaceName**

Directs Log Master to obtain information about and generate output to recover the dropped table space defined by a given database name and table space name.

**DSNDB04**

Directs Log Master to create syntax to recover a table space stored in DB2’s default database. This is the default value.

**databaseName**

Directs Log Master to create syntax to recover a table space stored in the database defined by `databaseName`.

**tablespaceName**

Directs Log Master to create syntax to recover the table space defined by `tablespaceName`.

**Recover parameters**

Specifies the values of keywords in RECOVER PLUS syntax. The specific keywords in the Recover parameters relate to the output image copies that RECOVER PLUS can create as it recovers a table space. For more information about these parameters, see “Recover parameters” on page 317.

Depending on the utility programs available in your environment, you might be able to recover a single dropped table space by using DSN1COPY instead of RECOVER PLUS. For more information, see “DSN1COPY DATASET Output definition” on page 315.
TABLE NAME *userId.tblName*

Directs Log Master to obtain information about and generate output to recover a specific table defined by *tblName*. Use this keyword when a table has been dropped, but the original database and table space still exist in the DB2 catalog.

*userId*

Specifies an optional user ID that Log Master uses to qualify the *tblName* in RECOVER PLUS syntax.

*tblName*

Directs Log Master to create RECOVER PLUS syntax to recover the table defined by *tblName*.

To recover a dropped table, Log Master generates data sets containing DDL, SQL and RECOVER PLUS syntax. If you use the Log Master online interface, Log Master also generates JCL that uses the generated data sets to:

- create a temporary copy of the original table space
- create a temporary copy of the empty table within the temporary table space
- re-create the empty table structures within the original table space
- recover data into the temporary copy of the table

If the original dropped table is a dependent of a referential integrity constraint, you might need to modify the JCL before running it. For more information about troubleshooting automated drop recoveries, see the chapter on recovering dropped objects in the *Log Master for DB2 User Guide*.

- migrate data from the temporary copy of the table into the empty table structure within the original table space
- delete the temporary table and temporary table space

**NOTE**

When a table space contains a single table, you can improve drop recovery performance by recovering the dropped table directly into the original table space. For more information, see “INDEP YES | NO” on page 321.
Temporary objects definition

Defines whether Log Master creates temporary copies of the table defined by `tableName` (and the table space that `tableName` was originally part of) during drop recovery. These keywords also define the physical and logical location where RECOVER PLUS can store the temporary copies. When defined, Log Master uses this information in the RECOVER PLUS INDEP OUTSPACE syntax. For more information, see “Temporary objects definition” on page 320.

RECREATE DATASET Output definition

Specifies the characteristics of the output data set that contains generated data definition language (DDL) statements. When executed, the DDL statements recreate the structures of the dropped objects (using UNDO DDL) and re-establish their relationships (including granting authorizations, re-creating VIEWs, and so forth). They also create structures for the temporary objects used to recover a dropped table. Output choices for this data set are similar to other data sets. For more information, see “LOGSCAN output definition” on page 260.

EXECUTE

Directs Log Master to execute the generated DDL statements in the data set specified by the RECREATE DATASET keyword.

SQLID  `authID`

Directs Log Master to generate a SET CURRENT SQLID statement and insert it at the beginning of the data set specified by the RECREATE DATASET keyword. When you execute the DDL in the data set, DB2 runs it under the authority granted to the authorization ID represented by `authid`.

Specify an authorization ID that has the appropriate authority in your environment.

BINDOWNER  `authID`

Specifies an authorization ID that has authority within DB2 security to bind programs and generate plans. Log Master uses this authorization ID to bind an application plan when Log Master executes the SQL or DDL statements that it generates as part of a drop recovery action.

The default value of this keyword depends on the value of the BINDOWN installation option (for more information, see “BINDOWN=” on page 43). If your environment does not specify a value for BINDOWN, Log Master uses the user ID of the user who submitted the job.
**RECOVER DATASET Output definition**

Specifies the characteristics of the output data set that contains generated SYSIN syntax for RECOVER PLUS. When RECOVER PLUS executes the syntax, it recovers the dropped objects. Depending on the type of recovery you request, RECOVER PLUS recovers the objects either to their original database location or to a temporary location. The output choices for this data set are the same as those for other product data sets. For more information, see “LOGSCAN output definition” on page 260.

Log Master supports the automatic drop recovery of a clone table. To accomplish this, Log Master generates RECOVER PLUS syntax in the following files:

- **RECOVER output** has RECOVER PLUS syntax for recovering base tables and table spaces.
- **RECOVER2 output** has RECOVER PLUS syntax for clone tables.

If no clone tables are involved in the recovery, RECOVER2 is empty.

**NOTE**

Image copies of clone tables are required to recover clone tables.

If dropped objects have clone tables, Log Master generates syntax that instructs RECOVER PLUS to use image copies to recover the clone tables. (You do not have to make any DROPRECOVERY command syntax change for the clone recovery.) Log Master allocates a RECOVER2 output data set by using the same attributes as ones for RECOVER output. RECOVER2 output can be DSN model, PDS, or GDG, as well as OUTCOPYDDN for clone tables. The naming convention of the RECOVER2 output data set is based on types of the RECOVER data set. Table 25 describes each data set type for RECOVER 2 data sets and Table 26 provides examples of RECOVER and RECOVER2 output file names.

**Table 25  Data set type and description of RECOVER2 data set name**

<table>
<thead>
<tr>
<th>Data set type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN model</td>
<td>The RECOVER2 data set name is RECOVER DSN name model appended with .R2. Log Master limits the RECOVER DSN name model to 41 bytes or less.</td>
</tr>
<tr>
<td>Partitioned Data Set (PDS) member names</td>
<td>RECOVER2 uses the same PDS as RECOVER. The RECOVER2 member name is RECOVER member name appended with R if the name is less than 8 bytes; otherwise, the 8th byte is replaced with R.</td>
</tr>
<tr>
<td>Generated Data Group (GDG)</td>
<td>RECOVER2 uses a GDG base similar to the RECOVER data set.</td>
</tr>
</tbody>
</table>
**DROPRECOVERY statement**

**LOGONLY**

Directs Log Master to generate syntax that instructs RECOVER PLUS to use only data from log records to perform the recovery. By default, Log Master instructs RECOVER PLUS to use image copies during recovery. Use this keyword only if you do not want RECOVER PLUS to use image copies (for example, if your site restores data from backups outside of DB2 during recovery). This keyword applies to all of the DB2 objects in the current drop recovery action.

**OUTCOPY**

Directs Log Master to generate syntax that determines how RECOVER PLUS creates output image copies for partitioned DB2 objects. Depending on the input image copies (the image copies that Log Master selects as input sources for the drop recovery), RECOVER PLUS can create separate output image copies for each partition or create one combined image copy for the entire object. This keyword applies to the output image copies that RECOVER PLUS creates for all table spaces that you are recovering.

Be sure to define data set names for your output image copies before you run the automated drop recovery. For more information, see “Recover parameters” on page 317.

If a table space to recover has a clone table, Log Master directs RECOVER PLUS to create separate output image copies for the clone.

**ASCODED**

Creates output image copies in the same format (either DSNUM 0 or partitioned) as the input image copies. If you want to know the format of the input image copies, review the information in the Drop Recovery Report.

**BYPART**

Creates an output image copy for each partition in the table space (regardless of the format of the input image copies).

---

**Table 26 Examples for RECOVER and RECOVER2 output file names**

<table>
<thead>
<tr>
<th>Data set type</th>
<th>RECOVER</th>
<th>RECOVER2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN model</td>
<td>MYTS.DROPREC.CNTL</td>
<td>MYTS.DROPREC.CNTL.R2</td>
</tr>
<tr>
<td>PDS</td>
<td>MYTS.DROPREC.CNTLPSD(MYCNTL)</td>
<td>MYTS.DROPREC.CNTLPSD(MYCNTLR)</td>
</tr>
<tr>
<td>GDG</td>
<td>MYTS.DROPREC.CNTLGDG(+1)</td>
<td>MYTS.DROPREC.CNTLGDG(+1)</td>
</tr>
</tbody>
</table>
You might be able to recover a single dropped table space by using DSN1COPY instead of RECOVER PLUS. For more information, see “DSN1COPY DATASET Output definition.” You cannot generate RECOVER PLUS syntax and DSN1COPY syntax in the same drop recovery action.

**DSN1COPY DATASET Output definition**

Specifies the characteristics of a partitioned data set (PDS) that contains generated SYSIN syntax for the IBM utility DSN1COPY and other associated files. Log Master supports the use of DSN1COPY only to recover a single dropped table space.

The JCL that Log Master generates for a drop recovery action with DSN1COPY contains additional job steps to perform tasks that are otherwise handled by RECOVER PLUS. You cannot generate RECOVER PLUS syntax and DSN1COPY syntax in the same drop recovery action.

Log Master generates the following members in this PDS:

- SYSIN syntax for the DSN1COPY utility (member name COPY)
- object identifier (OBID) information for DSN1COPY (member name XLAT)
- commands for the DSNUTILB utility to rebuild indexes on the table space (member name REBUILD)
- SYSIN syntax for an additional run of Log Master to generate MIGRATE SQL (member name MIGRATE)

Log Master generates MIGRATE SQL to capture changes to the table space from the point when the image copy data set was created to the point when the table space was dropped.

For the output choices available for this PDS, see “LOGSCAN output definition” on page 260.

**CHECK DATASET Output definition**

DB2 might place a table space into check pending status after the recovery job step in drop recovery processing. This option specifies the characteristics of the output data set that contains generated DB2 commands that remove the check pending status. When the commands are executed, they perform

- CHECK DATA on table spaces with an RI constraint
- CHECK DATA on LOB or XML table spaces
- CHECK DATA on auxiliary LOB table spaces
- REBUILD INDEX on temporary auxiliary LOB tables
The output choices for this data set are the same as those for other product data sets. For more information, see “LOGSCAN output definition” on page 260.

**MIGRATE DATASET Output definition**

Specifies the characteristics of the output data set that contains generated SQL statements. Use this keyword only if you have used the TABLE NAME keyword to recover an individual table. When the SQL statements are executed, they migrate data from a temporary table in a temporary table space into a re-created empty version of the table in the original table space. The output choices for this data set are the same as those for other product data sets. For more information, see “LOGSCAN output definition” on page 260.

**RUNSTATS DATASET Output definition**

Specifies the characteristics of the output data set that contains generated RUNSTATS commands. When the commands are executed, they execute the RUNSTATS utility before the REBIND step is executed. Depending on the dropped object that you are recovering, Log Master writes RUNSTATS commands into the RUNSTATS data set as follows:

- If you are recovering a dropped database, Log Master generates the RUNSTATS command for all dropped table spaces, (except auxiliary table spaces), for the specified database. (No statistics on the LOB table space affect access path selection.) For example:

  | RUNSTATS TABLESPACE DBNAME.TNAME_1 TABLE(ALL) INDEX(ALL) |
  | RUNSTATS TABLESPACE DBNAME.TNAME_2 TABLE(ALL) INDEX(ALL) |

- If you are recovering a dropped table space, Log Master generates the RUNSTATS command for the specified table space. For example:

  | RUNSTATS TABLESPACE DBNAME.TNAME TABLE(ALL) INDEX(ALL) |

- If you are recovering a dropped table, Log Master generates the RUNSTATS command for the specified table and its indexes. For example:

  | RUNSTATS TABLESPACE DBNAME.TNAME TABLE(CREATOR.MYTABLE) |
  | RUNSTATS INDEX (CREATOR.MYTABLE_INDEX_1) |
  | RUNSTATS INDEX(CREATOR.MYTABLE_INDEX_2) |

The output choices for this data set are the same as those for other product data sets. For more information, see “LOGSCAN output definition” on page 260.
REBIND DATASET Output definition

Specifies the characteristics of the output data set that contains generated DB2 commands. When the commands are executed, they rebind any application plans that were invalidated when the DB2 objects were dropped. The output choices for this data set are the same as those for other product data sets. For more information, see “LOGSCAN output definition” on page 260.

FROM Range definition TO Range definition

Specifies a portion of the DB2 log when the DB2 objects were dropped. Log Master scans this portion of the log (also referred to as a time frame) to obtain information needed for the drop recovery action. FROM indicates the start point and TO indicates the end point of the log scan. Each Range definition defines one point in the DB2 log. For more information, see “Range definition” on page 272.

REPORT definition

Directs Log Master to create an output report. The Drop Recovery report contains information about the specified recovery, including information about the dropped DB2 objects, the object ID translation (OBIDXLAT) information, and the application plans that were invalidated when the DB2 objects were dropped. For more information, see “Report definition” on page 322.

Recover parameters

Figure 41 on page 318 shows the Recover parameters syntax used in the DROPRECOVERY statement. During a drop recovery step, Log Master creates syntax that it passes to RECOVER PLUS. Using the syntax, RECOVER PLUS can recover a table space that has been dropped from the DB2 catalog.

The Recover parameters specify the values of some keywords in this syntax. The specific keywords of this definition relate to the output image copies that RECOVER PLUS can create as it recovers the table space.

Depending on the utility programs available in your environment, you might be able to recover a single dropped table space by using other methods (possibly with DSN1COPY). For more information, see “DSN1COPY DATASET Output definition” on page 315.
Recover parameters

**REGISTER**

Directs Log Master to create SYSIN syntax that defines how RECOVER PLUS registers _output_ image copies in SYSIBM.SYSCOPY within the DB2 catalog.

**ALL**

Specifies that all output image copies are registered in the SYSCOPY table.

**NONE**

Specifies that none of the output image copies are registered in the SYSCOPY table.

**_(ddName)_**

Specifies a set of individual output image copies that are registered in the SYSCOPY table, using the DD statement name defined by _ddname_. To register multiple image copies, separate DD names with commas.

**OUTCOPYDDN**

Directs Log Master to create SYSIN syntax that defines what DD names or DD name prefixes RECOVER PLUS uses to create local output image copies.

**NOTE**

If a dropped table space has a clone table, Log Master directs RECOVER PLUS to use the same DD name prefix for output image copies of the clone table in the RECOVER PLUS syntax as is used for the dropped table space. In the DD name definition for the clone, be sure to define a data set name for the clone output that is different than the one for base table space OUTCOPY.
**BMCCPY**

Directs RECOVER PLUS to use the default DD name prefix for local primary output image copies. If the table space has less than 100 partitions, RECOVER PLUS uses the characters BMCCPY, followed by a two-digit number. If the table space has 100 partitions or more, RECOVER PLUS uses the characters BMCCY and a three-digit number.

**BMCCPZ**

Directs RECOVER PLUS to use the default DD name prefix for local backup output image copies. If the table space has less than 100 partitions, RECOVER PLUS uses the characters BMCCPZ, followed by a two-digit number. If the table space has 100 partitions or more, RECOVER PLUS uses the characters BMCCZ and a three-digit number.

**ddname01 | ddname02**

Directs RECOVER PLUS to use exact DD names for local output image copies. Specify *ddname01* for local primary image copies or *ddname02* for local backup image copies.

**ddprefix01 | ddprefix02**

Directs RECOVER PLUS to use DD name prefixes for local output image copies. Specify *ddprefix01* for local primary image copies or *ddprefix02* for local backup image copies.

**RECOVERYDDN**

Directs Log Master to create SYSIN syntax that defines what DD names or DD name prefixes RECOVER PLUS uses to create remote output image copies.

**BMCRCY**

Directs RECOVER PLUS to use the default DD name prefix for remote primary output image copies. If the table space has less than 100 partitions, RECOVER PLUS uses the characters BMCRCY, followed by a two-digit number. If the table space has 100 partitions or more, RECOVER PLUS uses the characters BMCRY and a three-digit number.

**BMCRCZ**

Directs RECOVER PLUS to use the default DD name prefix for remote backup output image copies. If the table space has less than 100 partitions, RECOVER PLUS uses the characters BMCRCZ, followed by a two-digit number. If the table space has 100 partitions or more, RECOVER PLUS uses the characters BMCRZ and a three-digit number.
**Temporary objects definition**

**ddname03 | ddname04**

Directs RECOVER PLUS to use exact DD names for remote output image copies. Specify `ddname03` for remote primary image copies or `ddname04` for remote backup image copies.

**ddprefix03 | ddprefix04**

Directs RECOVER PLUS to use DD name prefixes for remote output image copies. Specify `ddprefix03` for remote primary image copies or `ddprefix04` for remote backup image copies.

**Temporary objects definition**

Figure 42 shows the Temporary objects definition syntax used in the DROPRECOVERY statement. This syntax determines

- whether Log Master creates temporary copies of the dropped table and table space as it recovers a dropped table.
- the physical and logical locations where Log Master stores the temporary copies.

This syntax applies only to recovery of individual tables, not table spaces. Log Master includes this information in generated data definition language (DDL) statements, the INDEP OUTSPACE syntax that it can pass to RECOVER PLUS, and generated SQL statements.

**Figure 42   Temporary objects definition syntax diagram**
INDEP YES | NO

Specifies whether Log Master creates and uses temporary copies to recover the dropped table. Use the default value of YES for normal processing.

YES

Directs Log Master to create temporary copies of the dropped table and the table space that originally contained the dropped table. This is the default value. Log Master generates INDEP OUTSPACE syntax and passes it to RECOVER PLUS. For more information, see “TABLE NAME userID.tableName” on page 311.

NO

Use this value only when a table space contains a single table. Log Master directs RECOVER PLUS to recover the dropped table directly into the original table space (which makes the table space unavailable for a period of time). For a single-table table space, this value improves drop recovery performance because Log Master avoids generating and executing MIGRATE SQL to move data from the temporary copy of the table into the original location. When you specify NO, Log Master ignores any locations that you specify for temporary objects.

STOGROUP storageGroupName

Specifies which DB2-defined storage group RECOVER PLUS uses to create temporary copies of DB2 objects during the recovery of a dropped table. If you do not specify a storage group, Log Master directs RECOVER PLUS to use the storage group where the original table space was stored.

DATABASE databaseName

Specifies the DB2 database where RECOVER PLUS creates temporary copies of DB2 objects during the recovery of a dropped table. The default value for databaseName is ALPTMPDB.

TABLESPACE tableSpaceName

Specifies the table space where RECOVER PLUS creates temporary copies of a dropped table during the recovery. The default value for tableSpaceName is ALPTMPTS.
OWNER userID

Specifies the user ID that RECOVER PLUS uses as it creates temporary copies of a dropped table during the recovery. The default value for userID is ALPTMPUS.

Report definition

Figure 43 shows the Report definition syntax of the DROPRECOVERY statement. Use this syntax to specify a report template and an output location for the report that Log Master generates.

Figure 43 DROPRECOVERY report syntax diagram

TEMPLATE

Directs Log Master to format the generated report by using a report template that is stored in an external data set or in the Log Master Repository. If you do not specify a template, Log Master uses a standard report format.

To create a report template, use the Log Master online interface to define the template. Use options in the interface to export the template into a data set or store it in the Repository. Log Master stores the report template in the form of XML elements and attribute values.

The report type that you specify in your job or job step must match the report type defined in the template. A template reflects the report options that were selected when the template was created. To change the options, you must modify the template.
DATASET

Defines a data set that contains a template that Log Master uses to format the generated report. The data set must contain a valid Log Master report template.

ID

Defines the identity of a template in the Log Master Repository. Log Master uses the template to format your generated report. To use a template from the Repository, the template must have been created in or imported into the Repository on the DB2 subsystem (SSID) where your job or job step runs. To use the same template on multiple subsystems, export the template to a data set.

DATASET Output definition

Directs Log Master to place the Drop Recovery report in a data set. When you specify the DATASET keyword, you also must specify the data set name and disposition. The output choices for this data set are the same as those for other product data sets. For more information, see “LOGSCAN output definition” on page 260.

SYSOUT SYSOUT definition

Directs Log Master to send the Drop Recovery report to the Job Entry System (JES) spool queue (known as SYSOUT). For more information about specifying JES SYSOUT parameters, see “SYSOUT definition” on page 197.

Symbolic substitutions

Table 27 contains a list of valid symbolic substitutions. You can use these symbolic names with Log Master syntax or the online interface as indicated. When your job runs, Log Master replaces the symbolic name with the corresponding generated data.

<table>
<thead>
<tr>
<th>Substitution</th>
<th>Description</th>
<th>Input data set names</th>
<th>Input member name</th>
<th>Input log mark names</th>
<th>Output data set names</th>
<th>Output member names</th>
<th>Output log mark names</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATE</td>
<td>Current date (ymmd)</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>&amp;GATN</td>
<td>Current group attachment name</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>&amp;JOBNAME</td>
<td>Batch job name</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
If you use either the &TABNAME or &WORKID symbolic names in a data set name, Log Master must process the original name of the table or work ID to create a valid operating system data set name. To create a valid data set name, Log Master performs the following actions:

- splits the original name into nodes of 8 characters (as many as required)
- (optional) inserts an alphabetic qualifier at the beginning of each node ('T' for a table name or 'W' for work ID)

Use the DSN SYMPRE installation option (page 51) to change how Log Master sets the symbolic prefix for the data set name. If YES, the prefix is added at the beginning of each node. If NO, the prefix is added only if needed to create a valid data set name.

- uses 7 characters from the original name in each node
- honors the operating system’s 44-character limit on the length of a data set name

  If necessary, Log Master truncates the original name to ensure a 44-character name. Log Master preserves any constant strings in the data set name that you specified. Table 28 on page 325 shows examples of how Log Master generates and truncates data set names.

- if the table name length is greater than 18 characters, uses the table’s short name, which is the first 9 bytes of the table name and “#DBIDOBID” after translation, where DBIDOBID is the table’s DBID and OBID in hexadecimal (for example, TABLENAME#00AB00A1)

<table>
<thead>
<tr>
<th>Data set name in SYSIN syntax</th>
<th>Original name of table or work id</th>
<th>Data set name in generated JCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATASET &amp;WORKID.LOAD.DATA</td>
<td>LOGSCAN24</td>
<td>WLOGSCAN.W24.LOAD.DATA</td>
</tr>
<tr>
<td>DATASET &amp;TABNAME.LOAD.DATA</td>
<td>AVERYLONGTABLENAME</td>
<td>TAVERYLO.TNGTABLE.TNAME.LOAD.DATA</td>
</tr>
<tr>
<td>DATASET PROG01.SORTED.&amp;TABNAME.LOAD.DATA</td>
<td>AVERYLONGTABLENAME</td>
<td>PROG01.SORTED.TAVERYLO.TNGTABLE.TN.LOAD.DATA</td>
</tr>
</tbody>
</table>

**&DATE or &TIME in data set or log mark names**

If you use either the &DATE or &TIME symbolic names in a data set name or a log mark name, insert a one-letter prefix before the variable. For example, use 'D&DATE' or 'T&TIME'.

**&SSID, &GATN, and &ZPREFIX resolved by online interface**

When the &SSID or the &GATN symbolic names are defined on the Generate JCL panel of the online interface, Log Master resolves the symbolics as it generates JCL. In any other context, the symbolics are resolved when the job or job step is executed.

The &ZPREFIX symbolic name is available only in the online interface and is resolved when JCL is generated. If no defined prefix exists when JCL is generated, Log Master inserts the literal string &ZPREFIX. If you specify this name in batch syntax, Log Master issues an error message.
Log Master nonSYSIN syntax

Log Master supports some additional types of syntax that are not included in the SYSIN input. You can enter the following types of syntax:

- Syntax that defines the structures of DB2 objects that are no longer in the DB2 catalog and are not stored in the Repository.

  Enter this syntax in an old objects data set. Use either the online interface or the OLD OBJECTS keyword of the OPTION statement to direct Log Master to use the structures that you have defined.

- Syntax that defines how Log Master responds to SQL codes as it executes generated SQL statements against a database.

  Enter this syntax in an SQL codes data set. Use DD statements in your JCL to define your data set as the SQLCODES data set. Log Master can then use your specifications to respond to any SQL codes.

- Syntax that defines how Log Master translates table or column names as it generates SQL.

  Enter this syntax in an SQL translate data set. Use DD statements in your JCL to define your data set as the SQLXLAT data set. Log Master can then use your specifications to change the DB2 object names in generated SQL.

Old objects data set syntax

Figure 44 shows the syntax used within the old objects data set. To direct Log Master to read an old objects data set as input, use the appropriate syntax of the OPTION statement. For more information about the OPTION syntax, see “OLD OBJECTS dataSetName” on page 122. For more information about the old objects data set, see the Log Master for DB2 User Guide.
Log Master can read descriptions of DB2 objects that are no longer in the DB2 catalog from an old objects data set. The basic syntax, displayed in Figure 44 on page 327, is similar to the DB2 syntax for a CREATE TABLE statement. Notice that the old objects data set syntax precedes the actual DDL that created the table (createTableDDL).

The $n$ values used with the CCSID keyword represent coded character set IDs, (in order, a single-byte character set ID, a double-byte character set ID, or a mixed/graphic CCSID).

You can repeat the old objects definition syntax as many times as necessary to define multiple objects in a single old objects data set. In addition to the fields shown in Figure 44 on page 327, you can use the UNIQUE or PRIMARY KEY clause of the CREATE TABLE data definition language (DDL) statement to associate an index with a table definition. The index name is tbcreator.OLDINDEX. To allow for ascending or descending columns in the index, Log Master supports the optional keywords ASC or DESC after each column in the UNIQUE or PRIMARY KEY clause.
**SQLCODES data set syntax**

Figure 45 shows the syntax that you can use in the SQLCODES data set. For more information, see “SQLCODES” on page 101. For information about how to define responses to SQL codes, see the chapter on executing SQL in the *Log Master for DB2 User Guide*.

**Figure 45  SQLCODES syntax diagram**

![SQLCODES syntax diagram]

**NOTE**

The x value of the SQLWARN keyword can be a numeric character from zero through nine, or the letter A.

**SQLXLAT data set syntax**

Figure 46 shows the syntax to use within the SQLXLAT data set. Use this syntax to change the names of tables and columns as you generate SQL. This capability can be helpful when you use SQL statements to migrate data from one system or platform to another. Specify the existing names of tables or columns and the corresponding new names that you want to use in the generated SQL statements.

**Figure 46  SQLXLAT syntax diagram**

![SQLXLAT syntax diagram]
Remember the following points:

- The columns referenced in a COLUMN specification must be part of the table referenced in the preceding TABLE specification.

- To translate a column without translating a table name, you must specify both tableName and newTableName as the same character string.

Example:

    TABLE user01.firstTable -> user01.firstTable
    COLUMN account -> otherAccount

- To specify multiple tables in one specification, use an asterisk (*) as a wildcard character in place of any userID, tableName, or newTableName. Log Master translates the names of all tables that match your specified pattern. The asterisk represents a complete user ID or table name (you cannot match on part of a name).

You cannot specify multiple columns in one specification (Log Master does not support wildcard characters in column names).

For an example of SQL name translation, see the Log Master for DB2 User Guide. For more information about the SQLXLAT DD statement, see “SQLXLAT” on page 101.
SQLXLAT data set syntax
This chapter describes the format and content of the logical log that the Log Master for DB2 product creates. The chapter also describes how Log Master handles log records with long record lengths (segmented logical log records). Specifically, this chapter contains the following topics:

Logical log components .......................................................... 332
Logical log control file ............................................................ 332
  Type record (XTYP) ............................................................ 333
  Control file ID record (DCNT) .............................................. 335
  Data sharing member record (DDSM) ..................................... 336
  Work ID record (DWRK) .................................................... 336
  Work ID description record (DWKD) ...................................... 337
  Logical log data set record (DLDS) ....................................... 338
  Logical log description record (DLUS) ................................. 339
  Log information record (DLGI) ........................................... 340
  Log information end record (DLGE) ...................................... 341
  Open transaction record (DOTR) .......................................... 342
  Current path record (XPTH) .............................................. 343
  SysStrings base record (DSS1) .......................................... 343
  SysStrings transtable record (DSS2) .................................... 344
  Data change header description record (XHDF) ....................... 345
  Table space information record (DTSI) ................................. 346
  Table information record (DTBI) ........................................ 347
  Table byte count record (DTBB) ......................................... 349
  Table byte count record (DTBB) ......................................... 349
  Table record count record (DTBC) ...................................... 350
  Table synonym and alias record (DTOB) ............................... 351
  DB2 column information record (DCLI) ................................. 352
  DB2 column description record part two (DCL2) ...................... 354
  DB2 column description record part two (DCL2) ...................... 354
  Logical log column information record (DLCI) ....................... 355
  LOB column description record (DLOB) ............................... 357
  LOB column description record (DLOB) ............................... 357
  XML column description record (DXML) .............................. 359
  XMLSTRING data set name record (DXSF) ............................ 361
  LOB/XML cluster data set record (CLUS) ............................. 361
Logical log components

The product creates the following physical files for each logical log that you request:

- A logical log control file that contains information about the format and content of the logical log.

- A logical log data file that contains actual change information (updates, deletes, and inserts) in logical log format (for more information, see "Logical log data file" on page 369).

- (optional) A logical log XMLSTRING control file that defines the characteristics of a data set where the product stores the string IDs and string data that DB2 uses to encode the data in XML columns.

The product needs this file only to migrate data to a different DB2 subsystem. When you apply the changes from the logical log to a different target subsystem, Log Master or the High-speed Apply Engine use the string IDs and data to serialize your XML data correctly. The product obtains string IDs and data from the DB2 catalog table SYSIBM.SYSXMLSTRINGS. By default, the product does not generate this file.

Logical log control file

The logical log control file defines the layout and content of the logical log data file. On mainframe targets, the control file is a variable block (VB) output file. Think of the control file as linked to the corresponding logical log data file. Be sure to include it if you export a logical log for use on another platform or with other software.
The format of the logical log control file does not define an order for the different types of records that it contains. The only requirement is that the Type Record always appears first.

This section lists the logical log control records in the order in which they normally occur in a logical log, showing the layout of each record. The section also defines a record status for each record to describe the scope of the record’s information. The status can be one of the following:

- **DB2 specific**: contains information that is specific to the DB2 log and DB2 databases.

- **Platform independent**: contains information that can apply to other database software.

### Type record (XTYP)

Each logical log contains one instance of the Type Record. This record is present in a logical log created from any source. The record describes the originating DB2 subsystem, the version of the logical log format, and information about the default character sets and encoding schemes used in the originating database environment.

Record status: Platform independent

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: XTYP – Type Record</td>
</tr>
<tr>
<td>CNTLFILETYPE</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Type (format) of control file – ‘DB2’ represents DB2</td>
</tr>
<tr>
<td>CNTLFILETYPEVERSION</td>
<td>08</td>
<td>CHAR(6)</td>
<td>Version of control file, for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'V1010' = Log Master version 10.1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'ALP920' = Log Master version 9.2.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ This version might differ from the current version of Log Master, depending on when the file was generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ The version format for Log Master version 10.1.00 and later uses the format Vvrm; version 9.2.00 and earlier use the format ALPvrm.</td>
</tr>
<tr>
<td>CNTLFILESYSID</td>
<td>14</td>
<td>CHAR(4)</td>
<td>Originating DB2 subsystem ID</td>
</tr>
<tr>
<td>SQLDELIMITER</td>
<td>18</td>
<td>CHAR(1)</td>
<td>SQL delimiter is defined by the DSNZPARM on the system where the logical log was created</td>
</tr>
<tr>
<td>Field name</td>
<td>Offset</td>
<td>Definition</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------</td>
<td>--------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIXED</td>
<td>19</td>
<td>CHAR(1)</td>
<td>Y – mixed DBCS/SBCS allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – mixed DBCS/SBCS not allowed</td>
</tr>
<tr>
<td>DECIMALPOINT</td>
<td>20</td>
<td>CHAR(1)</td>
<td>Decimal point character, either a comma or a period</td>
</tr>
<tr>
<td>EBCDICSCSINGLECCSID</td>
<td>21</td>
<td>CHAR(5)</td>
<td>Default Coded Character Set ID (CCSID) used for single-byte EBCDIC on originating DB2 subsystem</td>
</tr>
<tr>
<td>EBCDICDOUBLEDCCSID</td>
<td>26</td>
<td>CHAR(5)</td>
<td>Default CCSID used for double-byte EBCDIC on originating DB2 subsystem</td>
</tr>
<tr>
<td>EBCDICMIXEDCCSID</td>
<td>31</td>
<td>CHAR(5)</td>
<td>Default CCSID used for mixed EBCDIC on originating DB2 subsystem</td>
</tr>
<tr>
<td>ASCIISINGLECCSID</td>
<td>36</td>
<td>CHAR(5)</td>
<td>Default CCSID used for single-byte ASCII on originating DB2 subsystem</td>
</tr>
<tr>
<td>ASCIIDOUBLECCSID</td>
<td>41</td>
<td>CHAR(5)</td>
<td>Default CCSID used for double-byte ASCII on originating DB2 subsystem</td>
</tr>
<tr>
<td>ASCIIIMIXEDCCSID</td>
<td>46</td>
<td>CHAR(5)</td>
<td>Default CCSID used for mixed ASCII on originating DB2 subsystem</td>
</tr>
<tr>
<td>UNICODESINGLECCSID</td>
<td>51</td>
<td>CHAR(5)</td>
<td>Default CCSID used for single-byte Unicode on originating DB2 subsystem (if DB2 subsystem does not support Unicode, field contains blanks)</td>
</tr>
<tr>
<td>UNICODEDOUBLEDCCSID</td>
<td>56</td>
<td>CHAR(5)</td>
<td>Default CCSID used for double-byte Unicode on originating DB2 subsystem (if DB2 subsystem does not support Unicode, field contains blanks)</td>
</tr>
<tr>
<td>UNICODEMIXEDECCSID</td>
<td>61</td>
<td>CHAR(5)</td>
<td>Default CCSID used for mixed Unicode on originating DB2 subsystem (if DB2 subsystem does not support Unicode, field contains blanks)</td>
</tr>
<tr>
<td>ENCODINGSHEME</td>
<td>66</td>
<td>CHAR(1)</td>
<td>A – ASCII</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E – EBCDIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>U – Unicode</td>
</tr>
<tr>
<td>APPENCODINGSHEME</td>
<td>67</td>
<td>CHAR(1)</td>
<td>A – ASCII</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E – EBCDIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>U – Unicode</td>
</tr>
<tr>
<td>DB2VERSION</td>
<td>68</td>
<td>CHAR(3)</td>
<td>Version of DB2 that the product was running against when the logical log was created. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>910 – Version 9.1.nn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A10 – Version 10.1.nn</td>
</tr>
<tr>
<td>DB2CATALOGMODE</td>
<td>71</td>
<td>CHAR(1)</td>
<td>Migration mode of the DB2 catalog when the logical log was created.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>‘ ’ – migration mode not set (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C – conversion mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E – enabling-new-function mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – new-function mode</td>
</tr>
</tbody>
</table>
Each logical log contains one instance of the Control File ID Record. This record contains the data set name of this control file when the file was created.

Record status: DB2 specific

### Table 29  Type record (XTYP) (part 3 of 3)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
</table>
| UNTRANOBJNAMEINSCAN         | 72     | 48  |   | CHAR(1)    | Y – untranslatable characters encountered in table or column names during log scan  
N – no untranslatable characters encountered |
| UNTRANOBJNAMEINSQNL         | 73     | 49  |   | CHAR(1)    | Y – untranslatable characters present in DB2 object names within generated SQL  
N – no untranslatable characters present |
| UNTRANOBJNAMEINDDDL         | 74     | 4A  |   | CHAR(1)    | Y – untranslatable characters present in generated DDL objects  
N – no untranslatable characters present  
U – unknown (for example, when the product generates only logical log output and the INCLUDEDDL value is Yes) |
| FUNCTIONLEVEL               | 75     | 4B  |   | CHAR(3)    | Function level of the DB2 catalog when the logical log was created. Log Master uses this field to identify DB2 version-specific functionality when you are skipping a version during migration.  
For example:  
910 – Version 9.1.nn  
A10 - Version 10.1.nn |
| (Total Length)              | 78     | 4E  |   |            | n/a |

### Control file ID record (DCNT)

Each logical log contains one instance of the Control File ID Record. This record contains the data set name of this control file when the file was created.

Record status: DB2 specific

### Table 30  Control file id record (DCNT)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00</td>
<td></td>
<td>CHAR(4)</td>
<td>Control record type: DCNT – Control File ID Record (DB2)</td>
</tr>
<tr>
<td>CNTLDSNAME</td>
<td>04</td>
<td>04</td>
<td></td>
<td>CHAR(44)</td>
<td>Original data set name of this control file</td>
</tr>
<tr>
<td>CNTLMEMNAME</td>
<td>48</td>
<td>30</td>
<td></td>
<td>CHAR(8)</td>
<td>Original member name of this control file, if PDS</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>56</td>
<td>38</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
</tbody>
</table>
Data sharing member record (DDSM)

Each logical log contains one instance of the Data Sharing Member Record for each DB2 subsystem that is a member of the data sharing group where the logical log data and control files were created.

Record status: DB2 specific

Table 31  Data sharing member record (DDSM)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DDSM – Data Sharing Member (DB2)</td>
</tr>
<tr>
<td>CNTLGROUPEANME</td>
<td>04 04</td>
<td>CHAR(8)</td>
<td>Data Sharing Group Name</td>
</tr>
<tr>
<td>CNTLMEMBERID</td>
<td>12 0C</td>
<td>CHAR(3)</td>
<td>DB2 member identifier (numeric)</td>
</tr>
<tr>
<td>CNTLMEMBERNAME</td>
<td>15 0F</td>
<td>CHAR(8)</td>
<td>DB2 member name</td>
</tr>
<tr>
<td>CNTLSSID</td>
<td>23 17</td>
<td>CHAR(4)</td>
<td>DB2 subsystem identifier (SSID)</td>
</tr>
<tr>
<td>CNTLCONVERSIONRBA</td>
<td>27 1B</td>
<td>BIN(10)</td>
<td>For the originating subsystem, the MAX RBA FOR TORA value; otherwise, X'000000000000000000000000'.</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>37 15</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

a  BIN(10) RBA is extended format RBA. Non-extended format RBA, BIN(6) is placed at offset 4 (for example, X'123456789ABC' -> X'0000000123456789ABC').

Work ID record (DWRK)

Each logical log contains one instance of the Work ID Record. This record contains information about the work ID that created this logical log.

Record status: DB2 specific

Table 32  Work ID record (DWRK) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DWRK – Work ID Record (DB2)</td>
</tr>
<tr>
<td>WORKIDUSERID</td>
<td>04 04</td>
<td>CHAR(8)</td>
<td>User ID of the Work ID that created this logical log</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>12 0C</td>
<td>CHAR(8)</td>
<td>Work ID name</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>30 1E</td>
<td>CHAR(8)</td>
<td>Job name of creating job</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>38 26</td>
<td>CHAR(5)</td>
<td>JES job number of creating job</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>43 2B</td>
<td>BIN(17)</td>
<td>Creation time of this logical log</td>
</tr>
<tr>
<td></td>
<td>60 3C</td>
<td>BIN(3)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Each logical log contains one instance of the Work ID Description Record. This record type contains a text description of the work ID that created this logical log. You can enter the description through the online interface or by using the WORKID statement. For more information, see “WORKID statement” on page 156.

Record status: DB2 specific

### Table 32 Work ID record (DWRK) (part 2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHASEID</td>
<td>63</td>
<td>3F CHAR(4)</td>
<td>Phase ID from creating the Work ID</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>67</td>
<td>43 CHAR(4)</td>
<td>Step number from creating the Work ID</td>
</tr>
<tr>
<td>FILTERREL</td>
<td>71</td>
<td>47 CHAR(1)</td>
<td>Relationship between multiple filters (if present) A – and O – or</td>
</tr>
<tr>
<td>OVERTIME</td>
<td>72</td>
<td>48 CHAR(1)</td>
<td>Y – overtime execution was used N – current execution mode used</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>73</td>
<td>49 CHAR(7)</td>
<td>Run sequence number</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>80</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

a BIN(17) timestamps are of the format X'YYYYMMDDHHMMSSHHMMNNNNNNPPPPPPPFF00000' where
YYYYMMDD is the date given as year, month, day
HHMMSS is the time given as hour, minute, seconds
HHMMNNNNNNPPPPPPPFF00000 represents fractional seconds and padding

### Work ID description record (DWKD)

Each logical log contains one instance of the Work ID Description Record. This record type contains a text description of the work ID that created this logical log. You can enter the description through the online interface or by using the WORKID statement. For more information, see “WORKID statement” on page 156.

Record status: DB2 specific

### Table 33 Work ID description record (DWKD)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00 CHAR(4)</td>
<td>Control record type: DWKD – Work ID Description Record (DB2)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>04</td>
<td>04 CHAR(64)</td>
<td>Work ID text description</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>68</td>
<td>44 n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Logical log data set record (DLDS)

If Log Master creates an output logical log file as a data set, the control file contains one instance of the Logical Log Data Set Record (DLDS). The DLDS record contains the data set name of the logical log data set when it was created by Log Master. The product initially duplicates this information in the platform-independent Current Path Record. If you move the logical log, be sure to update the Current Path Record to indicate the new location of the logical log data set. Do not update the Logical Log Data Set Record.

Use the SEGMENTED field of this record to determine when the logical log data file contains segmented logical log records. (For more information, see “Normal and segmented logical log record formats” on page 376).

Record status: DB2 specific

### Table 34  Logical log data set record (DLDS) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DLDS – Logical Log Data Set Record (DB2)</td>
</tr>
<tr>
<td>ORIGDATADSNAME</td>
<td>04 04</td>
<td>CHAR(44)</td>
<td>Name of logical log data set when created</td>
</tr>
<tr>
<td>RECORDFORMAT</td>
<td>48 30</td>
<td>CHAR(3)</td>
<td>Logical log file RECFM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VB – variable blocked</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VBS – variable block spanned</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(VBS is tolerated, but ignored in versions later than 2.2.00.)</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>51 33</td>
<td>CHAR(8)</td>
<td>Date/Times in logical log file are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DB2I – DB2 DB2ISO format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SASTIME – SAS format</td>
</tr>
<tr>
<td>EXPANDVAR</td>
<td>59 3B</td>
<td>CHAR(1)</td>
<td>N – VAR fields are not expanded to full length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Y – VAR fields are expanded to full length or padded as required</td>
</tr>
<tr>
<td>SEGMENTED</td>
<td>60 3C</td>
<td>CHAR(1)</td>
<td>N – logical log file does not contain segmented records</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Y – logical log file contains segmented records</td>
</tr>
<tr>
<td>DDLOBJECTS</td>
<td>61 3D</td>
<td>CHAR(1)</td>
<td>N – logical log file does not contain DDL information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Y – logical log file contains DDL information</td>
</tr>
<tr>
<td>LOBSINCLUDED</td>
<td>62 3E</td>
<td>CHAR(1)</td>
<td>N – logical log file does not contain LOB data (column values)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Y – logical log file contains LOB data</td>
</tr>
</tbody>
</table>
Logical log description record (DLUS)

Each logical log contains one instance of the Logical Log Description Record. This record type contains a 64-byte field that holds a text description of this logical log.

Record status: DB2 specific

Table 35  Logical log description record (DLUS)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DLUS – Logical Log Description Record (DB2)</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>04</td>
<td>04</td>
<td>CHAR(64)</td>
<td>User-supplied description of this logical log</td>
<td></td>
</tr>
<tr>
<td>(Total Length)</td>
<td>68</td>
<td>44</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>
Log information record (DLGI)

Each logical log contains one instance of the Log Information Record. This record type contains information about the logical log data file.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00 Dec</td>
<td>Control record type: DLGI – Logical Log Information Record (DB2)</td>
</tr>
<tr>
<td>RECORDCOUNT</td>
<td>04 04 Dec</td>
<td>Number of log records on logical log data file</td>
</tr>
<tr>
<td>BYTECOUNT</td>
<td>14 0E Dec</td>
<td>Number of bytes in logical log data file</td>
</tr>
<tr>
<td>STARTRBA</td>
<td>24 18 Bin</td>
<td>Starting RBA/LRSN for logical log data file</td>
</tr>
<tr>
<td>STARTTIME</td>
<td>36 24 Bin</td>
<td>Starting time for logical log data file</td>
</tr>
<tr>
<td>MAXRECORDSIZE</td>
<td>56 38 Char</td>
<td>Length of longest record in logical log data file (in bytes)</td>
</tr>
</tbody>
</table>

\( ^{a} \) BIN(17) timestamps are of the format: YYYYMMDDHHMMSSHHMNNNPPPFF00000' where
- YYYYMMDD is the date given as year, month, day
- HHMMSS is the time given as hour, minute, seconds
- HHMNNNPPPFF00000 represents fractional seconds and padding

BIN(10) LRSN is extended format LRSN.
- Non-extended format LRSN, BIN(6) is placed at offset 1 (for example, X'CA670FBBF3D3' -> X'00CA670FBBF3D30000000000000').

BIN(10) RBA is extended format RBA.
- Non-extended format RBA, BIN(6) is placed at offset 4 (for example, X'123456789ABC' -> X'00000000123456789ABC').
Log information end record (DLGE)

Each logical log contains one instance of the Log Information End Record. This record type contains information about the end point and RBA/LRSN data in the logical log.

Record Status: DB2 specific

Table 37 Log information end record (DLGE)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DLGE – Logical Log Information End Record (DB2)</td>
<td></td>
</tr>
<tr>
<td>ENDRBA</td>
<td>04</td>
<td>04</td>
<td>BIN(10)</td>
<td>Ending RBA/LRSN for logical log file</td>
<td></td>
</tr>
<tr>
<td>ENDTIME</td>
<td>16</td>
<td>10</td>
<td>BIN(17)</td>
<td>Time that the product finished creating the logical log file</td>
<td></td>
</tr>
<tr>
<td>DEPENDENTRBA</td>
<td>36</td>
<td>24</td>
<td>BIN(10)</td>
<td>The RBA/LRSN of the oldest unit of recovery (lowest RBA/LRSN value) within the current logical log that contains at least one log record that Log Master was unable to complete</td>
<td></td>
</tr>
<tr>
<td>(Total Length)</td>
<td>48</td>
<td>30</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

* BIN(17) timestamps are of the format `X'YYYYMMDDHHMMSSHHMMMNPPPPPPP00000'` where
  `YYYYMMDD` is the date given as year, month, day
  `HHMMSS` is the time given as hour, minute, seconds
  `HHMMMNPPPPPPP00000` represents fractional seconds and padding

BIN(10) LRSN is extended format LRSN.
Non-extended format LRSN, BIN(6) is placed at offset 1 (for example, `X'CA670FBBF3D3'` -> `X'00CA670FBBF3D3000000'`).

BIN(10) RBA is extended format RBA.
Non-extended format RBA, BIN(6) is placed at offset 4 (for example, `X'123456789ABC'` -> `X'0000000123456789ABC'`).
Open transaction record (DOTR)

The logical log contains one instance of the Open Transaction Record for each transaction that was still in process at the end of log processing.

Record Status: DB2 specific

Table 38  Open transaction record (DOTR)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DOTR – Open Transaction Record (DB2)</td>
</tr>
<tr>
<td>URID</td>
<td>04 04</td>
<td>BIN(10)</td>
<td>Unit of recovery identifier</td>
</tr>
<tr>
<td></td>
<td>14 0E</td>
<td>BIN(2)</td>
<td>Reserved</td>
</tr>
<tr>
<td>URIDLRSN</td>
<td>16 10</td>
<td>BIN(10)</td>
<td>Unit of recovery LRSN</td>
</tr>
<tr>
<td></td>
<td>26 1A</td>
<td>BIN(2)</td>
<td>Reserved</td>
</tr>
<tr>
<td>MEMBERID</td>
<td>28 1C</td>
<td>CHAR(3)</td>
<td>Member ID if from a data sharing group</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>31 1F</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

a  BIN(10) LRSN is extended format LRSN.
    Non-extended format LRSN, BIN(6) is placed at offset 1 (for example, X'CA670FBBF3D3' -> X'00CA670FBBF3D30000000').
    BIN(10) RBA is extended format RBA.
    Non-extended format RBA, BIN(6) is placed at offset 4 (for example, X'123456789ABC' -> X'00000000123456789ABC').
Current path record (XPTH)

Each logical log contains at least one instance of the Current Path Record, which contains the current location of the logical log data set. The native operating system of Log Master for DB2 uses only 44 bytes to describe the data set location. To accommodate other platforms and operating systems, the location of the data set can be up to 255 bytes long. (To accommodate the fixed 80-byte requirement for the native version of the logical log control file, the 255 bytes must be divided into multiple XPTH records.)

Record status: Platform independent

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: XPTH – Current Path Record</td>
</tr>
<tr>
<td>SEQNUM</td>
<td>04 04</td>
<td>CHAR(1)</td>
<td>Sequence number of this path record</td>
</tr>
<tr>
<td>NUMRECS</td>
<td>05 05</td>
<td>CHAR(1)</td>
<td>Total number of path records in this control file</td>
</tr>
<tr>
<td>LENGTH</td>
<td>06 06</td>
<td>CHAR(2)</td>
<td>Length of data in following field</td>
</tr>
<tr>
<td>DATA</td>
<td>08 08</td>
<td>CHAR(72)</td>
<td>Path to get to logical log data set (maximum of 255 characters in all records)</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>80 50</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

SysStrings base record (DSS1)

The logical log contains one instance of the SysStrings Base Record for each Coded Character Set ID (CCSID) used during a run of Log Master. The product uses CCSIDs when it encounters a table that matches the filter criteria and that is defined with an Encoding Scheme of ASCII or Unicode. The product includes a separate set of records for the InCcsid and the OutCcsid in the logical log control file.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DSS1 – SysStrings Description Record Part One</td>
</tr>
<tr>
<td>INCCSID</td>
<td>04 04</td>
<td>CHAR(10)</td>
<td>Input Coded Character Set ID</td>
</tr>
<tr>
<td>OUTCCSID</td>
<td>14 0E</td>
<td>CHAR(10)</td>
<td>Output Coded Character Set ID</td>
</tr>
</tbody>
</table>
The logical log contains a set of multiple SysStrings TransTable Records for each pair of SysStrings Base Records (DSS1) that occurs in the logical log. To determine how many records, look at the TRANTABLERECS field in the SysStrings Base Record.

Record status: DB2 specific

### Table 40  SysStrings base record (DSS1) (part 2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSTYPE</td>
<td>24</td>
<td>18</td>
<td>CHAR(2)</td>
<td></td>
<td>Indicates the nature of the conversion</td>
</tr>
<tr>
<td>ERRORBYTENULL</td>
<td>26</td>
<td>1A</td>
<td>CHAR(1)</td>
<td>T – error byte is null</td>
<td>F – error byte is not null</td>
</tr>
<tr>
<td>ERRORBYTE</td>
<td>27</td>
<td>1B</td>
<td>CHAR(1)</td>
<td>The byte used in the conversion table as an error byte</td>
<td></td>
</tr>
<tr>
<td>SUBBYTENULL</td>
<td>28</td>
<td>1C</td>
<td>CHAR(1)</td>
<td>T – sub byte is null</td>
<td>F – sub byte is not null</td>
</tr>
<tr>
<td>SUBBYTE</td>
<td>29</td>
<td>1D</td>
<td>CHAR(1)</td>
<td>The byte used in the conversion table as a substitution character</td>
<td></td>
</tr>
<tr>
<td>TRANSPROC</td>
<td>30</td>
<td>1E</td>
<td>CHAR(8)</td>
<td>Blank or the name of a module to do the translation</td>
<td></td>
</tr>
<tr>
<td>IBMREQ</td>
<td>38</td>
<td>26</td>
<td>CHAR(1)</td>
<td>Whether the row came from the basic machine-readable material tape</td>
<td></td>
</tr>
<tr>
<td>TRANTABLERECS</td>
<td>39</td>
<td>27</td>
<td>CHAR(2)</td>
<td>Number of records used for the translation table</td>
<td></td>
</tr>
<tr>
<td>(Total Length)</td>
<td>41</td>
<td>29</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

### SysStrings transtable record (DSS2)

The logical log contains a set of multiple SysStrings TransTable Records for each pair of SysStrings Base Records (DSS1) that occurs in the logical log. To determine how many records, look at the TRANTABLERECS field in the SysStrings Base Record.

Record status: DB2 specific

### Table 41  SysStrings base record (DSS2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00</td>
<td>CHAR(4)</td>
<td></td>
<td>Control record type: DSS2 – SysStrings Description Record Part Two</td>
</tr>
<tr>
<td>SEQNUM</td>
<td>04</td>
<td>04</td>
<td>CHAR(1)</td>
<td></td>
<td>Sequence number in record group</td>
</tr>
<tr>
<td>NUMRECS</td>
<td>05</td>
<td>05</td>
<td>CHAR(1)</td>
<td></td>
<td>Number of this type of record</td>
</tr>
<tr>
<td>LENGTH</td>
<td>06</td>
<td>06</td>
<td>CHAR(2)</td>
<td></td>
<td>Length of the following field</td>
</tr>
<tr>
<td>TRANSTABLE</td>
<td>08</td>
<td>08</td>
<td>CHAR(72)</td>
<td></td>
<td>Portion of 256-byte table</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>80</td>
<td>50</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>
The logical log contains one instance of the Data Change Header Record for every header field in the data change record of the logical log data set (for more information, see “Logical log data file” on page 369). It describes the format of each header field in the change record to allow a program to process the change records.

Record status: Platform independent

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: XHDF – Data Change Header Description Record</td>
</tr>
<tr>
<td>HEADFIELDNAME</td>
<td>04 04</td>
<td>CHAR(18)</td>
<td>Name of this header field</td>
</tr>
<tr>
<td>HEADFIELDTYPE</td>
<td>22 16</td>
<td>CHAR(4)</td>
<td>Type of data in this field</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BIN – binary / hexadecimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR – character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INT – integer (length is 2 or 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTTM – timestamp value (length is 26)</td>
</tr>
<tr>
<td>HEADFIELDLENGTH</td>
<td>26 1A</td>
<td>CHAR(1)</td>
<td>Format of data within type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>U – unsigned format (BIN subtype)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No subtype exists for CHAR, INT, or DTTM</td>
</tr>
<tr>
<td>HEADFIELDPOS</td>
<td>32 20</td>
<td>CHAR(5)</td>
<td>Offset of this field in the logical log data change record</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>37 25</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
**Table space information record (DTSI)**

The logical log contains one instance of the Table Space Information Record for each table space that has information represented in the logical log. This record type contains information that describes a DB2 table space.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DTSI – Table Space Information Record (DB2)</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>CHAR(4)</td>
<td>Originating database ID (hexadecimal)</td>
</tr>
<tr>
<td>PSID</td>
<td>12</td>
<td>CHAR(4)</td>
<td>Originating table space PSID (hexadecimal, page set descriptor)</td>
</tr>
<tr>
<td>OBJECTTYPE</td>
<td>16</td>
<td>CHAR(2)</td>
<td>Type of object PS – table</td>
</tr>
<tr>
<td>NUMPARTS</td>
<td>18</td>
<td>CHAR(4)</td>
<td>Number of partitions in originating table space (maximum 4096)</td>
</tr>
<tr>
<td>PGSIZE</td>
<td>22</td>
<td>CHAR(2)</td>
<td>Originating table space PGSIZE (4, 8, 16, or 32 KB)</td>
</tr>
<tr>
<td>SEGSIZE</td>
<td>24</td>
<td>CHAR(3)</td>
<td>Originating table space SEGSIZE</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>27</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same table space (the default value is 001)</td>
</tr>
<tr>
<td>LARGE</td>
<td>30</td>
<td>CHAR(1)</td>
<td>L – large space</td>
</tr>
<tr>
<td>DSSIZE</td>
<td>31</td>
<td>CHAR(10)</td>
<td>Data set size value from DB2 catalog table SYSIBM.SYSTABLESPACE (expressed in 1-KB units)</td>
</tr>
<tr>
<td>ENCODINGSHEME</td>
<td>41</td>
<td>CHAR(1)</td>
<td>A – ASCII</td>
</tr>
<tr>
<td>SBCSCCSID</td>
<td>42</td>
<td>CHAR(10)</td>
<td>Single-byte Coded Character Set ID</td>
</tr>
<tr>
<td>DBCSCCSID</td>
<td>52</td>
<td>CHAR(10)</td>
<td>Double-byte Coded Character Set ID</td>
</tr>
<tr>
<td>MIXDCCSID</td>
<td>62</td>
<td>CHAR(10)</td>
<td>Mixed/graphic Coded Character Set ID</td>
</tr>
<tr>
<td>VCATNAMELEN</td>
<td>72</td>
<td>CHAR(3)</td>
<td>Length of originating table space VCAT name</td>
</tr>
<tr>
<td>VCATNAME</td>
<td>75</td>
<td>CHAR(24)</td>
<td>Originating table space VCAT name</td>
</tr>
<tr>
<td>DBNAMELEN</td>
<td>99</td>
<td>CHAR(3)</td>
<td>Length of originating database name</td>
</tr>
<tr>
<td>DBNAME</td>
<td>102</td>
<td>CHAR(24)</td>
<td>Originating database name</td>
</tr>
<tr>
<td>TSNAMELEN</td>
<td>126</td>
<td>CHAR(3)</td>
<td>Length of originating table space name</td>
</tr>
<tr>
<td>TSNAME</td>
<td>129</td>
<td>CHAR(24)</td>
<td>Originating table space name</td>
</tr>
<tr>
<td>UNTRANDBNAMEFLAG</td>
<td>153</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in database name</td>
</tr>
</tbody>
</table>

**Table 43  Table space description record (DTSI) (part 1 of 2)**
Table information record (DTBI)

The logical log contains one instance of the Table Information Record for each table that has change information represented in this logical log. This record type contains information describing a DB2 table.

Record status: DB2 specific

### Table 43  Table space description record (DTSI) (part 2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIDBNAMELEN</td>
<td>154</td>
<td>9A CHAR(3)</td>
<td>Length of Unicode database name</td>
</tr>
<tr>
<td>UNIDBNAME</td>
<td>157</td>
<td>9D CHAR(24)</td>
<td>Unicode database name</td>
</tr>
<tr>
<td>UNTRANSCATNAMEFLAG</td>
<td>181</td>
<td>B5 CHAR(1)</td>
<td>Y – untranslatable characters in table space name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNITSNAMELEN</td>
<td>182</td>
<td>B6 CHAR(3)</td>
<td>Length of Unicode table space name</td>
</tr>
<tr>
<td>UNITSNAME</td>
<td>185</td>
<td>B9 CHAR(24)</td>
<td>Unicode table space name</td>
</tr>
<tr>
<td>UNTRANSCATNAMEFLAG</td>
<td>209</td>
<td>D1 CHAR(1)</td>
<td>Y – untranslatable characters in VCAT name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNIVCATNAMELEN</td>
<td>210</td>
<td>D2 CHAR(3)</td>
<td>Length of Unicode VCAT name</td>
</tr>
<tr>
<td>UNIVCATNAME</td>
<td>213</td>
<td>D5 CHAR(24)</td>
<td>Unicode VCAT name</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>237</td>
<td>ED n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

### Table 44  Table information record (DTBI) (part 1 of 3)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00 CHAR(4)</td>
<td>Control record type: DTBI – Table Information Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>04 CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>08 CHAR(4)</td>
<td>Originating database ID (hexadecimal)</td>
</tr>
<tr>
<td>PSID</td>
<td>12</td>
<td>0C CHAR(4)</td>
<td>Originating table space ID (hexadecimal)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>16</td>
<td>10 CHAR(4)</td>
<td>Originating table ID (hexadecimal)</td>
</tr>
<tr>
<td>OBJECTTYPE</td>
<td>20</td>
<td>14 CHAR(2)</td>
<td>Type of object TB – table</td>
</tr>
<tr>
<td>DB2NUMCOL</td>
<td>22</td>
<td>16 CHAR(3)</td>
<td>Number of columns in originating table</td>
</tr>
<tr>
<td>DB2MAXROWLEN</td>
<td>25</td>
<td>19 CHAR(5)</td>
<td>Maximum row length (in DB2 internal format)</td>
</tr>
<tr>
<td>LLNUMCOL</td>
<td>30</td>
<td>1E CHAR(3)</td>
<td>Number of columns in logical log data record</td>
</tr>
<tr>
<td>LLMAXROWLEN</td>
<td>33</td>
<td>21 CHAR(5)</td>
<td>Maximum row length of data in logical log record</td>
</tr>
</tbody>
</table>
### Table 44  Table information record (DTBI) (part 2 of 3)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ROWTYPE</strong></td>
<td>38</td>
<td>CHAR(1)</td>
<td>V – varying F – fixed length</td>
</tr>
<tr>
<td><strong>SEQUENCENUMBER</strong></td>
<td>39</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001. This sequence number is equal to the sequence number in the other records associated with the definition of this table (DIXI, DCXI, DCLI, DLCI).</td>
</tr>
<tr>
<td><strong>CREATERBA</strong></td>
<td>42</td>
<td>BIN(10) *</td>
<td>Create RBA/LRSN of this instance of the table</td>
</tr>
<tr>
<td><strong>DROPRBA</strong></td>
<td>54</td>
<td>BIN(10) *</td>
<td>Drop RBA/LRSN of this instance of the table</td>
</tr>
<tr>
<td><strong>ENCODINGSHEME</strong></td>
<td>66</td>
<td>CHAR(1)</td>
<td>A – ASCII E – EBCDIC U – Unicode</td>
</tr>
<tr>
<td><strong>EDITPROCNAMELEN</strong></td>
<td>67</td>
<td>CHAR(3)</td>
<td>Length of the edit proc name (zero if none)</td>
</tr>
<tr>
<td><strong>EDITPROCNAME</strong></td>
<td>70</td>
<td>CHAR(24)</td>
<td>Name of edit proc for this table (blanks if none)</td>
</tr>
<tr>
<td><strong>DBNAMELEN</strong></td>
<td>94</td>
<td>CHAR(3)</td>
<td>Length of the database name</td>
</tr>
<tr>
<td><strong>DBNAME</strong></td>
<td>97</td>
<td>CHAR(24)</td>
<td>Database name to which this table belongs</td>
</tr>
<tr>
<td><strong>TSNAMELEN</strong></td>
<td>121</td>
<td>CHAR(3)</td>
<td>Length of the table space name</td>
</tr>
<tr>
<td><strong>TSNAME</strong></td>
<td>124</td>
<td>CHAR(24)</td>
<td>Table space name to which this table belongs</td>
</tr>
<tr>
<td><strong>TABLEOWNERLEN</strong></td>
<td>148</td>
<td>CHAR(3)</td>
<td>Length of the owner portion of the table name</td>
</tr>
<tr>
<td><strong>TABLEOWNER</strong></td>
<td>151</td>
<td>CHAR(128)</td>
<td>Originating table owner name</td>
</tr>
<tr>
<td><strong>SHORTOWNER</strong></td>
<td>279</td>
<td>CHAR(8)</td>
<td>Truncated version of the table owner name (The first eight bytes of the table owner name. This name version is included in records in the logical log data file.)</td>
</tr>
<tr>
<td><strong>TABLENAMELEN</strong></td>
<td>287</td>
<td>CHAR(3)</td>
<td>Length of table name</td>
</tr>
<tr>
<td><strong>TABLENAME</strong></td>
<td>290</td>
<td>CHAR(128)</td>
<td>Table name</td>
</tr>
<tr>
<td><strong>SHORTNAME</strong></td>
<td>418</td>
<td>CHAR(18)</td>
<td>Shortened version of the table name (This version is included in records in the logical log data file (page 371). For more information on how the product generates the shortened name, see page 238.)</td>
</tr>
<tr>
<td><strong>VERSION</strong></td>
<td>436</td>
<td>CHAR(3)</td>
<td>Version number of the table (when the logical log file was generated)</td>
</tr>
<tr>
<td><strong>UNTRANOWNERFLAG</strong></td>
<td>439</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in table owner N – no untranslatable characters present</td>
</tr>
<tr>
<td><strong>UNITABLEOWNERLEN</strong></td>
<td>440</td>
<td>CHAR(3)</td>
<td>Length of original Unicode table owner</td>
</tr>
</tbody>
</table>
Table byte count record (DTBB)

This record contains byte counts for a particular record type (UNDO or REDO), categorized by the type of update (insert, delete, or update).

Record status: DB2 specific

Table 45  Table byte count record (DTBB) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DTBB – Table Byte Count Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08 08</td>
<td>CHAR(4)</td>
<td>Originating Database ID (Hex)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>12 0C</td>
<td>CHAR(4)</td>
<td>Originating Table ID (Hex)</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>16 10</td>
<td>CHAR(1)</td>
<td>U – UNDO R – REDO</td>
</tr>
<tr>
<td>INSERTBYTES</td>
<td>17 11</td>
<td>CHAR(10)</td>
<td>Total bytes in nonRI insert records</td>
</tr>
<tr>
<td>DELETEBYTES</td>
<td>27 1B</td>
<td>CHAR(10)</td>
<td>Total bytes in nonRI delete records</td>
</tr>
<tr>
<td>UPDATEBYTES</td>
<td>37 25</td>
<td>CHAR(10)</td>
<td>Total bytes in nonRI update records</td>
</tr>
<tr>
<td>RIDELETEBYTES</td>
<td>47 2F</td>
<td>CHAR(10)</td>
<td>Total bytes in delete records due to RI</td>
</tr>
</tbody>
</table>
Table record count record (DTBC)

Table 45  Table byte count record (DTBB) (part 2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIUPDATEBYTES</td>
<td>57 39</td>
<td>CHAR(10)</td>
<td>Total bytes in update records due to RI</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>67 43</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table record count record (DTBC)

This record contains counts of records for each record type (UNDO or REDO), categorized by the type of update (insert, update, or delete).

Record status: DB2 specific

Table 46  Table record count record (DTBC)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DTBC – Table Record Count Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08 08</td>
<td>CHAR(4)</td>
<td>Originating Database ID (Hex)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>12 0C</td>
<td>CHAR(4)</td>
<td>Originating Table ID (Hex)</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>16 10</td>
<td>CHAR(1)</td>
<td>U – UNDO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R – REDO</td>
</tr>
<tr>
<td>INSERTCOUNT</td>
<td>17 11</td>
<td>CHAR(10)</td>
<td>Number of nonRI insert records</td>
</tr>
<tr>
<td>DELETECOUNT</td>
<td>27 1B</td>
<td>CHAR(10)</td>
<td>Number of nonRI delete records</td>
</tr>
<tr>
<td>UPDATECOUNT</td>
<td>37 25</td>
<td>CHAR(10)</td>
<td>Number of nonRI update records</td>
</tr>
<tr>
<td>RIDELETECOUNT</td>
<td>47 2F</td>
<td>CHAR(10)</td>
<td>Number of delete records due to RI</td>
</tr>
<tr>
<td>RIUPDATECOUNT</td>
<td>57 39</td>
<td>CHAR(10)</td>
<td>Number of update records due to RI</td>
</tr>
<tr>
<td>TRIGININSERTCOUNT</td>
<td>67 43</td>
<td>CHAR(10)</td>
<td>Number of insert records due to trigger activity</td>
</tr>
<tr>
<td>TRIGDELETECOUNT</td>
<td>77 4D</td>
<td>CHAR(10)</td>
<td>Number of delete records due to trigger activity</td>
</tr>
<tr>
<td>TRIGUPDATECOUNT</td>
<td>87 57</td>
<td>CHAR(10)</td>
<td>Number of update records due to trigger activity</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>97 61</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Table synonym and alias record (DTOB)

This record contains information on any synonyms and aliases for a given table that existed when the logical log file was created. Depending on the synonyms and aliases that exist, there can be zero, one, or multiple DTOB records in the logical log file.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td></td>
<td>Control record type: DTOB – Table Synonym and Alias Record</td>
<td></td>
</tr>
<tr>
<td>SYSID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td></td>
<td>Originating SYSID</td>
<td></td>
</tr>
<tr>
<td>DBID</td>
<td>08 08</td>
<td>CHAR(4)</td>
<td></td>
<td>Database ID (hexadecimal) of table associated with synonym or alias</td>
<td></td>
</tr>
<tr>
<td>PSID</td>
<td>12 0C</td>
<td>CHAR(4)</td>
<td></td>
<td>Table space PSID (hexadecimal, page set descriptor) of table associated with synonym or alias</td>
<td></td>
</tr>
<tr>
<td>OBID</td>
<td>16 10</td>
<td>CHAR(4)</td>
<td></td>
<td>Table ID (hexadecimal) of table associated with synonym or alias</td>
<td></td>
</tr>
<tr>
<td>OBJTYPE</td>
<td>20 14</td>
<td>CHAR(1)</td>
<td></td>
<td>Type of object: S – synonym, A – alias</td>
<td></td>
</tr>
<tr>
<td>OBJCREATORLEN</td>
<td>21 15</td>
<td>CHAR(3)</td>
<td></td>
<td>Length of the creator portion of the synonym or alias name</td>
<td></td>
</tr>
<tr>
<td>OBJCREATOR</td>
<td>24 18</td>
<td>CHAR(128)</td>
<td></td>
<td>Creator of the synonym or alias</td>
<td></td>
</tr>
<tr>
<td>OBJNAMELEN</td>
<td>152 98</td>
<td>CHAR(3)</td>
<td></td>
<td>Length of the synonym or alias name</td>
<td></td>
</tr>
<tr>
<td>OBJNAME</td>
<td>155 9B</td>
<td>CHAR(128)</td>
<td></td>
<td>Name of the synonym or alias</td>
<td></td>
</tr>
<tr>
<td>(Total Length)</td>
<td>283 11B</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DB2 column information record (DCLI)

The logical log contains one instance of the DB2 Column Information Record for each column in every table represented in the logical log. This record type contains information describing the column as it exists in the DB2 table.

Record status: DB2 specific

### Table 48  DB2 column information record (DCLI) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>0</td>
<td>CHAR(4)</td>
<td>Control record type: DCLI – Column Information Record (DB2)</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>CHAR(4)</td>
<td>Originating database ID (hexadecimal)</td>
</tr>
<tr>
<td>PSID</td>
<td>12</td>
<td>CHAR(4)</td>
<td>Originating table space ID (hexadecimal)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>16</td>
<td>CHAR(4)</td>
<td>Originating table ID (hexadecimal)</td>
</tr>
<tr>
<td>DB2COLUMNNUM</td>
<td>20</td>
<td>CHAR(3)</td>
<td>Column number in DB2 row</td>
</tr>
<tr>
<td>DB2COLUMNTYPE</td>
<td>23</td>
<td>CHAR(4)</td>
<td>Column data type in DB2 row:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR – character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DEC – packed decimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INT – integer (length is 2 or 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTTM – timestamp value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FLOT – floating point (length is 4 or 8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VCHR – variable length character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GRPH – graphic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VGRF – variable length graphic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DATE – date (length is 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIME – time (length is 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LVCH – long variable character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LVGR – long variable graphic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLOB – binary large object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CLOB – character large object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBCL – double-byte character large object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ROWI – row ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTTZ – timestamp with time zone</td>
</tr>
<tr>
<td>DB2COLUMNSUBTYPE</td>
<td>27</td>
<td>CHAR(1)</td>
<td>If DB2COLUMNTYPE is CHAR or VCHR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B – for bit data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S – single-byte character set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M – mixed-byte character set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I – inline LOB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DB2COLUMNTYPE is BIGINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D – DOCID column (XML columns)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All other DB2COLUMNTYPE values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>‘ ’ (blank) – not valued</td>
</tr>
<tr>
<td>Field name</td>
<td>Offset</td>
<td>Definition</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DB2COLUMNLENGTH</td>
<td>28</td>
<td>CHAR(5)</td>
<td>The value from the “LENGTH COLUMN” for this Column in the DB2 SYSIBM.SYSCOLUMNS table</td>
</tr>
<tr>
<td>DB2SCALE</td>
<td>33</td>
<td>CHAR(2)</td>
<td>Column scale for decimal data</td>
</tr>
<tr>
<td>DB2NULLS</td>
<td>35</td>
<td>CHAR(1)</td>
<td>Y – nulls allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – not null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>L – nullable large object (LOB)</td>
</tr>
<tr>
<td>FIELDPROC</td>
<td>36</td>
<td>CHAR(24)</td>
<td>Name of field procedure (fieldproc). If no procedure defined, this column is filled with blanks.</td>
</tr>
<tr>
<td>INCLFLAG</td>
<td>60</td>
<td>CHAR(1)</td>
<td>I – this column included in logical log</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E – this column not included in logical log</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>61</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001. This sequence number is equal to the sequence number in the Table Information Record (DTBI) of this table.</td>
</tr>
<tr>
<td>DB2COLUMNCREATEDTS</td>
<td>64</td>
<td>BIN(17)</td>
<td>Timestamp value when this column was created</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>81 51 BIN(9) Reserved</td>
</tr>
<tr>
<td>COLUMNNAMELELEN</td>
<td>90</td>
<td>CHAR(3)</td>
<td>Length of the column name</td>
</tr>
<tr>
<td>COLUMNNAME</td>
<td>93</td>
<td>CHAR(128)</td>
<td>Column name</td>
</tr>
<tr>
<td>VERSION</td>
<td>221</td>
<td>CHAR(3)</td>
<td>Version number of the table containing the column (when the logical log file was generated)</td>
</tr>
<tr>
<td>UNTRANNAMEFLAG</td>
<td>224</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in column name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNICOLUMNNAMELELEN</td>
<td>225</td>
<td>CHAR(3)</td>
<td>Length of original Unicode column name</td>
</tr>
<tr>
<td>UNICOLUMNNAME</td>
<td>228</td>
<td>CHAR(128)</td>
<td>Original Unicode column name</td>
</tr>
<tr>
<td>COLUMNCCSID</td>
<td>356</td>
<td>CHAR(5)</td>
<td>CCSID of column</td>
</tr>
<tr>
<td>FIELDPROLEN</td>
<td>361</td>
<td>CHAR(3)</td>
<td>Length of field procedure name for this column</td>
</tr>
<tr>
<td>UNTRANFIELDPROCFLAG</td>
<td>364</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in field procedure name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNIFIELDPROCLEN</td>
<td>365</td>
<td>CHAR(3)</td>
<td>Length of Unicode field procedure name for this column</td>
</tr>
<tr>
<td>UNIFIELDPROC</td>
<td>368</td>
<td>CHAR(24)</td>
<td>Unicode field procedure name for this column</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>392</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

* BIN(17) timestamps are of the format 'X'YYYYMMDDHHMMSSHHMMMMNNNPPPPFP00000' where YYYYMMDD is the date given as year, month, day HHMMSS is the time given as hour, minute, seconds HHMMMMNNNPPPPFP00000 represents fractional seconds and padding
DB2 column description record part two (DCL2)

The logical log contains one instance of the DB2 Column Description Record (Part Two) for each column contained in the logical log that uses a distinct type (user-defined data type) within DB2. This record type contains information describing the schema and name of the distinct type.

Record status: DB2 specific

Table 49  DB2 column description record part two (DCL2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DCL2 – Column Description Record (DB2) Part Two</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>04</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>08</td>
<td>08</td>
<td>CHAR(4)</td>
<td>Originating database ID (hexadecimal)</td>
</tr>
<tr>
<td>PSID</td>
<td>12</td>
<td>0C</td>
<td>0C</td>
<td>CHAR(4)</td>
<td>Originating table space ID (hexadecimal)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>16</td>
<td>10</td>
<td>10</td>
<td>CHAR(4)</td>
<td>Originating table ID (hexadecimal)</td>
</tr>
<tr>
<td>DB2COLUMNNUM</td>
<td>20</td>
<td>14</td>
<td>14</td>
<td>CHAR(3)</td>
<td>Column number in DB2 row</td>
</tr>
<tr>
<td>TYPESCHEMALEN</td>
<td>23</td>
<td>17</td>
<td>17</td>
<td>CHAR(3)</td>
<td>Length of distinct type schema name</td>
</tr>
<tr>
<td>TYPESCHEMA</td>
<td>26</td>
<td>1A</td>
<td>1A</td>
<td>CHAR(128)</td>
<td>Distinct type schema name</td>
</tr>
<tr>
<td>TYPENAMELEN</td>
<td>154</td>
<td>9A</td>
<td>9A</td>
<td>CHAR(3)</td>
<td>Length of distinct type name</td>
</tr>
<tr>
<td>TYPENAME</td>
<td>157</td>
<td>9D</td>
<td>9D</td>
<td>CHAR(128)</td>
<td>Distinct type name</td>
</tr>
<tr>
<td>UNTRANTYPESCHEMAFLAG</td>
<td>285</td>
<td>11D</td>
<td>11D</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in distinct type schema name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNITYYPESCHEMALEN</td>
<td>286</td>
<td>11E</td>
<td>11E</td>
<td>CHAR(3)</td>
<td>Length of Unicode distinct type schema name</td>
</tr>
<tr>
<td>UNITYYPESCHEMA</td>
<td>289</td>
<td>121</td>
<td>121</td>
<td>CHAR(128)</td>
<td>Unicode distinct type schema name</td>
</tr>
<tr>
<td>UNTRANTYPENAMEFLAG</td>
<td>417</td>
<td>1A1</td>
<td>1A1</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in distinct type name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNITYYPENAMELEN</td>
<td>418</td>
<td>1A2</td>
<td>1A2</td>
<td>CHAR(3)</td>
<td>Length of Unicode distinct type name</td>
</tr>
<tr>
<td>UNITYYPENAME</td>
<td>421</td>
<td>1A5</td>
<td>1A5</td>
<td>CHAR(128)</td>
<td>Unicode distinct type name</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>549</td>
<td>225</td>
<td>225</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Logical log column information record (DLCI)

The logical log contains one instance of the Logical Log Column Information Record for every column actually created in a logical log data change record for one of the tables selected. This record contains information about the location and format of a column as it exists in the logical log data change record. It also contains information about how this column fits into the index key selected for this table.

Record status: DB2 specific

Table 50  Logical log column information record (DLCI) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DLCI – Logical Log Column Information Record (DB2)</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>CHAR(4)</td>
<td>Originating database ID (hexadecimal)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>12</td>
<td>CHAR(4)</td>
<td>Originating table ID (hexadecimal)</td>
</tr>
<tr>
<td>LLCOLUMNNUM</td>
<td>16</td>
<td>CHAR(3)</td>
<td>Column number</td>
</tr>
<tr>
<td>LLCOLUMNTYPE</td>
<td>19</td>
<td>CHAR(4)</td>
<td>Column data type in the logical log</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR – character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DEC – packed decimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INT – integer (length is 2 or 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTTM – timestamp value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FLOT – floating point (length is 4 or 8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VCHR – variable length character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GRPH – graphic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VGRF – variable length graphic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DATE – date (length is 10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIME – time (length is 8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LVCH – long variable character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LVGR – long variable graphic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLOB – binary large object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CLOB – character large object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBCL – double-byte character large object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ROWI – row ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTTZ – timestamp with time zone</td>
</tr>
<tr>
<td>LLCOLUMNLEN</td>
<td>23</td>
<td>CHAR(5)</td>
<td>Length of the column in the logical log file</td>
</tr>
<tr>
<td>LLSCALE</td>
<td>28</td>
<td>CHAR(2)</td>
<td>Column scale for decimal data</td>
</tr>
<tr>
<td>LLNULLS</td>
<td>30</td>
<td>CHAR(1)</td>
<td>Y – nulls allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – not null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>L – nullable large object (LOB)</td>
</tr>
</tbody>
</table>
Logical log column information record (DLCI)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLCOLUMNPOS</td>
<td>31 1F</td>
<td>CHAR(5)</td>
<td>If the logical log data record does not contain a VARCHAR column, this value represents Offset of the current column within the DATA field of the logical log data record. If the current column’s position in the logical log data record is after the first VARCHAR column (or the current column is the first VARCHAR column), this value represents Offset of the current column within the DATA field of the logical log data record. If the current column’s position in the logical log data records is after the first VARCHAR column, the product sets this value to -1.</td>
</tr>
<tr>
<td>LLCOLUMNNSUBTYPE</td>
<td>36 24</td>
<td>CHAR(1)</td>
<td>If LLCOLUMNTYPE is CHAR or VCHR B – for bit data S – single-byte character set M – mixed-byte character set I – inline LOB If DB2COLUMNTYPE is BIGINT D – DOCID column (XML columns) All other LLCOLUMNTYPE values ‘’ – not valued</td>
</tr>
<tr>
<td>KEYSEQ</td>
<td>37 25</td>
<td>CHAR(3)</td>
<td>Key for SQL generation/ reporting by key</td>
</tr>
<tr>
<td>KEYORDERING</td>
<td>40 28</td>
<td>CHAR(1)</td>
<td>A – ascending D – descending</td>
</tr>
<tr>
<td>DB2ROWBYTES</td>
<td>41 29</td>
<td>CHAR(5)</td>
<td>Number of bytes on a DB2 row (encoded/internalized)</td>
</tr>
<tr>
<td>FLDPROCBYTES</td>
<td>46 2E</td>
<td>CHAR(5)</td>
<td>Number of bytes after field proc (decoded/externalized)</td>
</tr>
<tr>
<td>LOGLOGBYTES</td>
<td>51 33</td>
<td>CHAR(5)</td>
<td>Number of bytes on a logical log record</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>56 38</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001. This sequence number is equal to the sequence number in the Table Information Record (DTBI) of this table.</td>
</tr>
<tr>
<td>COLUMNNAMELEN</td>
<td>59 3B</td>
<td>CHAR(3)</td>
<td>Length of the column name</td>
</tr>
<tr>
<td>COLUMNNAME</td>
<td>62 3E</td>
<td>CHAR(128)</td>
<td>Column name</td>
</tr>
<tr>
<td>VERSION</td>
<td>190 B4</td>
<td>CHAR(3)</td>
<td>Version number of the table containing the column (when the logical log file was generated)</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>193 C1</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
</table>

Table 50  Logical log column information record (DLCI) (part 2 of 2)
LOB column description record (DLOB)

The logical log contains one LOB Column Description Record for each large object (LOB) column that occurs in selected log records and for which the product creates at least one LOB VSAM file to contain LOB column values.

Record status: DB2 specific

Table 51  LOB column description record (DLOB) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DLOB – LOB Column Description Record (DB2)</td>
</tr>
<tr>
<td>NOLOGGING</td>
<td>04 04</td>
<td>CHAR(1)</td>
<td>Y – Logging disabled for current LOB table space</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – Logging enabled for current LOB table space</td>
</tr>
<tr>
<td>DBNAME</td>
<td>05 05</td>
<td>CHAR(8)</td>
<td>Name of database that contains auxiliary LOB table space</td>
</tr>
<tr>
<td>TSNAME</td>
<td>13 0D</td>
<td>CHAR(8)</td>
<td>Name of auxiliary LOB table space</td>
</tr>
<tr>
<td>BASETSNAME</td>
<td>21 15</td>
<td>CHAR(8)</td>
<td>Name of base table space that contains current LOB column</td>
</tr>
<tr>
<td>NPART</td>
<td>29 1D</td>
<td>CHAR(5)</td>
<td>Number of partitions in the base table that contains current LOB columns</td>
</tr>
<tr>
<td>DSNUM</td>
<td>34 22</td>
<td>CHAR(5)</td>
<td>Total number of LOB VSAM files created for current LOB column</td>
</tr>
<tr>
<td>PGSIZE</td>
<td>39 27</td>
<td>CHAR(10)</td>
<td>Page size of LOB VSAM files created for current LOB column (4K, 8K, 16K, and so forth)</td>
</tr>
<tr>
<td>DSSIZE</td>
<td>49 31</td>
<td>CHAR(10)</td>
<td>Maximum size (in kilobytes) of LOB VSAM files created for current LOB column</td>
</tr>
<tr>
<td>DBID</td>
<td>59 3B</td>
<td>CHAR(4)</td>
<td>Database identifier (DBID) of database that contains auxiliary LOB table</td>
</tr>
<tr>
<td>PSID</td>
<td>63 3F</td>
<td>CHAR(4)</td>
<td>Page set identifier (PSID) of LOB table space that contains auxiliary table</td>
</tr>
<tr>
<td>OBID</td>
<td>67 43</td>
<td>CHAR(4)</td>
<td>Object identifier (OBID) of auxiliary LOB table</td>
</tr>
<tr>
<td>BASEDBID</td>
<td>71 47</td>
<td>CHAR(4)</td>
<td>Database identifier (DBID) of database that contains base table with current LOB column</td>
</tr>
<tr>
<td>BASEPSID</td>
<td>75 4B</td>
<td>CHAR(4)</td>
<td>Page set identifier (PSID) of table space that contains base table with current LOB column</td>
</tr>
<tr>
<td>BASEOBID</td>
<td>79 4F</td>
<td>CHAR(4)</td>
<td>Object identifier (OBID) of base table with LOB column</td>
</tr>
<tr>
<td>TBCREATOR</td>
<td>83 53</td>
<td>CHAR(128)</td>
<td>Creator of auxiliary LOB table</td>
</tr>
<tr>
<td>TBNAME</td>
<td>211 D3</td>
<td>CHAR(128)</td>
<td>Name of auxiliary LOB table</td>
</tr>
<tr>
<td>COLNAME</td>
<td>339 153</td>
<td>CHAR(128)</td>
<td>Name of current LOB column in base and auxiliary tables</td>
</tr>
<tr>
<td>BASETNAME</td>
<td>467 1D3</td>
<td>CHAR(128)</td>
<td>Name of base table with current LOB column</td>
</tr>
<tr>
<td>BASETBCREATOR</td>
<td>595 253</td>
<td>CHAR(128)</td>
<td>Creator of base table with current LOB column</td>
</tr>
<tr>
<td>NUMROLLS</td>
<td>723 2D3</td>
<td>CHAR(10)</td>
<td>Number of rollback actions that affect current LOB column</td>
</tr>
<tr>
<td>NUMMAPRECS</td>
<td>733 2DD</td>
<td>CHAR(10)</td>
<td>Number of LOB map records encountered for current LOB column</td>
</tr>
<tr>
<td>Field name</td>
<td>Offset</td>
<td>Definition</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NUMDATARECS</td>
<td>743</td>
<td>CHAR(10)</td>
<td>Number of LOB data records encountered for current LOB column</td>
</tr>
<tr>
<td>NUMMAPPAGES</td>
<td>753</td>
<td>CHAR(10)</td>
<td>Number of pages within LOB map records encountered for current LOB column</td>
</tr>
<tr>
<td>NUMDATAPAGES</td>
<td>763</td>
<td>CHAR(10)</td>
<td>Number of pages within LOB data records encountered for current LOB column</td>
</tr>
<tr>
<td>NUMUPDATES</td>
<td>773</td>
<td>CHAR(10)</td>
<td>Number of updates to current LOB column</td>
</tr>
<tr>
<td>NUMINSERTS</td>
<td>783</td>
<td>CHAR(10)</td>
<td>Number of inserts to current LOB column</td>
</tr>
<tr>
<td>NUMDELETEs</td>
<td>793</td>
<td>CHAR(10)</td>
<td>Number of deletes to current LOB column</td>
</tr>
<tr>
<td>CREATETIMESTAMP</td>
<td>839</td>
<td>BIN(17)</td>
<td>DB2 timestamp value when auxiliary LOB table was created</td>
</tr>
<tr>
<td>ALTERTIMESTAMP</td>
<td>859</td>
<td>BIN(17)</td>
<td>Timestamp when auxiliary LOB table was altered</td>
</tr>
<tr>
<td>DROPPEDTIMESTAMP</td>
<td>879</td>
<td>BIN(17)</td>
<td>Timestamp when auxiliary LOB table was dropped</td>
</tr>
<tr>
<td>TABCONCREATERBA</td>
<td>899</td>
<td>BIN(10)</td>
<td>RBA when base table that contains current LOB column was created</td>
</tr>
<tr>
<td>OTSEQNUM</td>
<td>911</td>
<td>CHAR(3)</td>
<td>Overtime sequence number (if applicable)</td>
</tr>
<tr>
<td>MAXCOLLENGTH</td>
<td>914</td>
<td>CHAR(10)</td>
<td>Maximum column length</td>
</tr>
<tr>
<td>INLINELENGTH</td>
<td>915</td>
<td>CHAR(10)</td>
<td>Inline column length</td>
</tr>
<tr>
<td>INLINELOB</td>
<td>925</td>
<td>CHAR(1)</td>
<td>Y – The LOB is defined as INLINE  N – The LOB is not defined as INLINE</td>
</tr>
<tr>
<td>V10FORMAT</td>
<td>926</td>
<td>CHAR(1)</td>
<td>Y – The LOB is in DB2 Version 10 format  N – The LOB is not in DB2 Version 10 format</td>
</tr>
<tr>
<td>UTSDEFINED</td>
<td>927</td>
<td>CHAR(1)</td>
<td>Y – The LOB is defined in a universal table space  N – The LOB is not defined in a universal table space</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>928</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Chapter 4 Logical log files

XML column description record (DXML)

The logical log contains one XML Column Description Record for each XML column that occurs in selected log records and for which the product creates at least one XML VSAM file to contain data from an XML column.

Record status: DB2 specific

Table 52 XML column description record (DXML) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DXML – XML Column Description Record (DB2)</td>
</tr>
<tr>
<td>NOLOGGING</td>
<td>04 04</td>
<td>CHAR(1)</td>
<td>Y – Logging disabled for current XML table space</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N – Logging enabled for current XML table space</td>
</tr>
<tr>
<td>DBNAME</td>
<td>05 05</td>
<td>CHAR(8)</td>
<td>Name of database that contains XML table space</td>
</tr>
<tr>
<td>TSNAME</td>
<td>13 0D</td>
<td>CHAR(8)</td>
<td>Name of XML table space</td>
</tr>
<tr>
<td>BASETSNAME</td>
<td>21 15</td>
<td>CHAR(8)</td>
<td>Name of table space that contains current XML column</td>
</tr>
<tr>
<td>NPART</td>
<td>29 1D</td>
<td>CHAR(5)</td>
<td>Number of partitions in the table that contains current XML columns</td>
</tr>
<tr>
<td>DSNUM</td>
<td>34 22</td>
<td>CHAR(5)</td>
<td>Total number of XML VSAM files created for current XML column</td>
</tr>
<tr>
<td>PGSIZE</td>
<td>39 27</td>
<td>CHAR(10)</td>
<td>Page size of XML VSAM files created for current XML column (4K, 8K, 16K, and so forth)</td>
</tr>
<tr>
<td>DSSIZE</td>
<td>49 31</td>
<td>CHAR(10)</td>
<td>Maximum size (in kilobytes) of XML VSAM files created for current XML column</td>
</tr>
<tr>
<td>DBID</td>
<td>59 3B</td>
<td>CHAR(4)</td>
<td>Database identifier (DBID) of database that contains XML table</td>
</tr>
<tr>
<td>PSID</td>
<td>63 3F</td>
<td>CHAR(4)</td>
<td>Page set identifier (PSID) of XML table space that contains table</td>
</tr>
<tr>
<td>OBID</td>
<td>67 43</td>
<td>CHAR(4)</td>
<td>Object identifier (OBID) of XML table</td>
</tr>
<tr>
<td>BASEDBID</td>
<td>71 47</td>
<td>CHAR(4)</td>
<td>Database identifier (DBID) of database that contains table with current XML column</td>
</tr>
<tr>
<td>BASEPSID</td>
<td>75 4B</td>
<td>CHAR(4)</td>
<td>Page set identifier (PSID) of table space that contains table with current XML column</td>
</tr>
</tbody>
</table>

a. BIN(17) timestamps are of the format ‘YYYYMMDDHHMMSSHHMMNNPPPPPPPPPPP000000’ where
   YYYYMMDD is the date given as year, month, day
   HHMMSS is the time given as hour, minute, seconds
   HHMMNNNPPPPPPPPPPP000000 represents fractional seconds and padding

BIN(10) LRSN is extended format LRSN.
Non-extended format LRSN, BIN(6) is placed at offset 1 (for example, ‘X’CA670FBBF3D3’ -> ‘X’00CA670FBBF3D3000000’).
BIN(10) RBA is extended format RBA.
Non-extended format RBA, BIN(6) is placed at offset 4 (for example, ‘X’123456789ABC’ -> ‘X’0000000123456789ABC’).
<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASEOBID</td>
<td>79</td>
<td>Object identifier (OBID) of table with XML column</td>
</tr>
<tr>
<td>TBCREATOR</td>
<td>83</td>
<td>Creator of XML table</td>
</tr>
<tr>
<td>TBNAME</td>
<td>211</td>
<td>Name of XML table</td>
</tr>
<tr>
<td>COLNAME</td>
<td>339</td>
<td>Name of current XML column in table that contains XML</td>
</tr>
<tr>
<td>BASETBNAMEx</td>
<td>467</td>
<td>Name of table with current XML column</td>
</tr>
<tr>
<td>BASETBCREATOR</td>
<td>595</td>
<td>Creator of table with current XML column</td>
</tr>
<tr>
<td>NUMROLLS</td>
<td>723</td>
<td>Number of rollback actions that affect current XML column</td>
</tr>
<tr>
<td>NUMMAPRECS</td>
<td>733</td>
<td>Number of map records encountered for current XML column</td>
</tr>
<tr>
<td>NUMMATAECS</td>
<td>743</td>
<td>Number of data records encountered for current XML column</td>
</tr>
<tr>
<td>NUMMAPPAGEs</td>
<td>753</td>
<td>Number of pages within map records encountered for current XML column</td>
</tr>
<tr>
<td>NUMDATAPAGEs</td>
<td>763</td>
<td>Number of pages within data records encountered for current XML column</td>
</tr>
<tr>
<td>NUMUPDATES</td>
<td>773</td>
<td>Number of updates to current XML column</td>
</tr>
<tr>
<td>NUMINSERTS</td>
<td>783</td>
<td>Number of inserts to current XML column</td>
</tr>
<tr>
<td>NUMDELETES</td>
<td>793</td>
<td>Number of deletes to current XML column</td>
</tr>
<tr>
<td>CREATeRBA</td>
<td>803</td>
<td>Relative byte address (RBA) when XML table was created</td>
</tr>
<tr>
<td>ALTERRBA</td>
<td>815</td>
<td>RBA when XML table was altered</td>
</tr>
<tr>
<td>DROPPEDRBA</td>
<td>827</td>
<td>RBA when XML table was dropped</td>
</tr>
<tr>
<td>CREATeTIMESTAMP</td>
<td>839</td>
<td>DB2 timestamp value when XML table was created</td>
</tr>
<tr>
<td>ALTERTIMESTAMP</td>
<td>859</td>
<td>Timestamp when XML table was altered</td>
</tr>
<tr>
<td>DROPPEDTIMESTAMP</td>
<td>879</td>
<td>Timestamp when XML table was dropped</td>
</tr>
<tr>
<td>TABCONCREATORBA</td>
<td>899</td>
<td>RBA when table that contains current XML column was created</td>
</tr>
<tr>
<td>OTSEQNUM</td>
<td>911</td>
<td>Overtime sequence number (if applicable)</td>
</tr>
<tr>
<td>MAXCOLLENGTH</td>
<td>914</td>
<td>Maximum column length</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>924</td>
<td></td>
</tr>
</tbody>
</table>
XMLSTRING data set name record (DXSF)

The logical log contains only one XMLSTRING Data Set Name Record. This record defines the name of the logical log XMLSTRING control file. This file contains XMLSTRING data for all XML columns included in the logical log file. The product creates this record only when a job or job step generates a logical log output file and the XMLSTRING keyword specifies a data set name.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DXSF – XMLSTRING Data Set Name Record (DB2)</td>
</tr>
<tr>
<td>DSNNAME</td>
<td>04</td>
<td>04</td>
<td>CHAR(44)</td>
<td>Data set name of file containing XML string data for this logical log file (data obtained from DB2 catalog table SYSIBM.SYSXMLSTRINGS)</td>
</tr>
</tbody>
</table>

(Total Length) 48 30 n/a n/a

LOB/XML cluster data set record (CLUS)

The logical log contains one LOB/XML Cluster Data Set Record for each VSAM file that Log Master creates to store data from large volume columns (such as LOB columns or XML columns).

Record status: DB2 specific
The logical log contains one instance of the Index Information Record for each selected index associated with a Table that has change information represented in this logical log. This record type contains information describing a DB2 index.

Table 54  LOB/XML cluster data set record (CLUS)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00 00</td>
<td>CHAR(4)</td>
<td>Control record type: CLUS – LOB/XML Cluster Data Set Record (DB2)</td>
</tr>
<tr>
<td>DBID</td>
<td>04 04 04</td>
<td>CHAR(4)</td>
<td>Database identifier (DBID) of database that contains large volume table space</td>
</tr>
<tr>
<td>PSID</td>
<td>08 08 08</td>
<td>CHAR(4)</td>
<td>Page set identifier (PSID) of LOB table space that contains large volume table space</td>
</tr>
<tr>
<td>OBID</td>
<td>12 0C 0C 0C</td>
<td>CHAR(4)</td>
<td>Object identifier (OBID) of large volume table space</td>
</tr>
<tr>
<td>BASEDBID</td>
<td>16 10 0C 0C</td>
<td>CHAR(4)</td>
<td>Database identifier (DBID) of database that contains table with large volume column</td>
</tr>
<tr>
<td>BASEPSID</td>
<td>20 14 14 14</td>
<td>CHAR(4)</td>
<td>Page set identifier (PSID) of table space that contains table with large volume column</td>
</tr>
<tr>
<td>BASEOBID</td>
<td>24 18 18 18</td>
<td>CHAR(4)</td>
<td>Object identifier (OBID) of table with large volume column</td>
</tr>
<tr>
<td>CREATERRBA</td>
<td>28 1C 26</td>
<td>BIN(10)</td>
<td>Relative byte address (RBA) when large volume table space was created.</td>
</tr>
<tr>
<td></td>
<td>38 26 26</td>
<td>BIN(2)</td>
<td>Reserved</td>
</tr>
<tr>
<td>DBID</td>
<td>40 28 28 28</td>
<td>CHAR(5)</td>
<td>Database identifier (DBID) of database that contains large volume table space</td>
</tr>
<tr>
<td>PSID</td>
<td>59 59 59</td>
<td>CHAR(44)</td>
<td>Page set identifier (PSID) of LOB table space that contains large volume table space</td>
</tr>
<tr>
<td>OBID</td>
<td>85 85 85</td>
<td>CHAR(44)</td>
<td>Object identifier (OBID) of large volume table space</td>
</tr>
<tr>
<td>BASEDBID</td>
<td>133 85 85 85</td>
<td>CHAR(44)</td>
<td>Database identifier (DBID) of database that contains table with large volume column</td>
</tr>
<tr>
<td>BASEPSID</td>
<td>177 177 177</td>
<td>CHAR(5)</td>
<td>Page set identifier (PSID) of table space that contains table with large volume column</td>
</tr>
<tr>
<td>BASEOBID</td>
<td>182 182 182</td>
<td>CHAR(5)</td>
<td>Object identifier (OBID) of table with large volume column</td>
</tr>
<tr>
<td>CREATERRBA</td>
<td>187 187 187</td>
<td>BIN(10)</td>
<td>Relative byte address (RBA) when large volume table space was created.</td>
</tr>
<tr>
<td></td>
<td>192 192 192</td>
<td>BIN(2)</td>
<td>Reserved</td>
</tr>
<tr>
<td>TYPE</td>
<td>195 195 195</td>
<td>CHAR(3)</td>
<td>Type of large-volume column file LOB – large object (LOB) column XML – XML column</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>198 C6 n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

a  BIN(10) LRSN is extended format LRSN.
Non-extended format LRSN, BIN(6) is placed at offset 1 (for example, X‘CA670FBBF3D3’ -> X‘00CA670FBBF3D30000000’).
BIN(10) RBA is extended format RBA.
Non-extended format RBA, BIN(6) is placed at offset 4 (for example, X‘123456789ABC’ -> X‘00000000123456789ABC’).
Record status: DB2 specific

### Table 55  Index description record (DIXI)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DIXI – Index Information Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08 08</td>
<td>CHAR(4)</td>
<td>Originating DBID (hexadecimal)</td>
</tr>
<tr>
<td>PSID</td>
<td>12 0C</td>
<td>CHAR(4)</td>
<td>Originating Index PSID (hexadecimal)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>16 10</td>
<td>CHAR(4)</td>
<td>Originating Table ID (hexadecimal)</td>
</tr>
<tr>
<td>NUMCOLSINKEY</td>
<td>20 14</td>
<td>CHAR(3)</td>
<td>Number of columns in the index key</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>23 17</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001. This sequence number is equal to the sequence number in the other records associated with the definition of this table (DTBI, DCXI).</td>
</tr>
<tr>
<td>TYPE</td>
<td>26 1A</td>
<td>CHAR(1)</td>
<td>Index type from SYSIBM.SYSINDEXES</td>
</tr>
<tr>
<td>IXCREATORLEN</td>
<td>27 1B</td>
<td>CHAR(3)</td>
<td>Length of the index creator name</td>
</tr>
<tr>
<td>IXCREATOR</td>
<td>30 1E</td>
<td>CHAR(128)</td>
<td>Originating index creator name</td>
</tr>
<tr>
<td>IXNAMELEN</td>
<td>158 9E</td>
<td>CHAR(3)</td>
<td>Length of the index name</td>
</tr>
<tr>
<td>IXNAME</td>
<td>161 A1</td>
<td>CHAR(128)</td>
<td>Originating Index Name</td>
</tr>
<tr>
<td>UNTRANIXCREATORFLAG</td>
<td>289 121</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in index creator name N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNIIXCREATORLEN</td>
<td>290 122</td>
<td>CHAR(3)</td>
<td>Length of Unicode index creator name</td>
</tr>
<tr>
<td>UNIIXCREATOR</td>
<td>293 125</td>
<td>CHAR(128)</td>
<td>Unicode index creator name</td>
</tr>
<tr>
<td>UNTRANIXNAMEFLAG</td>
<td>421 1A5</td>
<td>CHAR(1)</td>
<td>Y – untranslatable characters in index name N – no untranslatable characters present</td>
</tr>
<tr>
<td>UNIIXNAMELEN</td>
<td>422 1A6</td>
<td>CHAR(3)</td>
<td>Length of Unicode index name</td>
</tr>
<tr>
<td>UNIIXNAME</td>
<td>425 1A9</td>
<td>CHAR(128)</td>
<td>Unicode index name</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>553 229</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Index column description record (DICL)

The logical log contains at least one Index Column Description Record for each Index Description Record in the logical log control file. Additional records are used if there are more than 15 key columns. This record type contains mapping information for the columns that are part of the associated index. A single Index Column Description record can describe up to 14 columns of an index. The maximum number of columns for a DB2 index currently is 64.

Record status: DB2 specific

Table 56  Index column description record (DICL)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DICL – Index Description Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>CHAR(4)</td>
<td>Originating DBID (hex)</td>
</tr>
<tr>
<td>PSID</td>
<td>12</td>
<td>CHAR(4)</td>
<td>Originating Index PSID (hex)</td>
</tr>
<tr>
<td>IXCOLRECNUM</td>
<td>16</td>
<td>CHAR(2)</td>
<td>Record Number of this DICL record (01-05)</td>
</tr>
<tr>
<td>IXCOLRECKEYCOUNT</td>
<td>18</td>
<td>CHAR(2)</td>
<td>Number of entries of columns mapped on this DICL record</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>20</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001.</td>
</tr>
<tr>
<td>DICLKEYINFO</td>
<td>23</td>
<td>VARIABLE</td>
<td>IXCOLRECKEYCOUNT entries of the following layout:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KEYCOL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KEYCOLORDER</td>
</tr>
</tbody>
</table>

Clustering index information record (DCXI)

This record describes the clustering index format of a partitioned table. The record’s layout is similar to the Index Information Record, except that the record type is DCXI instead of DIXI.

Record status: DB2 specific

Table 57  Clustering index information record (DCXI) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DCXI – Clustering Index Information Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
</tbody>
</table>
This record describes the columns in a clustering index. The record’s layout is identical to the Index Column Description Record, except that the record type is DCCL instead of DICL.

Record status: DB2 specific

### Table 57  Clustering index information record (DCXI) (part 2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBID</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Originating DBID (hexadecimal)</td>
</tr>
<tr>
<td>PSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating PSID (hexadecimal)</td>
</tr>
<tr>
<td>TBOBID</td>
<td>16</td>
<td>CHAR(4)</td>
<td>Originating Table ID (hexadecimal)</td>
</tr>
<tr>
<td>NUMCOLSINKEY</td>
<td>20</td>
<td>CHAR(3)</td>
<td>Number of columns in the index key</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>23</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001. This sequence number is equal to the sequence number in the other records associated with the definition of this table (DTBI, DIXI).</td>
</tr>
<tr>
<td>TYPE</td>
<td>26</td>
<td>CHAR(1)</td>
<td>Index type from SYSIBM.SYSINDEXES</td>
</tr>
<tr>
<td>IXCREATORLEN</td>
<td>27</td>
<td>CHAR(3)</td>
<td>Length of the index creator name</td>
</tr>
<tr>
<td>IXCREATOR</td>
<td>30</td>
<td>CHAR(128)</td>
<td>Originating index creator name</td>
</tr>
<tr>
<td>IXNAMELEN</td>
<td>158</td>
<td>CHAR(3)</td>
<td>Length of the index name</td>
</tr>
<tr>
<td>IXNAME</td>
<td>161</td>
<td>CHAR(128)</td>
<td>Originating index name</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>289</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

### Clustering index column description record (DCCL)

This record describes the columns in a clustering index. The record’s layout is identical to the Index Column Description Record, except that the record type is DCCL instead of DICL.

Record status: DB2 specific

### Table 58  Clustering index column description record (DCCL) (part 1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DCCL – Index Description Record</td>
</tr>
<tr>
<td>SYSID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Originating SYSID</td>
</tr>
<tr>
<td>DBID</td>
<td>08</td>
<td>CHAR(4)</td>
<td>Originating DBID (hex)</td>
</tr>
<tr>
<td>PSID</td>
<td>12</td>
<td>CHAR(4)</td>
<td>Originating Index PSID (hex)</td>
</tr>
<tr>
<td>IXCOLRECNUM</td>
<td>16</td>
<td>CHAR(2)</td>
<td>Record Number of this DCCL record (01 - 05)</td>
</tr>
<tr>
<td>IXCOLRECKEYCOUNT</td>
<td>18</td>
<td>CHAR(2)</td>
<td>Number of entries of columns mapped on this DCCL record</td>
</tr>
<tr>
<td>SEQUENCENUMBER</td>
<td>20</td>
<td>CHAR(3)</td>
<td>Sequence number to distinguish multiple versions of the same database object. The default value is 001.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Filter ID record (DFID)

The logical log contains one instance of the Filter ID Record for each filter used to extract the logical log records that are in the logical log. This record type connects a filter with a numeric ID.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICLKEYINFO</td>
<td>23 17</td>
<td>VARIABLE</td>
<td>INCOLRECKEYCOUNT entries of the following layout:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KEYCOL CHAR(3) Column Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KEYCOLORDER CHAR(1) A – ascending</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D – descending</td>
</tr>
</tbody>
</table>

Filter anchor record (DFIL)

The logical log contains one instance of the Filter Anchor Record for every filter used to extract the logical log records that are in the logical log. This record type provides general information about this filter.

Record status: DB2 specific
The logical log contains one instance of the Filter Description Record for every filter used to extract the logical log records that are in this logical log. This record type provides a text description of the filter.

Record status: DB2 specific

### Table 60  Filter anchor record (DFIL)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DFIL – Filter Anchor Record (DB2)</td>
</tr>
<tr>
<td>FILTERID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td>Numeric filter ID</td>
</tr>
<tr>
<td>CREATEDBY</td>
<td>08 08</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the individual creating this filter</td>
</tr>
<tr>
<td>CREATEDTIMESTAMP</td>
<td>16 10</td>
<td>CHAR(20)</td>
<td>Date/time of creation of this filter</td>
</tr>
<tr>
<td>LASTUPDATEDBY</td>
<td>36 24</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the last user to update this filter</td>
</tr>
<tr>
<td>UPDATEDTIMESTAMP</td>
<td>44 2C</td>
<td>CHAR(20)</td>
<td>Date/time of last update</td>
</tr>
<tr>
<td>FILTERTYPE</td>
<td>64 40</td>
<td>CHAR(1)</td>
<td>S – structured fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F – freeform text</td>
<td></td>
</tr>
<tr>
<td>(Total Length)</td>
<td>65 41</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

### Filter description record (DFDS)

The logical log contains one instance of the Filter Description Record for every filter used to extract the logical log records that are in this logical log. This record type provides a text description of the filter.

Record status: DB2 specific

### Table 61  Filter description record (DFDS)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DFDS – Filter Description Record</td>
</tr>
<tr>
<td>FILTERID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td>Filter ID</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>08 08</td>
<td>CHAR(64)</td>
<td>User-supplied description of filter</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>72 48</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Filter line record (DFLN)

The logical log contains one instance of the Filter Line Record for every line of every filter used to extract the logical log records that are in the logical log. If you entered the filter as structured fields, the logical log control file contains one of these records for every structured selection criteria. If you entered the filter as free-form text, the logical log control file contains one of these records for every line of text.

Record status: DB2 specific

### Table 62  Filter line record (DFLN)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00 00</td>
<td>CHAR(4)</td>
<td>Control record type: DFLN – Filter Line Record</td>
</tr>
<tr>
<td>FILTERID</td>
<td>04 04</td>
<td>CHAR(4)</td>
<td>Filter ID</td>
</tr>
<tr>
<td>GROUPNUM</td>
<td>08 08</td>
<td>CHAR(5)</td>
<td>Structured fields, which group this belongs to</td>
</tr>
<tr>
<td>LINENUM</td>
<td>13 0D</td>
<td>CHAR(5)</td>
<td>Structured fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Free-form text</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>line number within the filter</td>
</tr>
<tr>
<td>LINETYPE</td>
<td>18 12</td>
<td>CHAR(8)</td>
<td>Type of information in Line Data records:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WHERECLS – where clause line</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CORRID – correlation ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>JOBNAME – job name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PLANNAME – plan name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>URID – unit of recovery identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>USERID – user ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CONNID – connection ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBNAME – database name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TSPACE – table space name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DBID – database ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PSID – table space ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBUSER – owner ID for table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBNAME – table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COLVAL – column value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COLCHANGE – column value changed</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>26 1A</td>
<td>CHAR(8)</td>
<td>Comparison operator used in structured fields:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= equal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>^= or &lt;&gt; not equal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; greater than</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt; less than</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt;= greater than or equal to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;= less than or equal to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LIKE string comparison</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHANGED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NOTNULL</td>
</tr>
</tbody>
</table>

(Total Length) 34 22 n/a n/a
Filter line data record (DLIN)

There are one to four instances of the Filter Line Data Record for every line of every filter used to extract the logical log records that are in the logical log. If you entered the filter as structured fields, this set of records has the structured information. If you entered the filter as free form text, the logical log control file contains one set of records for each line of each filter.

Record status: DB2 specific

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTLRECORDTYPE</td>
<td>00</td>
<td>CHAR(4)</td>
<td>Control record type: DLIN – Filter Line Data Record</td>
</tr>
<tr>
<td>FILTERID</td>
<td>04</td>
<td>CHAR(4)</td>
<td>Filter ID</td>
</tr>
<tr>
<td>SEQNUM</td>
<td>08</td>
<td>CHAR(1)</td>
<td>Sequence number of this line data record (for this line)</td>
</tr>
<tr>
<td>NUMREC</td>
<td>09</td>
<td>CHAR(1)</td>
<td>Number of line data records for this line</td>
</tr>
<tr>
<td>LINENUM</td>
<td>10</td>
<td>CHAR(5)</td>
<td>Structured fields – line within the group</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Free-form text – line number</td>
</tr>
<tr>
<td>LENGTH</td>
<td>15</td>
<td>CHAR(2)</td>
<td>Length of the data in the following field</td>
</tr>
<tr>
<td>LINEDATA</td>
<td>17</td>
<td>CHAR(63)</td>
<td>Line data information, a maximum of 254 characters per line</td>
</tr>
<tr>
<td>(Total Length)</td>
<td>80</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Logical log data file

The logical log data file contains the actual database changes, such as update, delete, and insert actions, for that part of the DB2 log defined (directly or indirectly) by your selection criteria.

The format of the logical log data file does not define an order for the records that the file contains. Normally the records in the data file are not in the same order as the original database changes reflected in DB2 log records.

Application programs that process a logical log data file might need to sort the records in the file before processing. For example, to duplicate the original order for units of recovery and the order of the changes within each unit of recovery, use the UORCOMMITLRSN and ORDERVALUE fields as sort keys. (If your logical log file contains segmented logical log records, be sure to account for multiple segments by using the SEGNUM field of the logical log header. For more information see “Normal and segmented logical log record formats” on page 376.)
Data change record (LLDF)

The logical log contains one instance of the Data Change Record for every change represented in the log that was defined (directly or indirectly) by your selection criteria. The basic layout of the Data Change Record is shown in Figure 47 on page 377. The record contains the following parts:

- logical log header

  The header contains unit of recovery information, object identifiers, timestamp, and other information about the data change represented in the record. The layout of the logical log header is described in Table 64 on page 371. The logical log control file also defines the layout of the logical log header in a set of description records. For more information, see “Data change header description record (XHDF)” on page 345).

**NOTE**

The length of the logical log header can change in different versions of the product. If you process output logical log files with a program other than Log Master, use the LENGTH field at the beginning of each logical log record to determine the length of the header. By using this technique, you can avoid changing your programs or processes when the length of the header changes.

- row image

  The row image begins with a two-byte length value that defines the length of the succeeding row image. The content of the row image varies depending on the type of change that the Data Change Record represents:

  — insert action (the content is an image of inserted row)
  — delete action (the content is an image of deleted row)
  — update action (the content is an image of the row before update is completed)

  Within a row image, Log Master places columns in order based on their logical position within the base table, regardless of their physical position within the table row. The column order is based on the COLNO column in the SYSIBM.SYSCOLUMNS table of the DB2 catalog.

- additional row image (when required)

  If the Data Change Record represents an update action, the record includes another row image that contains the after image of the row. The additional row image starts with a two-byte length field that begins immediately after the update action’s before image.
Records in the logical log data file can be segmented, depending on the length of the data represented in the record and the system-determined block size of the device where Log Master writes the logical log. For more information about how the product segments logical log records, see “Normal and segmented logical log record formats” on page 376.

Record status: DB2 specific

**NOTE**

Table 64 is updated for Log Master version 11.1 and later. However, Log Master allows for processing of earlier versions (back to version 7.3.00) of the logical log with normalization performed as needed to work with the new format.

In addition, Log Master version 11.1 and later supports extended precision of timestamp values.

### Table 64  Data change record (LLDF) (part 1 of 4)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENGTH</td>
<td>00 00</td>
<td>BIN(2)</td>
<td>Length of the header</td>
</tr>
<tr>
<td>SYSTEMID</td>
<td>02 02</td>
<td>CHAR(4)</td>
<td>Originating DB2 system</td>
</tr>
<tr>
<td>DBID</td>
<td>06 06</td>
<td>BIN(2)</td>
<td>Database ID of the object that the record is for</td>
</tr>
<tr>
<td>PSID</td>
<td>08 08</td>
<td>BIN(2)</td>
<td>Table Space ID of the object that the record is for</td>
</tr>
<tr>
<td>TBOBID</td>
<td>10 0A</td>
<td>BIN(2)</td>
<td>Table OBID</td>
</tr>
<tr>
<td>TBOWNERLEN</td>
<td>12 0C</td>
<td>BIN(2)</td>
<td>Length of the TABLEOWNER field</td>
</tr>
<tr>
<td>TBNAMELEN</td>
<td>14 0E</td>
<td>BIN(2)</td>
<td>Length of the TABLENAME field</td>
</tr>
<tr>
<td>DBNAME</td>
<td>16 10</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>TSNAME</td>
<td>24 18</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>TABLEOWNER</td>
<td>32 20</td>
<td>CHAR(8)</td>
<td>Owner ID for the table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When TBOWNERLEN (page 371) &lt;= 8 actual owner name of the table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When TBOWNERLEN &gt; 8 a truncated version of the owner name (the first 8 bytes),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>corresponds to the SHORTOWNER field of the table’s DTBI record in the logical log control file (page 348)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information about the shortened owner name, see page 238.</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>40 28</td>
<td>CHAR(18)</td>
<td>Table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If TBNAMELEN (page 371) &lt;= 18 actual table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If TBNAMELEN &gt; 18 a unique, shortened version of the table name that</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>corresponds to the SHORTNAME field of the table’s DTBI record in the logical log control file (page 348)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information about the shortened owner name, see page 238.</td>
</tr>
<tr>
<td>PARTNUM</td>
<td>58 3A</td>
<td>BIN(2)</td>
<td>Partition number (if table space)</td>
</tr>
</tbody>
</table>
### Table 64  Data change record (LLDF) (part 2 of 4)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>60</td>
<td>3C</td>
<td>0x03</td>
<td>BIN(17)</td>
<td>Timestamp for this log record</td>
</tr>
<tr>
<td>LOGLRSN</td>
<td>77</td>
<td>4D</td>
<td>0x04</td>
<td>BIN(10)</td>
<td>LRSN for this update</td>
</tr>
<tr>
<td>LOGRBA</td>
<td>87</td>
<td>57</td>
<td>0x05</td>
<td>BIN(10)</td>
<td>RBA for this update</td>
</tr>
<tr>
<td>MEMBERID</td>
<td>97</td>
<td>61</td>
<td>0x06</td>
<td>BIN(2)</td>
<td>Data sharing member ID (zero if not data sharing)</td>
</tr>
<tr>
<td>RID</td>
<td>99</td>
<td>63</td>
<td>0x06</td>
<td>BIN(5)</td>
<td>Row ID (page</td>
</tr>
<tr>
<td>CHANGE TYPE</td>
<td>104</td>
<td>68</td>
<td>0x40</td>
<td>CHAR(2)</td>
<td>Type of update:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'UB' – update</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'I ' – insert</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'D ' – delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'DM' – mass delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'DT' – truncate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'DR' – truncate with reuse option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'IL' – insert due to load action</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'CO' – catalog object (DDL information)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'E' – exchange</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'CM' – DB2 command</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>106</td>
<td>6A</td>
<td>0x06</td>
<td>CHAR(1)</td>
<td>U – UNDO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R – REDO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'(blank) – other</td>
</tr>
<tr>
<td>LOGRECDISP</td>
<td>107</td>
<td>6B</td>
<td>0x06</td>
<td>CHAR(1)</td>
<td>Disposition of log record (as opposed to disposition of URID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C – record was committed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A – record was aborted (included in a ROLLBACK action)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S – record was included in a ROLLBACK TO SAVEPOINT action</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O – record was reversed by DB2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(for example, in response to some negative SQL codes)</td>
</tr>
<tr>
<td>SQLSRC TYPE</td>
<td>108</td>
<td>6C</td>
<td>0x06</td>
<td>CHAR(1)</td>
<td>R – record generated by RI constraint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T – record generated by trigger activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B – record generated by both RI constraint and trigger activity (RI takes precedence)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H – record generated by history activity on a system-maintained temporal table</td>
</tr>
<tr>
<td>LOGBYTES</td>
<td>109</td>
<td>6D</td>
<td>0x06</td>
<td>BIN(4)</td>
<td>Number of bytes contained in DB2 log record(s) for this transaction</td>
</tr>
<tr>
<td>LOGDELTA</td>
<td>113</td>
<td>71</td>
<td>0x47</td>
<td>BIN(2)</td>
<td>Number of log bytes that would have been saved had DATA CAPTURE CHANGES been off or the number of additional bytes that would have been logged had DATA CHANGES been on</td>
</tr>
<tr>
<td>ANOMALYROWID</td>
<td>115</td>
<td>73</td>
<td>0x47</td>
<td>BIN(1)</td>
<td>Anomaly Anchor ID from RID</td>
</tr>
<tr>
<td>ANOMALYTYPE</td>
<td>116</td>
<td>74</td>
<td>0x47</td>
<td>CHAR(1)</td>
<td>Tells why this record is in the logical log – N for normal record selected by the filter</td>
</tr>
<tr>
<td>ANOMALYRBA</td>
<td>117</td>
<td>75</td>
<td>0x47</td>
<td>BIN(10)</td>
<td>RBA/LRSN of the anomaly anchor point</td>
</tr>
<tr>
<td>UORTIMESTAMP</td>
<td>127</td>
<td>7F</td>
<td>0x4F</td>
<td>BIN(17)</td>
<td>Timestamp for this unit of recovery</td>
</tr>
</tbody>
</table>
### Table 64  Data change record (LLDF) (part 3 of 4)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UORCOMMITTIMESTAMP*</td>
<td>144</td>
<td>90</td>
<td>BIN(17) *</td>
<td>Timestamp value for the commit record for this unit of recovery</td>
<td></td>
</tr>
<tr>
<td>UORDISP</td>
<td>161</td>
<td>A1</td>
<td>CHAR(1)</td>
<td>Disposition of unit of recovery:</td>
<td>C – committed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A – aborted</td>
</tr>
<tr>
<td>UORIDLRSN</td>
<td>162</td>
<td>A2</td>
<td>BIN(10) *</td>
<td>RBA/LRSN of unit of recovery record</td>
<td></td>
</tr>
<tr>
<td>UORID</td>
<td>172</td>
<td>AC</td>
<td>BIN(10) *</td>
<td>RBA of unit of recovery record</td>
<td></td>
</tr>
<tr>
<td>SEGLLEN</td>
<td>182</td>
<td>B6</td>
<td>BIN(2)</td>
<td>Length of segment</td>
<td></td>
</tr>
<tr>
<td>TOTALSEGS</td>
<td>184</td>
<td>B8</td>
<td>BIN(2)</td>
<td>Total number of segments</td>
<td></td>
</tr>
<tr>
<td>SEGNUM</td>
<td>186</td>
<td>BA</td>
<td>BIN(2)</td>
<td>Sequence number of the current segment</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BC</td>
<td>BIN(4)</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>UORCOMMITLRSN</td>
<td>192</td>
<td>C0</td>
<td>BIN(10) *</td>
<td>RBA/LRSN of the commit record for this unit of recovery</td>
<td></td>
</tr>
<tr>
<td>UORCOMMITPOINT</td>
<td>202</td>
<td>CA</td>
<td>BIN(10) *</td>
<td>RBA of the commit record for this unit of recovery</td>
<td></td>
</tr>
<tr>
<td>CONNECTIONTYPE</td>
<td>212</td>
<td>D4</td>
<td>CHAR(2)</td>
<td>Type of connection</td>
<td></td>
</tr>
<tr>
<td>CONNECTID</td>
<td>214</td>
<td>D6</td>
<td>CHAR(8)</td>
<td>Connection ID</td>
<td></td>
</tr>
<tr>
<td>CORRELATIONID</td>
<td>222</td>
<td>DE</td>
<td>CHAR(12)</td>
<td>Correlation ID</td>
<td></td>
</tr>
<tr>
<td>AUTHID</td>
<td>234</td>
<td>EA</td>
<td>CHAR(8)</td>
<td>Authorization ID</td>
<td></td>
</tr>
<tr>
<td>PLAN</td>
<td>242</td>
<td>F2</td>
<td>CHAR(8)</td>
<td>Plan name</td>
<td></td>
</tr>
<tr>
<td>LUWNETWORKID</td>
<td>250</td>
<td>FA</td>
<td>CHAR(8)</td>
<td>Logical unit of work (LUW) network ID</td>
<td></td>
</tr>
<tr>
<td>LUWNAME</td>
<td>258</td>
<td>I02</td>
<td>CHAR(8)</td>
<td>LUW name</td>
<td></td>
</tr>
<tr>
<td>LUWINSTANCENO</td>
<td>266</td>
<td>10A</td>
<td>BIN(6)</td>
<td>LUW instance number</td>
<td></td>
</tr>
<tr>
<td>LUWSEQUENCENO</td>
<td>272</td>
<td>110</td>
<td>BIN(2)</td>
<td>LUW commit sequence number</td>
<td></td>
</tr>
<tr>
<td>INCOMPLETETRANS</td>
<td>274</td>
<td>112</td>
<td>CHAR(1)</td>
<td>Y – record is part of a transaction (URID) that contains at least one log record that Log Master cannot complete</td>
<td></td>
</tr>
<tr>
<td>INCOMPLETEDEP</td>
<td>275</td>
<td>113</td>
<td>CHAR(1)</td>
<td>Y – record is part of a transaction (URID) that might depend on another URID that contains at least one record that Log Master cannot complete</td>
<td></td>
</tr>
<tr>
<td>UORHASCOMP</td>
<td>276</td>
<td>114</td>
<td>CHAR(1)</td>
<td>Y – this record is part of a transaction (URID) that contains at least one compensation log record</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N – this record is part of a transaction that contains no compensation log records</td>
<td></td>
</tr>
</tbody>
</table>
The layout of the information in the DATA portion of the Data Change Record is similar to the layout of a DB2 log record, with the following exceptions:

- All numeric fields (DB2 Types INT, SMALL INT, DEC, and FLOAT) are translated from the DB2 internal format to an external format. This translation enables the numeric value to be used in SAS or other application programs. However, if a column contains negative values, the translated value cannot be sorted correctly.

- DATE, TIME, and TIMESTAMP fields are translated from the DB2 internal format to an external format (DB2, ISO, or SAS.) This translation means that more space is used to represent these fields in character format.

- If a column (field) is nullable, the null byte precedes the data for that column.
Variable length fields contain a two-byte length attribute before the actual data. (This format is the same as in the DB2 log).

For variable length fields that are nullable, the position of the null byte in the logical log is different than its position in the DB2 log. In the following example, \texttt{x'1111'} represents the two-byte field length information and \texttt{x'nn'} represents the null byte.

\begin{itemize}
  \item DB2 log:
    \begin{verbatim}
    x'1111'x'nn'
    \end{verbatim}
  \item Logical log:
    \begin{verbatim}
    x'nn'x'1111'
    \end{verbatim}
\end{itemize}

In addition to the null byte order being reversed, the length value in the DB2 log \emph{includes} the one-byte length of the null byte. The extra byte is \emph{not included} in the length value in the logical log.

Examples:

For a five-character field, not nullable:

\begin{itemize}
  \item DB2 log:
    \begin{verbatim}
    x'0005'ABCDE
    \end{verbatim}
  \item Logical log:
    \begin{verbatim}
    x'0005'ABCDE
    \end{verbatim}
\end{itemize}

For a five-character field, nullable (not set to null):

\begin{itemize}
  \item DB2 log:
    \begin{verbatim}
    x'0006'x'00'ABCDE
    \end{verbatim}
  \item Logical log:
    \begin{verbatim}
    x'00'x'0005'ABCDE
    \end{verbatim}
\end{itemize}

For a five-character field, nullable (set to null):

\begin{itemize}
  \item DB2 log:
    \begin{verbatim}
    x'0001'x'FF'
    \end{verbatim}
  \item Logical log:
    \begin{verbatim}
    x'FF'x'0000'
    \end{verbatim}
\end{itemize}

\textbf{NOTE}

If CHANGE TYPE is set to DM (mass delete), data segments might or might not exist. If the mass delete action is the result of a LOAD REPLACE of a specific partition, data segments will exist defining the limit key values.
Normal and segmented logical log record formats

As Log Master creates a logical log, it can encounter log records that are too long to write in the product’s default output record format of variable length, blocked (VB). This situation occurs when all of the following conditions exist:

- the log records reflect activity from table spaces with a page size of 32 KB
- the log records reflect updates to the database (update log records contain both a before image and an after image of the row)
- the total length of any update log record that the product writes (including the logical log header, the before image, and the after image) is greater than the system-determined block size of the device where the product writes the logical log

The situation is more likely to occur when you set the EXPANDVAR keyword to YES. The product detects when any update log records exceed the length restrictions of the VB format. If such records exist, the product automatically divides the record into multiple segments and writes those segments to a variable length, blocked file. When Log Master reads logical logs created by a 2.2.00 or later version of the product, it reassembles the segments into a complete record before it processes them.

You can use the following methods to determine if a logical log contains segmented logical log records:

- In the logical log control file, the Logical Log Data Set record contains a field named SEGMENTED. This field contains a Y if the logical log data file contains any segmented logical log records. For more information, see “Logical log data set record (DLDS)” on page 338.

- In the logical log data file, each record begins with a logical log header. This header contains a field named TOTALSEGS. If the logical log record takes up more than one segment, this field contains a number greater than 1. For more information, see “Data change record (LLDF)” on page 370.

To understand how the product divides a single record into segments, first consider the basic layout of a logical log record, shown in Figure 47 on page 377.
Log Master can divide a logical log record that reflects an update into one, two, or three segments. Figure 48 shows how the product divides the record into multiple segments and how the three indicator fields in the logical log header describe the segments.
Logical log files and ongoing processing

During an ongoing log scan, Log Master keeps track of each run of the log scan and any transactions (URIDs) that are still open at the end point of the log scan. For ongoing log scans, the product ensures that

- any transactions that are open at the end of the current log scan will be included in the next log scan (during the next run of the work ID)
- any transactions that were completed within the previous log scan are not processed twice, even though the product might scan part of the same log range again

Similarly, application programs must take similar action when they operate on a logical log file that was generated by an ongoing log scan. In practice, this means that the application program might need to ignore some records from a logical log data file that was generated by an ongoing log scan. The program must ignore these records if they are

- part of a transaction that is incomplete at the end of the ongoing log scan
- dependent on a transaction that is incomplete at the end of the ongoing log scan (when the logical log data records reflect changes to the same table rows that are affected by an incomplete transaction)

To determine if the program should ignore a record in the logical log data file, read the logical log control file that was created in conjunction with the data file. The logical log control file has one instance of a Log Information End (DLGE) record. (For more information, see page 341.) The DLGE record contains a field named DEPENDENTRBA, which is a twelve-byte character field that contains the externalized hexadecimal representation (using the EBCDIC character set) of a six-byte binary RBA/LRSN value. For more information, see “Dependent RBA/LRSN Value” on page 379.

- If the value of this field is x'FFFFFFFFFFFF', the logical log data file does not contain any records that should be ignored.
If the value of this field is not `x'FFFFFFFFFFFF'`, compare the value of the DEPENDENTRBA field to the value of the UORIDLRSN field in each record in the logical log data file. Remember that UORIDLRSN is a six-byte binary field, so the program must either convert DEPENDENTRBA to a binary number or convert UORIDLRSN to a character string before making the comparison.

— If the UORIDLRSN field is greater than or equal to the DEPENDENTRBA field, ignore the record. The same record will appear in the logical log data file produced by a subsequent run of the ongoing log scan.

— If the UORIDLRSN field is less than the DEPENDENTRBA field, process the record.

**Dependent RBA/LRSN Value**

Log Master stores a dependent RBA/LRSN value when any of the following conditions exist for a transaction (unit of recovery) that is selected by an ongoing log scan:

- the transaction is committed within the range of the log scan
- the transaction contains log records that the product does not complete during the log scan

The dependent RBA value is the log address of the Begin UR log record of the earliest transaction within the log scan that meets these conditions. If no transaction in the log scan meets the conditions, the product stores a dependent RBA value of `x'FFFFFFFFFFFF'`. Figure 49 shows a dependent RBA value.

**Figure 49  Dependent RBA value for ongoing log scans**

<table>
<thead>
<tr>
<th>Completed transactions</th>
<th>Open transactions</th>
<th>Incomplete transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specified start point (FROM value)</td>
<td>Specified end point (TO value)</td>
<td>Dependent RBA/LRSN value</td>
</tr>
</tbody>
</table>

Dependent RBA/LRSN Value

<table>
<thead>
<tr>
<th>Specified start point (FROM value)</th>
<th>Specified end point (TO value)</th>
<th>Dependent RBA/LRSN value</th>
</tr>
</thead>
</table>
The presence of a dependent RBA value affects the open transactions that Log Master includes in the ALPURID table of the Repository for use in subsequent ongoing log scans. Note the following points regarding Figure 49 on page 379:

- The incomplete transactions INC1 and INC2 cause Log Master to store a dependent RBA value which in turn causes the product to store only the open transaction OPN3 in the Repository (because it starts before the point of the dependent RBA).

- If transactions INC1 and INC2 were not present, Log Master would store a dependent RBA value of \textit{x'FFFFFFFFFFFF'} and store the open transactions OPN1, OPN2, and OPN3 in the ALPURID table of the Repository.
Chapter 5 Log Master for DB2 Repository

This chapter contains the following topics:

Repository overview .......................................................... 382
  Repository tables and table spaces ..................................... 383
  Repository table relationships .......................................... 384
  Data integrity considerations ............................................ 385
  Maintaining the Repository ............................................. 385
Log Master Repository tables ............................................. 386
  Automated drop recovery file table (ALPADDRF) ............... 387
  Column list table (ALPCOLS) ............................................. 388
  DDL file pointer table (ALPDDLFL) .................................. 388
  Dropped objects table (ALPDRLN) .................................... 389
  Filter header table (ALPFLTH) ........................................ 390
  Filter line table (ALPFLIN) ............................................. 391
  Filter pointer table (ALPFLTP) ........................................ 394
  Load file pointer table (ALPLODF) .................................. 394
  Logical log file pointer table (ALPLOGF) ......................... 395
  Mark table (ALPMARK) ................................................... 397
  Old objects table (ALPOLDO) .......................................... 398
  Open unit of recovery table (ALPURID) ............................ 399
  Report file pointer table (ALPRPTF) ................................. 400
  Report template table (ALPRTMPL) ................................... 403
  Security exception history table (ALPSECH) ...................... 404
  SQL codes handler table (ALPSQLC) ................................. 405
  SQL file pointer table (ALPSQLF) .................................... 406
  SQL index table (ALPIXSQ) ............................................. 408
  SQL name translation table (ALPSQLX) .............................. 408
  Work ID header table (ALPWHDR) .................................... 409
  Work ID history table (ALPWHIS) .................................... 413
  Work ID input file definitions table (ALPIFIL) .................. 414
  Work ID log scan history table (ALPWLSH) ....................... 416
  Work ID output file definitions table (ALPOFIL) ............... 417
  Work ID Step Table (ALPWSTP) ....................................... 432
Repository overview

The Log Master for DB2 product accumulates information as you use the product and stores that information in the Log Master for DB2 Repository. The Repository is a set of DB2 tables. To use the online interface, or to run a job that requires access to the stored information, DB2 must be operational.

The tables in the Repository support on the following product activities:

- **Work ID generation and maintenance**
  A work ID can consist of steps to perform any of the following tasks:

  - create a log mark
  - define a log scan
  - execute generated Structured Query Language (SQL) statements
  - recover dropped objects
  - generate High-speed Apply JCL

- **Retention of batch results**
  The product saves the results of a work ID in the Repository. The results include both the final status of the work ID and pointers to the various files generated as the work ID runs. These files include

  - logical log files
  - SQL files
  - load files
  - report files
  - drop recovery output files

When the product runs a work ID, it writes a row that defines an individual run of the work ID to the Work ID History Table. It also associates any generated log marks or files with this row.

To reduce the number of tables used, the Repository applies a significant amount of denormalization to its internal tables. Columns in a number of the tables use overloading depending on some other value in the row. For example, the Work ID Step Table contains different information depending on the type of step (log mark, log scan, execute SQL, or drop recovery) as indicated in the type field in the row.

One specific table in the Repository (called the Old Objects Table) provides support for overtime processing and compression dictionaries. For more information about this support, see “REPOS UPDATE” on page 167, or for information about updating the Repository, see the chapter 10 on objects over time in the Log Master for DB2 User Guide.
Repository tables and table spaces

Log Master stores information in a set of DB2 tables. The product groups these tables in table spaces based on how the tables relate to each other and the amount of data that they contain. Table 65 shows the Repository table spaces and tables. During installation, the product creates table spaces with a page size of 4 KB, except as indicated in the following table.

Table 65  Repository table spaces and tables

<table>
<thead>
<tr>
<th>Repository table space</th>
<th>Repository tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>work ID Header Table Space</td>
<td>Work ID Header Table</td>
</tr>
<tr>
<td>Work ID Step Table Space</td>
<td>Work ID Step Table</td>
</tr>
<tr>
<td>Work ID Input File Definitions Table</td>
<td>Work ID Input File Definitions Table</td>
</tr>
<tr>
<td>Column List Table</td>
<td>Column List Table</td>
</tr>
<tr>
<td>Filter Pointer Table</td>
<td>Filter Pointer Table</td>
</tr>
<tr>
<td>Work ID Output File Definitions Table</td>
<td>Work ID Output File Definitions Table (page size =32 KB)</td>
</tr>
<tr>
<td>Filter Table Space</td>
<td>Filter Header Table</td>
</tr>
<tr>
<td>Work ID History Table Space</td>
<td>Work ID History Table</td>
</tr>
<tr>
<td>Work ID Log Scan History Table</td>
<td>Work ID Log Scan History Table</td>
</tr>
<tr>
<td>Security Exception History Table</td>
<td>Security Exception History Table</td>
</tr>
<tr>
<td>Mark Table Space</td>
<td>Mark Table</td>
</tr>
<tr>
<td>File Pointer Table Space</td>
<td>SQL Index Table</td>
</tr>
<tr>
<td>Logical Log File Pointer Table</td>
<td>Logical Log File Pointer Table</td>
</tr>
<tr>
<td>Report File Pointer Table</td>
<td>Report File Pointer Table</td>
</tr>
<tr>
<td>SQL File Pointer Table</td>
<td>SQL File Pointer Table</td>
</tr>
<tr>
<td>Load File Pointer Table</td>
<td>Load File Pointer Table</td>
</tr>
<tr>
<td>SQL Codes Handler Table</td>
<td>SQL Codes Handler Table</td>
</tr>
<tr>
<td>DDL File Pointer Table</td>
<td>DDL File Pointer Table</td>
</tr>
<tr>
<td>SQL Name Translation Table</td>
<td>SQL Name Translation Table</td>
</tr>
<tr>
<td>Automated Drop Recovery File Table</td>
<td>Automated Drop Recovery File Table</td>
</tr>
<tr>
<td>Open Unit of Recovery (URID) Table Space</td>
<td>Open Unit of Recovery Table</td>
</tr>
<tr>
<td>Old Objects Table Space</td>
<td>Old Objects Table</td>
</tr>
<tr>
<td>(page size =32 KB)</td>
<td>(page size =32 KB)</td>
</tr>
<tr>
<td>Automated Drop Recovery Table Space</td>
<td>Recovery Analysis Output Table</td>
</tr>
<tr>
<td>Recovery Manager Transaction Table</td>
<td>Recovery Manager Transaction Table</td>
</tr>
<tr>
<td>Report Template Table Space</td>
<td>Report Template Table</td>
</tr>
<tr>
<td>(page size =32 KB)</td>
<td>(page size =32 KB)</td>
</tr>
</tbody>
</table>
Repository table relationships

Figure 50 shows the primary relationships between the Repository tables.

Each Repository table has a unique index. When the product primarily accesses the table by using the unique index, that index is a clustering index. The product creates additional indexes as needed for each table.
Data integrity considerations

BMC Software recommends that you do not directly update the Repository. Restrict insert, update, and delete privileges on Repository tables to personnel installing the product.

Maintaining the Repository

Log Master uses the tables in the Repository in the same way that an application program uses any group of DB2 tables. Use the following standard practices of recoverable DB2 operation to maintain the Repository:

Regular backups

Routinely create image copies of the Repository table spaces. Use a DB2 Modify utility regularly.

A sample job that backs up the Repository is included in the Log Master sample library downloaded from the installation tape. The name of the member is HLQ\DBSAM\P(\ALP$BKUP). The HLQ value represents a qualifier that your environment assigns during installation. This sample job uses the BMC Software product COPY PLUS for DB2.

Regular maintenance of Repository tables

Routinely run a DB2 Runstats utility on the Repository table spaces and indexes. Use the information reported by this utility to determine when to run a DB2 Reorg utility to reorganize table spaces and associated indexes. If you fail to maintain Repository table spaces and indexes, the performance of the product can degrade over time.

Reduce Repository size when required

The product stores information in the Repository each time it runs a job. Over time, the Repository grows in size. To reduce the size of the Repository, use one of these methods.

- Reduce overall size of Repository
You can periodically delete older rows from Repository tables. Use the following members that are included in the Log Master sample library that is downloaded from the installation tape. The name of the library is HLQ.DBSAMP. The HLQ value represents a qualifier that your environment assigns during installation.

- **ALPRCREX**: A sample REXX exec that generates SQL to delete information from the Repository
- **ALPRCJCL**: A sample JCL file that runs the REXX exec
- **ALPRCTXT**: A set of brief instructions about how to run the REXX exec and apply the SQL

Use these instructions to modify the sample files for your environments and delete stored information.

- **Reduce size of Old Objects (ALPOLDO) or Open Unit of Recovery (ALPURID) tables of the Repository**

To reduce the size of either or both of these tables, use the REPOS DELETE keyword of the LOGSCAN statement. For more information, see “REPOS DELETE” on page 171.

You can also use options on the Main Menu of the product’s online interface to generate JCL that deletes information from these tables, or to delete or display information interactively.

- **Reduce size of Open Unit of Recovery (ALPURID) and Work ID Log Scan History (ALPWLSH) tables of the Repository**

The RESET PURGE keywords of the LOGSCAN statement reduce the size of both the ALPURID and ALPWLSH tables. (For more information, see “RESET” on page 178. However these keywords operate on only one combination of user ID, work ID, and ongoing handle ID at a time. To obtain large reductions in the size of these tables, use the REXX exec described in the preceding paragraphs.

---

**Log Master Repository tables**

The following sections list the Repository tables in order of their full names. The headings include the abbreviated name in parentheses (for example, ALPADRF). The actual names of the Repository tables in DB2 are determined when the product is installed. The default DB2 table names use the format

```
BMCALP.xxxxxxxx
```

In this format xxxxxxx represents the abbreviated name.
Automated drop recovery file table (ALPDRF)

This table records the output files created as part of a drop recovery job step. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. For those columns that are overloaded, subsequent tables define the individual fields within the overloaded columns.

Table 66 shows columns that make up the unique index key in bold face type.

### Table 66  Automated drop recovery file table (ALPDRF)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNAME</td>
<td>CHAR(44)</td>
<td>Name of data set that contains automated drop recovery output</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(8)</td>
<td>Member name of the file (if the output file is a member of a partitioned data set)</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASID</td>
<td>SMALLINT</td>
<td>Work ID Phase Number</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Work ID step number that generated the data set</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>FILETIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date and time the file was created (hexadecimal format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000007D90512133B2600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7D9 year 2009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>05 month May</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 day 18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 hours 19:00.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3B minutes 00:59.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26 seconds 00:00.38</td>
</tr>
<tr>
<td>FILETYPE</td>
<td>SMALLINT</td>
<td>Code indicating the content of the data set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Recover utility control information (SYSIN syntax)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – SQL for post-recovery processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – DB2 commands (REBIND, and so on)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Copy control information (SYSIN syntax)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Check data for post-recovery processing</td>
</tr>
<tr>
<td>FILETS</td>
<td>TIMESTAMP</td>
<td>Date and time the file was created</td>
</tr>
</tbody>
</table>
**Column list table (ALPCOLS)**

This table stores the list of columns that a job selects from particular tables. This list corresponds to the Column Include/Exclude Definition (for more information, see “LOGSCAN column include/exclude definition” on page 267). If a table’s columns do not appear in this table, all columns are selected from the table. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 67 shows columns that make up the unique index key in bold face type.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Step number within the work ID</td>
</tr>
<tr>
<td>LINENUM</td>
<td>SMALLINT</td>
<td>File sequence number to associate to a particular output file</td>
</tr>
<tr>
<td>TABLEUSERID</td>
<td>VARCHAR(259)</td>
<td>User ID associated with the table</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>VARCHAR(259)</td>
<td>Name of the DB2 table that contains the column</td>
</tr>
<tr>
<td>COLUMNNAME</td>
<td>VARCHAR(259)</td>
<td>Column name within the table</td>
</tr>
<tr>
<td>INCLFLAG</td>
<td>SMALLINT</td>
<td>0 – column is excluded from selected data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nonzero – column is included in any log data selected from the table</td>
</tr>
</tbody>
</table>

**DDL file pointer table (ALPDDLF)**

This table contains a row for each data definition language (DDL) file generated. Unless otherwise noted, all columns have NOT NULL WITH DEFAULT attribute.

Table 68 shows columns that make up the unique index key in bold face type.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the DDL file data set</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(8)</td>
<td>Member name (if the file is a member of a PDS)</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
</tbody>
</table>
Dropped objects table (ALPDRLN)

This table contains a row for each database object that is the target of a drop recovery job step. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. For those columns that are overloaded, subsequent tables define the individual fields within the overloaded columns.

Table 69 shows columns that make up the unique index key in bold face type.

Table 68  DDL file pointer table (ALPDDLF) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASID</td>
<td>SMALLINT</td>
<td>Work ID Phase Number</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Work ID step number that generated the data set</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>FILETIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time created (See page 387 for format)</td>
</tr>
<tr>
<td>VALIDFILE</td>
<td>SMALLINT</td>
<td>0 – Data set is missing from operating system’s catalog 1 – Data set is valid</td>
</tr>
<tr>
<td>CREATECOUNT</td>
<td>INTEGER</td>
<td>Number of CREATE DDL statements in the file</td>
</tr>
<tr>
<td>DROPDCOUNT</td>
<td>INTEGER</td>
<td>Number of DROP DDL statements in the file</td>
</tr>
<tr>
<td>GRANTCOUNT</td>
<td>INTEGER</td>
<td>Number of GRANT DDL statements in the file</td>
</tr>
<tr>
<td>REVOKECOUNT</td>
<td>INTEGER</td>
<td>Number of REVOKE DDL statements in the file</td>
</tr>
<tr>
<td>ALTERCOUNT</td>
<td>INTEGER</td>
<td>Number of ALTER DDL statements in the file</td>
</tr>
<tr>
<td>COMMENTCOUNT</td>
<td>INTEGER</td>
<td>Number of COMMENT DDL statements in the file</td>
</tr>
<tr>
<td>LABELONCOUNT</td>
<td>INTEGER</td>
<td>Number of LABELON DDL statements in the file</td>
</tr>
<tr>
<td>UNDOREDOFLAG</td>
<td>SMALLINT</td>
<td>0 – UNDO DDL 1 – Reserved 2 – MIGRATE DDL</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>SMALLINT</td>
<td>Indicator that defines the format of dates within the data set: 0 – USA 1 – EUR 2 – ISO 3 – JIS</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the file</td>
</tr>
<tr>
<td>RENAMECOUNT</td>
<td>INTEGER</td>
<td>Number of RENAME DDL statements in the file</td>
</tr>
<tr>
<td>FILETS</td>
<td>TIMESTAMP</td>
<td>Date/time created</td>
</tr>
</tbody>
</table>
**Filter header table (ALPFLTH)**

This table contains a row for every filter defined. The row defines general information about the filter. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 70 shows columns that make up the unique index key in bold face type.

**Table 69  Dropped objects table (ALPDRLN)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| USERID          | CHAR(8)     | User ID associated with the work ID. This value can be:  
|                 |             | ■ defined by the user  
|                 |             | ■ obtained from the user’s TSO prefix  
|                 |             | ■ obtained from the user’s TSO ID                                                                                                                  |
| WORKIDNAME      | CHAR(18)    | Unique name associated with the work ID                                                                                                          |
| STEPNUM         | SMALLINT    | Sequential step number within the work ID                                                                                                          |
| LINENUM         | SMALLINT    | File sequence number                                                                                                                                |
| LINETYPE        | SMALLINT    | Code indicating the type of object information contained in the OBJDATA field:  
|                 |             | 0 – Database  
|                 |             | 1 – Table space  
|                 |             | 2 – Table                                                                                                                                     |
| OBJDATA         | VARCHAR(1024) | Contents dependent on the LINETYPE values                                                                                                         |
| ALPVERSION      | CHAR(6)     | Version of the product that created or updated the object data (format: vvvvmm, example: 090100)                                                                                                          |

**Table 70  Filter header table (ALPFLTH) (part 1 of 2)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User that created the filter</td>
</tr>
<tr>
<td>FILTERNAME</td>
<td>CHAR(18)</td>
<td>Name of the filter</td>
</tr>
<tr>
<td>TSOID</td>
<td>CHAR(8)</td>
<td>TSO ID of the creator of the filter for use with security checking</td>
</tr>
<tr>
<td>CREATEDBY</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the individual creating the filter</td>
</tr>
<tr>
<td>CREATEDTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/Time that the filter was created</td>
</tr>
<tr>
<td>UPDATEDBY</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the individual updating the filter</td>
</tr>
<tr>
<td>UPDATEDTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/Time filter was last updated</td>
</tr>
</tbody>
</table>
| FILTERTYPE      | SMALLINT    | 0 – Filter consists of structured fields  
|                 |             | 1 – Filter consists of free-form WHERE clause lines                                                                                                |
Table 70  Filter header table (ALPFLTH) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATEFORMAT</td>
<td>SMALLINT</td>
<td>0 – USA date 1 – EUR 2 – ISO 3 – JIS</td>
</tr>
<tr>
<td>ACCESSMODE</td>
<td>SMALLINT</td>
<td>0 – Universal access 1 – All users can read; only creator can update 2 – Filter cannot be viewed or updated by anyone other than the creator of the filter</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the filter</td>
</tr>
<tr>
<td>ALPVERSION</td>
<td>CHAR(6)</td>
<td>Version of the product that created or updated the filter (format: vvvrrmm, example: 090100)</td>
</tr>
<tr>
<td>CREATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time that the filter was created</td>
</tr>
<tr>
<td>UPDATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time that the filter was last updated</td>
</tr>
</tbody>
</table>

**Filter line table (ALPFLIN)**

This table contains rows that define the individual log selection criteria of a filter. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 71 shows columns that make up the unique index key in bold face type.

Table 71  Filter line table (ALPFLIN) (part 1 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User that created the filter</td>
</tr>
<tr>
<td>FILTERNAME</td>
<td>CHAR(18)</td>
<td>Name of the filter</td>
</tr>
<tr>
<td>LINENUM</td>
<td>SMALLINT</td>
<td>For structured filters, this field defines the line within a group. For free-form filters, this field defines the line number of the free-form text.</td>
</tr>
<tr>
<td>GROUPNUM</td>
<td>SMALLINT</td>
<td>When the filter is entered as a structured list of selections, this field defines the group of requirements to which the line belongs.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>LINETYPE</td>
<td>SMALLINT</td>
<td>Code indicating the type of line information contained in the LINEDATA field: 0 – WHERE clause line 1 – Unit of recovery identifier 2 – Correlation ID 3 – Plan name 4 – Auth ID 5 – Connection ID 6 – Database name 7 – Table space name 8 – Table name 9 – DBID 10 – PSID 11 – OBID 12 – Update type</td>
</tr>
<tr>
<td>(continued)</td>
<td>SMALLINT</td>
<td>Code indicating the type of line information contained in the LINEDATA field: 13 – Column value 14 – Connection type 15 – Continuation 16 – Last continuation 17 – Catalog activity type 18 – Catalog object type 19 – Member name 20 – Member ID 21 – DB2 subsystem ID (SSID) 22 – Commands</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>SMALLINT</td>
<td>Value indicating comparison operator to use: 0 – Equal 1 – Not equal 2 – Greater than 3 – Less than 4 – Greater than or equal to 5 – Less than or equal to 6 – In 7 – Not in 8 – Like 9 – Not like 10 – Changed 11 – Null 12 – Not null</td>
</tr>
</tbody>
</table>
Filter line table (ALPFLIN)

Table 71  Filter line table (ALPFLIN) (part 3 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINEDATA</td>
<td>VARCHAR(2048)</td>
<td>Value dependent on the LINETYPE values and operator. In the case of the IN operator, the value might be repeated as many times as will fit in the space provided.</td>
</tr>
<tr>
<td>FILTERIMAGE</td>
<td>SMALLINT</td>
<td>Code indicating which row image to filter against: 0 – Before image 1 – After image 2 – Both before and after images</td>
</tr>
</tbody>
</table>

### LINEDATA field: Catalog activity type

This table shows the layout of the LINEDATA field in the Filter Line Table (Table 71 on page 391) when the LINETYPE value is catalog activity type (17).

Table 72  LINEDATA field: Catalog activity line type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>CREATE</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>DELETE</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>DROP</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>GRANT</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>RENAME</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>REVOKE</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>BIND</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>REBIND</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
<tr>
<td>FREE</td>
<td>SMALLINT</td>
<td>0 – Not specified in filter 1 – Catalog activity type specified in filter</td>
</tr>
</tbody>
</table>
Filter pointer table (ALPFLTP)

This table contains rows that refer to (point to) the filters used with a work ID step. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 73 on page 394 shows columns that make up the unique index key in bold face type.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Step number within the work ID</td>
</tr>
<tr>
<td>FILTERUSERID</td>
<td>CHAR(8)</td>
<td>User that created the filter</td>
</tr>
<tr>
<td>FILTERNAME</td>
<td>CHAR(18)</td>
<td>Name of the filter</td>
</tr>
</tbody>
</table>

Load file pointer table (ALPLODF)

This table contains one row for each load file generated by the product. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 74 shows columns that make up the unique index key in bold face type.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the load file</td>
</tr>
<tr>
<td>CONTROLDSNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the load control file</td>
</tr>
<tr>
<td>CONTROLMEMBER</td>
<td>CHAR(8)</td>
<td>Member name of the load control file</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Work ID phase number</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Work ID step number that generated the data set</td>
</tr>
</tbody>
</table>
Logical log file pointer table (ALPLOGF)

This table contains one row for each logical log file generated by the product. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 75 shows columns that make up the unique index key in bold face type.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>MVW job number</td>
</tr>
<tr>
<td>FILETIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time created (for format, see page 387)</td>
</tr>
<tr>
<td>VALIDFILE</td>
<td>SMALLINT</td>
<td>0 – Data set is missing from operating system’s catalog 1 – Data set is valid</td>
</tr>
<tr>
<td>RECORDCOUNT</td>
<td>INTEGER</td>
<td>Number of records in the data set</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>SMALLINT</td>
<td>0 – USA 1 – EUR 2 – ISO 3 – JIS</td>
</tr>
<tr>
<td>TABLEUSERID</td>
<td>VARCHAR(128)</td>
<td>If the SEPARATE DATASETS keyword of the LOGSCAN statement is set to YES, this field contains the creator name (user ID) associated with the table name for this data set.</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>VARCHAR(128)</td>
<td>If the SEPARATE DATASETS keyword of the LOGSCAN statement is set to YES, this field contains the table name for this data set.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the file</td>
</tr>
<tr>
<td>FIELDS</td>
<td>INTEGER</td>
<td>Reserved</td>
</tr>
<tr>
<td>DELIMITED</td>
<td>SMALLINT</td>
<td>Indicator that defines the format of the output load file: 1 – Log Master native format 2 – UNLOAD PLUS format 3 – CSV (Comma Separated Value) format 4 – SDF (Standard Definition Format) format</td>
</tr>
<tr>
<td>DELIMITER</td>
<td>CHAR(1)</td>
<td>Character used to delimit fields in load records. This field is valid only if the format of the load file is character-delimited (for example, CSV)</td>
</tr>
<tr>
<td>CC_FORMAT</td>
<td>SMALLINT</td>
<td>Reserved</td>
</tr>
<tr>
<td>FILETS</td>
<td>TIMESTAMP</td>
<td>Date/time created</td>
</tr>
</tbody>
</table>

Table 74  Load file pointer table (ALPLODF) (part 2 of 2)
Table 75  Logical log file pointer table (ALPLOGF) (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the logical log control file data set</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(8)</td>
<td>Member name (if the file is a member of a PDS)</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Work ID phase number</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Work ID step number that generated the data set</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>DATADSNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the logical log data set</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>FILETIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time created (for format, see page 387)</td>
</tr>
<tr>
<td>VALIDFILE</td>
<td>SMALLINT</td>
<td>0 – Data set is missing from operating system’s catalog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Data set is valid</td>
</tr>
<tr>
<td>RECORDCOUNT</td>
<td>INTEGER</td>
<td>Number of log records in the data set</td>
</tr>
<tr>
<td>STARTTIME</td>
<td>VARCHAR(10)</td>
<td>Starting time represented in the logical log</td>
</tr>
<tr>
<td>ENDTIME</td>
<td>VARCHAR(10)</td>
<td>Ending time represented in the logical log</td>
</tr>
<tr>
<td>BYTECOUNT</td>
<td>INTEGER</td>
<td>Number of KB in the data set</td>
</tr>
<tr>
<td>STARTRBA</td>
<td>VARCHAR(10)</td>
<td>Starting RBA/LSRN scanned</td>
</tr>
<tr>
<td>ENDRBA</td>
<td>VARCHAR(10)</td>
<td>Ending RBA/LSRN scanned</td>
</tr>
<tr>
<td>OPENTRANSFLAG</td>
<td>SMALLINT</td>
<td>0 – No open transaction information saved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Open transaction information is saved in the control file</td>
</tr>
<tr>
<td>RECORDFORMAT</td>
<td>SMALLINT</td>
<td>0 – Variable, blocked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Variable, blocked, spanned</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>SMALLINT</td>
<td>0 – Dates are in DB2 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(YYYY-MM-DD-hh.mm.ss.nnnnnnn)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When running on DB2 Version 10 and later, the product supports precision</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timestamps up to 12 digits, and inclusion of a time zone in the timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Dates are in SAS character format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(YYYY/MM/DD/hh:mm:ss:nnnnnn)</td>
</tr>
<tr>
<td>EXPANDVAR</td>
<td>SMALLINT</td>
<td>0 – VAR columns are not expanded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – VAR columns are expanded to maximum size</td>
</tr>
<tr>
<td>RIFLAG</td>
<td>SMALLINT</td>
<td>0 – RI information not captured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – RI information captured</td>
</tr>
</tbody>
</table>
Mark table (ALPMARK)

This table contains one row for each log mark defined using the product. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 76 shows columns that make up the unique index key in bold face type.

Table 75  Logical log file pointer table (ALPLOGF) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>SMALLINT</td>
<td>1 - DB2 data</td>
</tr>
<tr>
<td>ALPVERSION</td>
<td>CHAR(6)</td>
<td>Version of the product that created or updated the logical log control file and logical log data file (format: vvvrrmm, example: 090100)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the file</td>
</tr>
<tr>
<td>FILETS</td>
<td>TIMESTAMP</td>
<td>Date/time created</td>
</tr>
<tr>
<td>STARTS</td>
<td>TIMESTAMP</td>
<td>Starting time represented in the logical log</td>
</tr>
<tr>
<td>ENDS</td>
<td>TIMESTAMP</td>
<td>Ending time represented in the logical log</td>
</tr>
</tbody>
</table>

Mark table (ALPMARK)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARKNAME</td>
<td>CHAR(40)</td>
<td>Name of the generated log mark</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
<td>Absolute version number (value is 0 if the log mark is not a member of a generation group)</td>
</tr>
<tr>
<td>MARKRBA</td>
<td>VARCHAR(10)</td>
<td>Current DB2 log RBA/LSRN at the time of the log mark</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>MARKTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time that the log mark was created</td>
</tr>
<tr>
<td>QUIESCEFLAG</td>
<td>SMALLINT</td>
<td>If zero, no quiesce was taken at time of the log mark</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>User-supplied text associated with the log mark</td>
</tr>
</tbody>
</table>
Log Master uses the information in this table to interpret and process log records that relate to database objects not currently defined in the DB2 catalog. Each set of rows in this table defines a DB2 object at a certain point in time (based upon the RBA/LSRN when an object was created or dropped).

For more information about using this table, see “REPOS UPDATE” on page 167. For examples and scenarios involving the Old Objects Table, see the chapter on objects over time in the Log Master for DB2 User Guide.

Table 77 shows columns that make up the unique index key in bold face type.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBID</td>
<td>SMALLINT</td>
<td>DBID related to the object this row describes</td>
</tr>
<tr>
<td>PSID</td>
<td>SMALLINT</td>
<td>PSID related to the object this row describes</td>
</tr>
<tr>
<td>OBID</td>
<td>SMALLINT</td>
<td>REC_TYPE=T - OBID of table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REC_TYPE=D - 0</td>
</tr>
<tr>
<td>BEGINRBA</td>
<td>VARCHAR(10)</td>
<td>Beginning RBA/LSRN of the object this row describes</td>
</tr>
<tr>
<td>ENDRBA</td>
<td>VARCHAR(10)</td>
<td>Ending RBA/LSRN of the object this row describes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For D type records, the RBA at which the compression dictionary was last rebuilt.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>VARCHAR(24)</td>
<td>Database name related to the object this row describes</td>
</tr>
<tr>
<td>TSPACE</td>
<td>VARCHAR(24)</td>
<td>Table space name related to the object this row describes</td>
</tr>
<tr>
<td>TABLEUSERID</td>
<td>VARCHAR(128)</td>
<td>Table creator name related to the object this row describes</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>VARCHAR(128)</td>
<td>Table name related to the object this row describes</td>
</tr>
<tr>
<td>REC_CNT</td>
<td>SMALLINT</td>
<td>Number of records in the Old Objects Table used to describe this object</td>
</tr>
<tr>
<td>REC_SEQ</td>
<td>SMALLINT</td>
<td>The sequence number of this row in the REC_CNT records used to describe this object</td>
</tr>
<tr>
<td>REC_TYPE</td>
<td>CHAR(1)</td>
<td>T – table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D – compression dictionary</td>
</tr>
</tbody>
</table>
This table contains information about the ongoing jobs run by the product. For each run of an ongoing job, the table contains one type 1 row (indicated by the RECORDTYPE field). It might contain multiple type 2 rows (if any URIDs are open at the time a log scan was completed) or multiple type 3 rows (if the log records of data sharing members were excluded from the run because of lack of activity). Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 78 shows columns that make up the unique index key in bold face type.

**Table 78 Open unit of recovery table (ALPURID) (part 1 of 2)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be: ■ defined by the user ■ obtained from the user’s TSO prefix ■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>HANDLE</td>
<td>INTEGER</td>
<td>Unique identifier for each ongoing log scan within a work ID</td>
</tr>
</tbody>
</table>
Report file pointer table (ALPRPTF)

This table contains one row for each report file generated by a given log scan step. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 79 shows columns that make up the unique index key in bold face type.

Table 79  Report file pointer table (ALPRPTF) (part 1 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the report file data set</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(8)</td>
<td>Member name (if the file is a member of a PDS)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Work ID phase number</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Work ID step number that generated the data set</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>REPORTTYPE</td>
<td>SMALLINT</td>
<td>Type of report to generate:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Audit report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Detail report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Summary report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Commit report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Rollback report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Quiet Point report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – Anomaly report – UNDO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – Image Copy report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – Open Transactions report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – Data Capture Changes report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – Object Activity Summary report (All Activity)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – Backout Integrity report (Summary)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – Drop Recovery report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – Catalog Activity report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 – Commands report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 - Log Bytes report</td>
</tr>
<tr>
<td>ANOMALYTYPE</td>
<td>SMALLINT</td>
<td>0 – Produce minimal anomaly information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Produce maximum anomaly information</td>
</tr>
</tbody>
</table>
### Table 79  Report file pointer table (ALPRPTF) (part 3 of 3)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDERBY1</td>
<td>SMALLINT</td>
<td>First key to use for breaks when sorting:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Unit of recovery identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Plan name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Auth ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Connection type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Connection ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – Correlation ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Database name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – Table space name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – Table space DSNUM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – Table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – DBID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – PSID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – OBID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – Primary key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 – Activity (descending)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 – Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 – Object type (Catalog Activity report only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 – Activity type (Catalog Activity report only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 – Member ID (data sharing only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 – Subsystem ID (SSID, data sharing only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 – Member name (data sharing only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 – Create RBA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24 – Command</td>
</tr>
<tr>
<td>ORDERBY2</td>
<td>SMALLINT</td>
<td>Second key to use for breaks when sorting – for values, see ORDERBY1 field</td>
</tr>
<tr>
<td>ORDERBY3</td>
<td>SMALLINT</td>
<td>Third key to use for breaks when sorting – for values, see ORDERBY1 field</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>FILETIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time created (for format, see page 387)</td>
</tr>
<tr>
<td>VALIDFILE</td>
<td>SMALLINT</td>
<td>0 – Data set is missing from operating system’s catalog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Data set is valid</td>
</tr>
<tr>
<td>PAGECOUNT</td>
<td>INTEGER</td>
<td>Number of pages in the report</td>
</tr>
<tr>
<td>LINCOUNT</td>
<td>INTEGER</td>
<td>Number of detail lines in the report</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the file</td>
</tr>
<tr>
<td>FILETS</td>
<td>TIMESTAMP</td>
<td>Date/time created</td>
</tr>
</tbody>
</table>
Report template table (ALPRTMPL)

This table contains at least one row for each report template that the product stores in the Repository. When required, the product uses multiple rows to store templates.

Table 80 shows columns that make up the unique index key in bold face type.

Table 80 Report template table (ALPRTMPL) (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOR</td>
<td>CHAR(8)</td>
<td>Qualifier portion of report template ID (frequently the user ID or creator of the template)</td>
</tr>
<tr>
<td>NAME</td>
<td>CHAR(18)</td>
<td>Template name portion of report template ID</td>
</tr>
<tr>
<td>RECORD_SEQUENCE</td>
<td>SMALLINT</td>
<td>Sequence number of current row within template (used when storage of template requires multiple rows)</td>
</tr>
<tr>
<td>REPORTTYPE</td>
<td>SMALLINT</td>
<td>Type of report defined in template:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Audit report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Summary report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Detail report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Commit report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Rollback report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – Quiet Point report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Image Copy report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – Open Transactions report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – Backout Integrity report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – Data Capture Changes report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – Object Activity Summary report (All Activity)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – Backout Integrity report (Summary)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – Drop Recovery report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – Catalog Activity report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 - Command report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 - Log Bytes report</td>
</tr>
<tr>
<td>ACCESSMODE</td>
<td>SMALLINT</td>
<td>0 – Universal access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – All users can read; only creator can modify</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Template cannot be viewed or updated by anyone other than the creator of the template</td>
</tr>
<tr>
<td>TOTAL_LENGTH</td>
<td>INT</td>
<td>Length of data stored in template (including multiple rows)</td>
</tr>
<tr>
<td>ALPVERSION</td>
<td>CHAR(6)</td>
<td>Version of the product that created the template (format: vvrrmm, example: 090100)</td>
</tr>
<tr>
<td>TSOID</td>
<td>CHAR(8)</td>
<td>TSO ID of the owner of the template (used for security checking)</td>
</tr>
<tr>
<td>CREATEDBY</td>
<td>CHAR(8)</td>
<td>TSO user ID of the individual creating the template</td>
</tr>
<tr>
<td>CREATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time work ID was created</td>
</tr>
<tr>
<td>UPDATEDBY</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the individual updating the template</td>
</tr>
<tr>
<td>UPDATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time work ID was last updated</td>
</tr>
<tr>
<td>DSNMAME</td>
<td>CHAR(54)</td>
<td>Data set name related to template (valued when template is imported from external data set)</td>
</tr>
</tbody>
</table>
Security exception history table (ALPSECH)

This table contains a row for each occasion when a Log Master job attempted to access DB2 data, but encountered security exceptions. The row contains information about the job that attempted access, the table that the job attempted to access, and the number of records that were suppressed in the job output. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. For those columns that are overloaded, subsequent tables define the individual fields within the overloaded columns.

Table 81 on page 404 shows columns that make up the unique index key in bold face type.

### Table 80  Report template table (ALPRTMPL) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(64)</td>
<td>Text description of template obtained from product online interface</td>
</tr>
<tr>
<td>DATA</td>
<td>VARCHAR(32400)</td>
<td>Portion of report template data (The product stores the report template in the form of XML tags and attribute values. When required, the product assembles a template from multiple rows, based on the content of the RECORD_SEQUENCE field.)</td>
</tr>
</tbody>
</table>

### Security exception history table (ALPSECH)

Table 81 Security exception history table (ALPSECH) (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| USERID          | CHAR(8)       | User ID associated with the work ID. This value can be:  
> defined by the user
> obtained from the user’s TSO prefix
> obtained from the user’s TSO ID |
| WORKIDNAME      | CHAR(18)      | Unique name associated with the work ID that attempted access                                                                           |
| JOBNAME         | CHAR(8)       | Operating system job name of the job that attempted access                                                                             |
| JOBNUM          | INTEGER       | Operating system job number of the job that attempted access                                                                           |
| RUN_TIMESTAMP   | CHAR(10)      | Date and time that the accessing job completed                                                                                        |
| RUNSEQNUM       | INTEGER       | Execution number of the job that attempted access. The product assigns this value sequentially starting with 1.                          |
| PRIMARY_AUTHID  | CHAR(8)       | Authorization ID associated with job that attempted access                                                                               |
| TABLE_CREATOR   | VARCHAR(128)  | User ID of the user who created the table against which security exceptions were encountered                                              |
| TABLE_NAME      | VARCHAR(128)  | Name of the table against which security exceptions were encountered                                                                     |
This table contains information that Log Master uses to respond to SQL codes as it executes generated SQL. The table contains a row for each SQL code specification defined through the online interface. For more information about how to set up SQL code handling through the online interface, see the chapter on executing SQL in the Log Master for DB2 User Guide.

Table 82 on page 405 shows columns that make up the unique index key in bold face type.

### SQL codes handler table (ALPSQLC)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>LINENUM</td>
<td>SMALLINT</td>
<td>Entry order sequence number</td>
</tr>
<tr>
<td>UPDTYPE</td>
<td>SMALLINT</td>
<td>Type of SQL statement:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – ALL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – INSERT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – DELETE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – UPDATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – DDL</td>
</tr>
<tr>
<td>SQLCOND</td>
<td>SMALLINT</td>
<td>SQL condition to handle:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Any positive SQL code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Any negative SQL code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Specific SQLCODE (see CODE field)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – SQL Warnings (see CODE field)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Multiple updates/deletes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – SYNERR</td>
</tr>
</tbody>
</table>

### Table 82 Security exception history table (ALPSECH) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECS_SUPPRESSED</td>
<td>INTEGER</td>
<td>Number of log records relating to the security exception that were suppressed in the job output</td>
</tr>
<tr>
<td>RUN_TS</td>
<td>TIMESTAMP</td>
<td>Date and time that the accessing job completed</td>
</tr>
</tbody>
</table>
This table contains a row for each SQL file generated by the product. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. Table 83 shows columns that make up the unique index key in bold face type.

Table 83  SQL file pointer table (ALPSQLF) (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNAME</td>
<td>CHAR(44)</td>
<td>Data set name of the SQL file data set</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(8)</td>
<td>Member name (if the file is a member of a PDS)</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Work ID phase number</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Work ID step number that generated data set</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>FILETIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time created (for format, see page 387)</td>
</tr>
<tr>
<td>VALIDFILE</td>
<td>SMALLINT</td>
<td>0 – Data set is missing from operating system’s catalog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Data set is valid</td>
</tr>
<tr>
<td>INSERTCOUNT</td>
<td>INTEGER</td>
<td>Number of SQL INSERT statements in the file</td>
</tr>
<tr>
<td>UPDATECOUNT</td>
<td>INTEGER</td>
<td>Number of SQL UPDATE statements in the file</td>
</tr>
</tbody>
</table>
### Table 83  SQL file pointer table (ALPSQLF) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETECOUNT</td>
<td>INTEGER</td>
<td>Number of SQL DELETE statements in the file</td>
</tr>
<tr>
<td>UPDATERICOUNT</td>
<td>INTEGER</td>
<td>Number of SQL INSERT statements due to referential integrity</td>
</tr>
<tr>
<td>DELETERICOUNT</td>
<td>INTEGER</td>
<td>Number of SQL DELETE statements due to referential integrity</td>
</tr>
<tr>
<td>UNDOREDOFLAG</td>
<td>SMALLINT</td>
<td>0 – UNDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – REDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – MIGRATE SQL</td>
</tr>
<tr>
<td>REDORBA</td>
<td>VARCHAR(10)</td>
<td>Not used</td>
</tr>
<tr>
<td>SQLWHEREMODE</td>
<td>SMALLINT</td>
<td>If an SQL output file, this flag determines the level of WHERE clause generation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Primary key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Specified index</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – All columns</td>
</tr>
<tr>
<td>INCLCHANGED</td>
<td>SMALLINT</td>
<td>0 – Changed columns not included</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Changed columns included</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>SMALLINT</td>
<td>Indicator that defines the format of dates within the data set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – USA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – EUR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – ISO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – JIS</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the file</td>
</tr>
<tr>
<td>INSERTTRCOUNT</td>
<td>INTEGER</td>
<td>Number of SQL INSERT statements due to trigger activity</td>
</tr>
<tr>
<td>UPDATETRCOUNT</td>
<td>INTEGER</td>
<td>Number of SQL UPDATE statements due to trigger activity</td>
</tr>
<tr>
<td>DELETETRCOUNT</td>
<td>INTEGER</td>
<td>Number of SQL DELETE statements due to trigger activity</td>
</tr>
<tr>
<td>FILETS</td>
<td>TIMESTAMP</td>
<td>Date/time created</td>
</tr>
</tbody>
</table>
**SQL index table (ALPIXSQ)**

This table contains a row for each alternate index defined within a given work ID. The product uses alternate indexes as it generates SQL statements.

Table 84 shows columns that make up the unique index key in bold face type.

---

**Table 84  SQL index table (ALPIXSQ)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this column contains ‘**** PROFILE ****’, the row has a global effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(independent of the work ID specified).</td>
</tr>
<tr>
<td>TABLEUSERID</td>
<td>VARCHAR(259)</td>
<td>User ID associated with the table</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>VARCHAR(259)</td>
<td>Name of the DB2 table that contains the column</td>
</tr>
<tr>
<td>INDEXUSERID</td>
<td>VARCHAR(259)</td>
<td>User ID associated with the table</td>
</tr>
<tr>
<td>INDEXNAME</td>
<td>VARCHAR(259)</td>
<td>Name of the index</td>
</tr>
<tr>
<td>CREATEDBY</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the individual creating the work ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To select alternate indexes, the product matches the user ID of the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>running the job (or the value specified with the TSOID keyword)</td>
</tr>
<tr>
<td>CREATEDTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/Time work ID was created</td>
</tr>
<tr>
<td>UPDATEDBY</td>
<td>CHAR(8)</td>
<td>TSO user ID or prefix of the individual updating the work ID</td>
</tr>
<tr>
<td>UPDATEDTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/Time work ID was last updated</td>
</tr>
<tr>
<td>CREATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time work ID was created</td>
</tr>
<tr>
<td>UPDATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time work ID was last updated</td>
</tr>
</tbody>
</table>

---

**SQL name translation table (ALPSQLX)**

This table contains SQL translation (SQLXLAT) information about database objects. The product uses this information as it generates SQL and during a drop recovery action.

Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. For those columns that are overloaded, subsequent tables define the individual fields within the overloaded columns.

Table 85 shows columns that make up the unique index key in bold face type.
This table contains a row for each defined work ID. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 86 shows columns that make up the unique index key in bold face type.
## Table 86  Work ID header table (ALPWHDR) (part 2 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZEMODE</td>
<td>SMALLINT</td>
<td>0 – Normal value, print analysis results 1 – Full run, no analysis results printed 2 – Analyze only</td>
</tr>
<tr>
<td>ACCESSMODE</td>
<td>SMALLINT</td>
<td>0 – Universal access 1 – All users can read; only creator can modify 2 – Work ID cannot be viewed or updated by anyone other than the creator of the work ID</td>
</tr>
<tr>
<td>FILTERREL</td>
<td>SMALLINT</td>
<td>0 – Multiple filters are related with the AND operator 1 – Multiple filters are related with the OR operator</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Text description of the work ID</td>
</tr>
<tr>
<td>DATAMEM</td>
<td>INTEGER</td>
<td>The amount of memory the product uses when allocating for Virtual Storage Access Method (VSAM) files</td>
</tr>
<tr>
<td>INDXMEM</td>
<td>INTEGER</td>
<td>Reserved</td>
</tr>
<tr>
<td>ALLOCUNIT</td>
<td>SMALLINT</td>
<td>Code indicating type of allocation: 0 – Tracks 1 – Cylinders</td>
</tr>
<tr>
<td>PRISPACE</td>
<td>INTEGER</td>
<td>Primary DASD allocation</td>
</tr>
<tr>
<td>SECSPACE</td>
<td>INTEGER</td>
<td>Secondary DASD allocation</td>
</tr>
<tr>
<td>VOLSERS</td>
<td>VARCHAR(1784)</td>
<td>List of VOLSERs for work VSAM files</td>
</tr>
<tr>
<td>STORCLASS</td>
<td>CHAR(8)</td>
<td>Data Facility Storage Management Subsystem (DFSMS) storage class for VSAM files</td>
</tr>
<tr>
<td>CLUSTERHLQ</td>
<td>CHAR(44)</td>
<td>Prefix used to derive the cluster name for VSAM files</td>
</tr>
<tr>
<td>DATAHLQ</td>
<td>CHAR(44)</td>
<td>Prefix used to derive the data portion name for VSAM files</td>
</tr>
<tr>
<td>MAINSIZE</td>
<td>INTEGER</td>
<td>The maximum amount of memory the sort routine uses</td>
</tr>
<tr>
<td>VSCORET</td>
<td>INTEGER</td>
<td>Associated with MAINSIZE</td>
</tr>
<tr>
<td>RESINV</td>
<td>INTEGER</td>
<td>The amount of memory reserved below the 16-MB line by the sort routine to allow for input/output processing</td>
</tr>
<tr>
<td>PSTORCLASS</td>
<td>CHAR(8)</td>
<td>Storage class to use for dynamic allocation of sort data sets</td>
</tr>
<tr>
<td>PUNIT</td>
<td>CHAR(8)</td>
<td>Unit name the product uses for dynamic allocation of sort data sets</td>
</tr>
<tr>
<td>SORTDYN</td>
<td>SMALLINT</td>
<td>Indicates whether to use the sort dynamic allocation rules</td>
</tr>
<tr>
<td>WORKNUM</td>
<td>SMALLINT</td>
<td>The number of sort work data sets to use</td>
</tr>
<tr>
<td>WSTORCLASS</td>
<td>CHAR(8)</td>
<td>The storage class to use for data set allocation</td>
</tr>
<tr>
<td>WUNIT</td>
<td>CHAR(8)</td>
<td>The unit name for dynamic allocations</td>
</tr>
<tr>
<td>PROCESSDB2LOG</td>
<td>SMALLINT</td>
<td>Indicates whether to process DB2 catalog records</td>
</tr>
<tr>
<td>EXECMODE</td>
<td>SMALLINT</td>
<td>Determines whether overtime processing is allowed</td>
</tr>
<tr>
<td>EARLYRECALL</td>
<td>SMALLINT</td>
<td>Code indicating if the product should attempt early recall of migrated data sets: 0 – No 1 – Yes</td>
</tr>
</tbody>
</table>
Table 86  Work ID header table (ALPWHDR) (part 3 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMIGRATE</td>
<td>SMALLINT</td>
<td>Code indicating if the product should remigrate recalled data sets:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td>MAXTASKS</td>
<td>SMALLINT</td>
<td>Maximum number of recall tasks that the product schedules at any one time</td>
</tr>
<tr>
<td>MT_PERMEMBER</td>
<td>SMALLINT</td>
<td>Code indicating if the value in MAXTASKS is per data sharing member:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td>DASD_DATASETS</td>
<td>SMALLINT</td>
<td>Maximum number of recalled data sets that can reside on DASD at any one time</td>
</tr>
<tr>
<td>DD_PERMEMBER</td>
<td>SMALLINT</td>
<td>Code indicating if the value in DASD_DATASETS is per data sharing member:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td>USELGRNG</td>
<td>SMALLINT</td>
<td>Code indicating if the product includes the USLGRNG keyword in the generated batch syntax:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not include keyword</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Include keyword with YES value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Include keyword with NO value</td>
</tr>
<tr>
<td>MINLOGPT</td>
<td>SMALLINT</td>
<td>Code indicating if the product includes the MINLOGPT keyword in the generated batch syntax:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not include keyword</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Include keyword with YES value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Include keyword with NO value</td>
</tr>
<tr>
<td>DO_PITS</td>
<td>SMALLINT</td>
<td>Indicates whether to process log records contained inside of a Point-in-time (PIT) recovery range</td>
</tr>
<tr>
<td>reserved</td>
<td>CHAR(80)</td>
<td>Reserved</td>
</tr>
<tr>
<td>MGMTCLASS</td>
<td>CHAR(8)</td>
<td>DFSMS management class for key store overflow files</td>
</tr>
<tr>
<td>SORTNOHIST</td>
<td>SMALLINT</td>
<td>Code indicating whether the product saves history information for this run of the current work ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not save history information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Save history information</td>
</tr>
<tr>
<td>HEADERFILESZ</td>
<td>SMALLINT</td>
<td>Code indicating the selected method for obtaining or calculating the sort file size estimate:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 – None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Log bytes (total volume of log records read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Use history</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Estimate (use estimate value supplied by user)</td>
</tr>
<tr>
<td>SORTFILESIZE</td>
<td>INTEGER</td>
<td>Estimated file size value entered by user</td>
</tr>
<tr>
<td>AVERAGERECLength</td>
<td>INTEGER</td>
<td>Average record length entered by user</td>
</tr>
</tbody>
</table>
### Table 86  Work ID header table (ALPWHDR) (part 4 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINETYPE</td>
<td>SMALLINT</td>
<td>Code indicating the input source for the work ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Logical log files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – DB2 subsystem (BDS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Individual DB2 log files</td>
</tr>
<tr>
<td>reserved</td>
<td>SMALLINT</td>
<td>Reserved</td>
</tr>
<tr>
<td>LOBPREFIX</td>
<td>CHAR(44)</td>
<td>Prefix that product uses to form names of LOB VSAM files</td>
</tr>
<tr>
<td>LOBDUPDS</td>
<td>SMALLINT</td>
<td>Code indicating how the product responds to duplicate data set condition when creating LOB VSAM files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Informational message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Warning message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Error message</td>
</tr>
<tr>
<td>LOBALLOCU</td>
<td>SMALLINT</td>
<td>Code indicating allocation unit for LOB VSAM files:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Tracks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Cylinders</td>
</tr>
<tr>
<td>LOBPRISPACE</td>
<td>INTEGER</td>
<td>Primary DASD allocation for LOB VSAM files</td>
</tr>
<tr>
<td>LOBSECSPACE</td>
<td>INTEGER</td>
<td>Secondary DASD allocation for LOB VSAM files</td>
</tr>
<tr>
<td>LOBMGMTCLASS</td>
<td>CHAR(8)</td>
<td>DFSMS management class for LOB VSAM files</td>
</tr>
<tr>
<td>LOBSTORCLASS</td>
<td>CHAR(8)</td>
<td>DFSMS storage class for LOB VSAM files</td>
</tr>
<tr>
<td>LOBDATACLASS</td>
<td>CHAR(8)</td>
<td>DFSMS data class for LOB VSAM files</td>
</tr>
<tr>
<td>LOBDATALIMIT</td>
<td>INTEGER</td>
<td>Maximum number of LOB VSAM files that product can create for data from one LOB column (or one partition of a LOB column)</td>
</tr>
<tr>
<td>LOBVOLLIST</td>
<td>VARCHAR(42)</td>
<td>List of VOLSERs for LOB VSAM files</td>
</tr>
<tr>
<td>CREATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time work ID was created</td>
</tr>
<tr>
<td>UPDATEDTS</td>
<td>TIMESTAMP</td>
<td>Date/Time work ID was last updated</td>
</tr>
<tr>
<td>XMLPREFIX</td>
<td>CHAR(44)</td>
<td>Prefix that product uses to form names of XML VSAM files</td>
</tr>
<tr>
<td>XMLDUPDS</td>
<td>SMALLINT</td>
<td>Code indicating how the product responds to duplicate data set condition when creating XML VSAM files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Informational message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Warning message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Error message</td>
</tr>
<tr>
<td>XMLALLOCU</td>
<td>SMALLINT</td>
<td>Code indicating allocation unit for XML VSAM files:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Tracks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Cylinders</td>
</tr>
<tr>
<td>XMLPRISPACE</td>
<td>INTEGER</td>
<td>Primary DASD allocation for XML VSAM files</td>
</tr>
<tr>
<td>XMLSECSPACE</td>
<td>INTEGER</td>
<td>Secondary DASD allocation for XML VSAM files</td>
</tr>
<tr>
<td>XMLMGMTCLASS</td>
<td>CHAR(8)</td>
<td>DFSMS management class for XML VSAM files</td>
</tr>
<tr>
<td>XMLSTORCLASS</td>
<td>CHAR(8)</td>
<td>DFSMS storage class for XML VSAM files</td>
</tr>
<tr>
<td>XMLDATACLASS</td>
<td>CHAR(8)</td>
<td>DFSMS data class for XML VSAM files</td>
</tr>
<tr>
<td>XMLDATALIMIT</td>
<td>INTEGER</td>
<td>Maximum number of XML VSAM files product can create for data from one XML column (or partition of XML column)</td>
</tr>
<tr>
<td>XMLVOLLIST</td>
<td>VARCHAR(42)</td>
<td>List of VOLSERs for XML VSAM files</td>
</tr>
</tbody>
</table>
Table 86  Work ID header table (ALPWHDR) (part 5 of 5)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATACLASS</td>
<td>CHAR(8)</td>
<td>DFSMS data class for key store overflow files</td>
</tr>
<tr>
<td>ALPVERSION</td>
<td>CHAR(6)</td>
<td>version of the product that created or updated a repository object. This field is empty if the records were created prior to Log Master version 11.1.</td>
</tr>
</tbody>
</table>

**Work ID history table (ALPWHIS)**

This table contains a row for each run of a work ID. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 87 shows columns that make up the unique index key in bold face type.

Table 87  Work ID history table (ALPWHIS) (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be: ▪ defined by the user ▪ obtained from the user’s TSO prefix ▪ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Work ID phase identification</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>JOBTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time job completed (for format, see page 387)</td>
</tr>
<tr>
<td>STATUS</td>
<td>SMALLINT</td>
<td>Job completion status: ▪ 0 – Job not successfully completed ▪ 1 – Job completed successfully</td>
</tr>
<tr>
<td>RETURNCODE</td>
<td>INTEGER</td>
<td>Job return code value</td>
</tr>
<tr>
<td>SYSTEMCC</td>
<td>INTEGER</td>
<td>System completion code</td>
</tr>
<tr>
<td>LASTSTEPNUM</td>
<td>SMALLINT</td>
<td>Last step number run</td>
</tr>
<tr>
<td>FC_KSCOUNT</td>
<td>INTEGER</td>
<td>Amount of storage used by the FC (forward completion) key store memory area</td>
</tr>
<tr>
<td>BC_KSCOUNT</td>
<td>INTEGER</td>
<td>Amount of storage used by the BC (backward completion) key store memory area</td>
</tr>
<tr>
<td>LR_KSCOUNT</td>
<td>INTEGER</td>
<td>Amount of storage used by the LR (log record) key store</td>
</tr>
<tr>
<td>UR_KSCOUNT</td>
<td>INTEGER</td>
<td>Amount of storage used by the UR (URID) key store</td>
</tr>
</tbody>
</table>
**Work ID input file definitions table (ALPIFIL)**

This table contains a row for each file defined as input to a work ID step. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 88 shows columns that make up the unique index key in bold face type.

**Table 87 Work ID history table (ALPWHIS) (part 2 of 2)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN_KSCOUNT</td>
<td>INTEGER</td>
<td>Amount of storage used by the AN (anomaly) key store</td>
</tr>
<tr>
<td>START_TIME</td>
<td>CHAR(10)</td>
<td>Start point of this work ID</td>
</tr>
<tr>
<td>END_TIME</td>
<td>CHAR(10)</td>
<td>End point of this work ID</td>
</tr>
<tr>
<td>START_RBA</td>
<td>VARCHAR(10)</td>
<td>Starting RBA/LRSN scanned</td>
</tr>
<tr>
<td>END_RBA</td>
<td>VARCHAR(10)</td>
<td>Ending RBA/LRSN scanned</td>
</tr>
<tr>
<td>JOBTS</td>
<td>TIMESTAMP</td>
<td>Date/time job completed</td>
</tr>
<tr>
<td>START_TS</td>
<td>TIMESTAMP</td>
<td>Start point of this work ID</td>
</tr>
<tr>
<td>END_TS</td>
<td>TIMESTAMP</td>
<td>End point of this work ID</td>
</tr>
</tbody>
</table>

**Work ID input file definitions table (ALPIFIL)**

This table contains a row for each file defined as input to a work ID step. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 88 shows columns that make up the unique index key in bold face type.

**Table 88 Work ID input file definitions table (ALPIFIL) (part 1 of 2)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user's TSO prefix</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Sequential step number within the work ID</td>
</tr>
<tr>
<td>LINENUM</td>
<td>SMALLINT</td>
<td>File sequence number within the work ID</td>
</tr>
<tr>
<td>LINETYPE</td>
<td>SMALLINT</td>
<td>Value indicating the type of input file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Input logical log file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – DB2 subsystem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Input DB2 log file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – SQL input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Old objects data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Input image copy (not cataloged in DB2)</td>
</tr>
<tr>
<td>SUBSYSTEMID</td>
<td>CHAR(4)</td>
<td>DB2 subsystem</td>
</tr>
<tr>
<td>DSNAME</td>
<td>CHAR(48)</td>
<td>Data set name</td>
</tr>
<tr>
<td>UNIT</td>
<td>CHAR(8)</td>
<td>Unit value (for uncataloged DB2 log files)</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(8)</td>
<td>Member name (if the file is a member of a PDS)</td>
</tr>
<tr>
<td>VOLSERS</td>
<td>VARCHAR(1784)</td>
<td>List of VOLSERS for uncataloged DB2 log files</td>
</tr>
</tbody>
</table>
## LINEDATA field: Input logical log file type

This table shows the layout of the LINEDATA field in the Work ID Input File Definitions Table (Table 88 on page 414) when the LINETYPE value specifies an input logical log file.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQNO</td>
<td>SMALLINT</td>
<td>Volume sequence number</td>
</tr>
<tr>
<td>LINEDATA</td>
<td>VARCHAR(254)</td>
<td>Data dependent on the LINETYPE requested (See Table 89 on page 415 through Table 91 on page 416)</td>
</tr>
<tr>
<td>ALPVERSION</td>
<td>CHAR(6)</td>
<td>Version of the product that processed the input file (format: vvvvmm, example: 090100)</td>
</tr>
</tbody>
</table>

### Table 89 LINEDATA field: Input logical log file type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTTYPE</td>
<td>CHAR(1)</td>
<td>Code indicating the input source of the logical log records:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D – Logical log input is from a data set</td>
</tr>
<tr>
<td>not used</td>
<td>CHAR(101)</td>
<td>Not used</td>
</tr>
</tbody>
</table>

## LINEDATA field: Input DB2 log file type

This table shows the layout of the LINEDATA field in the Work ID Input File Definitions Table (Table 88 on page 414) when the LINETYPE value specifies an input DB2 log file.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTTYPE</td>
<td>CHAR(1)</td>
<td>Code indicating the input source of the logical log records:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D – DB2 log input is from a data set</td>
</tr>
<tr>
<td>QMGR</td>
<td>CHAR(49)</td>
<td>Not used</td>
</tr>
<tr>
<td>CORRID</td>
<td>CHAR(25)</td>
<td>Not used</td>
</tr>
<tr>
<td>PURGE</td>
<td>CHAR(9)</td>
<td>Not used</td>
</tr>
<tr>
<td>WAITINTERVAL</td>
<td>CHAR(9)</td>
<td>Not used</td>
</tr>
<tr>
<td>DB2MEMBERNAME</td>
<td>CHAR(9)</td>
<td>Name of the data sharing member that the input DB2 log file contains log records from</td>
</tr>
</tbody>
</table>
LINEDATA field: Input (uncataloged) image copy file type

This table shows the layout of the LINEDATA field in the Work ID Input File Definitions Table (Table 88 on page 414) when the LINETYPE value specifies an input image copy.

Table 91  LINEDATA field: Input (uncataloged) image copy file type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td>CHAR(25)</td>
<td>Database name (null terminated, no trailing spaces)</td>
</tr>
<tr>
<td>TSNAME</td>
<td>CHAR(25)</td>
<td>Table space name (null terminated, no trailing spaces)</td>
</tr>
<tr>
<td>ICTYPE</td>
<td>CHAR(1)</td>
<td>Code indicating the type of image copy:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F – Full image copy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I – Incremental image copy</td>
</tr>
<tr>
<td>INLINEE</td>
<td>CHAR(1)</td>
<td>Code indicating whether this is an inline image copy:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘’ – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E – Yes</td>
</tr>
<tr>
<td>REMAP</td>
<td>CHAR(1)</td>
<td>Code indicating whether the image copy follows a run of a utility that reorganizes Row IDs:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘’ – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y – Yes</td>
</tr>
<tr>
<td>IMRBA</td>
<td>CHAR(10) FOR BIT DATA</td>
<td>Starting RBA/LRSN for image copy</td>
</tr>
<tr>
<td>DSNUM</td>
<td>SMALLINT</td>
<td>Data set number</td>
</tr>
</tbody>
</table>

Work ID log scan history table (ALPWLSH)

The product creates and maintains a row in this table each time it runs a log scan. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute.

Table 92 shows columns that make up the unique index key in bold face type.

Table 92  Work ID log scan history table (ALPWLSH) (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Work ID phase identification</td>
</tr>
<tr>
<td>RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
</tbody>
</table>
### Table 92  Work ID log scan history table (ALPWLSH) (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Sequence number of the log scan step within the current work ID</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>CHAR(8)</td>
<td>Operating system job name</td>
</tr>
<tr>
<td>JOBNUM</td>
<td>INTEGER</td>
<td>Operating system job number</td>
</tr>
<tr>
<td>JOBTIMESTAMP</td>
<td>CHAR(10)</td>
<td>Date/time job completed (for format, see page 387)</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>INTEGER</td>
<td>Total number of log records processed during log scan</td>
</tr>
<tr>
<td>RECORD_BYTES</td>
<td>INTEGER</td>
<td>Total number of bytes processed during log scan</td>
</tr>
<tr>
<td>LASTSTEPNUM</td>
<td>SMALLINT</td>
<td>The STEPNUM value of the last time this combination of user ID and work ID was run</td>
</tr>
<tr>
<td>AVG_RECORD_SIZE</td>
<td>INTEGER</td>
<td>Average size of the log records processed during the log scan</td>
</tr>
<tr>
<td>ACTUAL_FROM_RBA</td>
<td>VARCHAR(10)</td>
<td>The actual RBA/LSRN used as the start point of the log scan (as opposed to what was specified through online interface or in batch syntax)</td>
</tr>
<tr>
<td>ACTUAL_TO_RBA</td>
<td>VARCHAR(10)</td>
<td>The actual RBA/LSRN used as the end point of the log scan (as opposed to what was specified through online interface or in batch syntax)</td>
</tr>
<tr>
<td>ONGOING_HANDLE</td>
<td>INTEGER</td>
<td>Unique identifier for each URID generated by the job run</td>
</tr>
<tr>
<td>ONGOING_RUNSEQNUM</td>
<td>INTEGER</td>
<td>Execution number. The product assigns this value sequentially starting with 1.</td>
</tr>
<tr>
<td>REDORBA</td>
<td>VARCHAR(10)</td>
<td>RBA/LSRN of recovery point for REDO SQL</td>
</tr>
<tr>
<td>RECORD_DENSITY</td>
<td>SMALLINT</td>
<td>Ratio (expressed as a percentage) of log records selected to log records read during the log scan</td>
</tr>
<tr>
<td>JOBTS</td>
<td>TIMESTAMP</td>
<td>Date/time job completed</td>
</tr>
</tbody>
</table>

### Work ID output file definitions table (ALPOFIL)

This table contains a row for each output file that the product produces as it processes a log scan step or SQL execution step. If the output data set is defined as old or if a PDS member name is supplied, only the data set name and member name fields are valid. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. For those columns that are overloaded, subsequent tables define the individual fields within the overloaded columns.

Table 93 on page 418 shows columns that make up the unique index key in bold face type.
### Table 93  Work ID output file definitions table (ALPOFIL)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Step number within the work ID</td>
</tr>
<tr>
<td>LINENUM</td>
<td>SMALLINT</td>
<td>File sequence number</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(65)</td>
<td>Description of file</td>
</tr>
<tr>
<td>LINETYPE</td>
<td>SMALLINT</td>
<td>Value indicating the type of output file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – REDO anomaly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – UNDO anomaly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Logical log data file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Logical log control file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – UNDO SQL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – REDO SQL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – MIGRATE SQL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Report file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – Load data file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – Load control file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – UNDO SQL template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – REDO SQL template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – MIGRATE SQL template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – UNDO DDL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – MIGRATE DDL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 – OUTCOPY file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(parameters for subsequent image copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 – Recovery control file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 – Post-recovery SQL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 – Post-recovery rebind file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 – Drop recovery SYSIN syntax</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 – DSN1COPY SYSIN syntax</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(drop recovery of single table space)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 – LOB VSAM file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 – XML VSAM file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 – XML string file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24 – Post-recovery CHECK file</td>
</tr>
<tr>
<td>LINEDATA2</td>
<td>VARCHAR(538)</td>
<td>Data dependent on the LINETYPE requested (for possible formats, see Table 94 on page 419 through Table 101 on page 428)</td>
</tr>
<tr>
<td>OUTPUTTYPE</td>
<td>SMALLINT</td>
<td>0 – Output written to a data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Output written to a SYSOUT data set (reports only)</td>
</tr>
<tr>
<td>OUTPUTDATA</td>
<td>VARCHAR(286)</td>
<td>Output data set or SYSOUT definition depending on the OUTPUTTYPE value (for the possible formats, see Table 103 on page 430 and Table 104 on page 431)</td>
</tr>
<tr>
<td>VOLSERS</td>
<td>VARCHAR(1784)</td>
<td>A series of VOLSERS, separated by commas (used for image copy output)</td>
</tr>
<tr>
<td>ALPVERSION</td>
<td>CHAR(6)</td>
<td>Version of the product that wrote the output file (format: vvrmnm, example: 090100)</td>
</tr>
</tbody>
</table>
**LINEDATA2 field: Logical log file type**

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies logical log output.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECFMVB</td>
<td>BOOLEAN</td>
<td>0 – Output logical log is variable blocked spanned 1 – Output logical log is variable blocked</td>
</tr>
<tr>
<td>DATEFMTDB2I</td>
<td>BOOLEAN</td>
<td>0 – Dates are in SAS character format (YYYY/MM/DD/hh:mm:ss:nnnnn) When running on DB2 Version 10 and later, the product supports precision timestamps up to 12 digits, and inclusion of a time zone in the timestamp (YYYY-MM-DD- HH:MM:SS.nnnnnnnnnnnn±HH:MM). 1 – Dates are in DB2 format (YYYY-MM-DD-hh.mm.ss.nnnnn)</td>
</tr>
<tr>
<td>EXPANDVAR</td>
<td>BOOLEAN</td>
<td>0 – VARCHAR fields in the output logical log are not expanded 1 – VARCHAR fields in the output logical log are expanded (filled) to their full length.</td>
</tr>
<tr>
<td>ABORTEDTRANS</td>
<td>INTEGER</td>
<td>0 – Include rollback value is NO. 1 – Include rollback value is YES. 2 – Include rollback value is ONLY</td>
</tr>
<tr>
<td>DDLOBJECTS</td>
<td>CHAR(1)</td>
<td>N – Output logical log does not contain DDL information. Y – Output logical log contains DDL information.</td>
</tr>
<tr>
<td>INCLUDELOBS</td>
<td>INTEGER</td>
<td>0 – Output logical log does not contain LOB data. 1 – Output logical log contains LOB data. 2 – Reserved 3 – Output logical log contains INLINE LOB data.</td>
</tr>
<tr>
<td>INCLUDEXML</td>
<td>BOOLEAN</td>
<td>0 – Output logical log does not contain XML data. 1 – Output logical log contains XML data.</td>
</tr>
<tr>
<td>INCLUDECMDS</td>
<td>BOOLEAN</td>
<td>0 – Output logical log does not contain DB2 command information. 1 – Output logical log contains DB2 command information.</td>
</tr>
</tbody>
</table>
## LINEDATA2 field: SQL file type

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies SQL output.

### Table 95  LINEDATA2 field: SQL file type  (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEREMODE</td>
<td>INTEGER</td>
<td>Determines the level of WHERE clause generation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Primary key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Specified index</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – All columns</td>
</tr>
<tr>
<td>BIT FLAGS</td>
<td>CHAR(4)</td>
<td>Single word defined as a series of <em>individual bit flags</em>, describing various keywords in the SQL syntax. Flags are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>listed in order using the format [byte position:bit position]. If the bit flag is set to 1, the product takes the action listed, otherwise, no action is taken</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:1]– Changed columns are included in the generated WHERE clause</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:2]– Comments are generated and included in generated SQL (VERBOSE output)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:3]– Rows changed as a result of referential integrity constraints are included in the generated UNDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:4]– Rows changed as a result of referential integrity constraints are included in the generated REDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:5]– Rows changed as a result of referential integrity constraints are included in the generated MIGRATE SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:6]– The product automatically executes the UNDO SQL after it is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:7]– All columns in the row are specified in the generated SQL, regardless of whether the value in the column is changed (UPDATE ALL COLUMNS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:8]– The product interprets the commit frequency as a number of statements (as opposed to transactions)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:1]– DDL statements are included in the generated UNDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:2]– DDL statements are included in the generated REDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:3]– DDL statements are included in the generated MIGRATE SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:4]– UNDO SQL output record format is set to VB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:5]– REDO SQL output record format is set to VB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:6]– MIGRATE SQL output record format is set to VB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:7]– Rows changed as a result of trigger activity are included in generated UNDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:8]– Rows changed as a result of trigger activity are included in generated REDO SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:1]– Rows changed as a result of trigger activity are included in generated MIGRATE SQL</td>
</tr>
<tr>
<td>COMMITCOUNT</td>
<td>SMALLINT</td>
<td>Number of statements or URIDs (transactions) between COMMITS as specified through the online interface or in batch syntax</td>
</tr>
</tbody>
</table>
Table 95  LINEDATA2 field: SQL file type (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDOABORT</td>
<td>INTEGER</td>
<td>Code indicating include rollback value for UNDO SQL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Only</td>
</tr>
<tr>
<td>REDOABORT</td>
<td>INTEGER</td>
<td>Code indicating include rollback value for REDO SQL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Only</td>
</tr>
<tr>
<td>MIGRATEABORT</td>
<td>INTEGER</td>
<td>Code indicating include rollback value for MIGRATE SQL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Only</td>
</tr>
<tr>
<td>SETSQLID</td>
<td>CHAR(9)</td>
<td>Authorization ID used to execute the generated SQL</td>
</tr>
<tr>
<td>INCLUDELOBS</td>
<td>BOOLEAN</td>
<td>0 – Output SQL file does not contain LOB data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Output SQL file contains LOB data</td>
</tr>
<tr>
<td>INCLUDEXML</td>
<td>BOOLEAN</td>
<td>0 – Output SQL file does not contain XML data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Output SQL file contains XML data</td>
</tr>
<tr>
<td>Bit flags</td>
<td>CHAR(4)</td>
<td>Single word defined as a series of individual bit flags,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For Bit Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>describing various keywords in the SQL syntax. Flags are listed in order</td>
</tr>
<tr>
<td></td>
<td></td>
<td>using the format [byte position:bit position]. If the bit flag is set to 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the product takes the action listed; otherwise, no action is taken.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:1] Include History rows in undo SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:2] Include History rows in redo SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:3] Include History rows in migrates SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:4] Exclude INSERT column Names</td>
</tr>
<tr>
<td>UNDOLOBSINCLUDED</td>
<td>INTEGER</td>
<td>Code indicating Include LOB for Undo SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Inline</td>
</tr>
</tbody>
</table>

**LINEDATA2 field: Report file type**

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies a report.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTTYPE</td>
<td>INTEGER</td>
<td>Type of report to generate:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Audit report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Summary report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Detail report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Commit report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Rollback report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – Quiet Point report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Image Copy report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – Open Transactions report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – Backout Integrity report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – Data Capture Changes report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – Object Activity Summary report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – Backout Integrity Summary report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – Drop Recovery report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 – Command report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 - Log Bytes report</td>
</tr>
<tr>
<td>ORDERBY1</td>
<td>INTEGER</td>
<td>First key to use for breaks when sorting:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Unit of recovery identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Plan name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Auth ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Connection type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Connection ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – Correlation ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Database name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – Table space name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – Table space DSNUM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – Table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – DBID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – PSID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – OBID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – Primary key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 – Activity (descending)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 – Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 – Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 – Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 – Data sharing member ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 – DB2 subsystem ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 – Data sharing member name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 – Command</td>
</tr>
<tr>
<td>ORDERBY2</td>
<td>INTEGER</td>
<td>Second key to use for breaks when sorting – for values, see ORDERBY1 field</td>
</tr>
<tr>
<td>ORDERBY3</td>
<td>INTEGER</td>
<td>Third key to use for breaks, sorting – or values, see ORDERBY1 field</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>BOOLEAN</td>
<td>F – Do not generate a quiesce into SYSCOPY for quiet point reporting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T – Generate a quiesce into SYSCOPY for quiet point reporting</td>
</tr>
</tbody>
</table>
### LINEDATA2 field: Load file type

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value is load file.

#### Table 96  LINEDATA2 field: Report file type (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| BIT FLAGS         | CHAR(4) FOR BIT DATA | Single word defined as a series of individual bit flags, describing various keywords in the report syntax. Flags are listed in order using the format [byte position:bit position]. If the bit flag is set to 1, the report characteristic is true:  
[1:1]–Backout Integrity report is a summary report  
[1:2]–Reserved  
[1:3]–Summary report is an Object Activity Summary report  
[1:4]–Summary report written in CSV format  
[1:5]–Summary report written in SDF format  
[1:6]–Summary report in CSV or SDF format includes subtotal lines and report total lines  
[1:7]–TEMPLATEINFO field is populated  
[1:8]–TEMPLATEINFO reflects data set |
| CMMTROLLTYPE      | INTEGER     | 0 – Not valued  
1 – Include rollback detail  
2 – Do not include rollback detail |
| ABORTEDTRANS      | INTEGER     | 0 – Include rollback value is NO  
1 – Include rollback value is YES  
2 – Include rollback value is ONLY |
| REDOTRANS         | BOOLEAN     | Code indicating whether report includes REDO information:  
0 – No REDO information (report includes records selected by filter)  
1 – REDO information included (report includes records within table spaces that are not selected by filter) |
| DURATION          | CHAR(7)     | Minimum duration for quiet ranges included in Quiet Point report (in hexadecimal format) |
| MAXURIDS          | INTEGER     | Maximum number of open transactions (URIDs) specified by user. A range of log can contain no more than this number of open URIDs to be included in Quiet Point report. If zero, report contains only completely quiet ranges. |
| TEMPLATEINFO      | CHAR(55)    | Empty if [1:7] of BIT FLAGS is false, otherwise Either  
ID of report template used to generate report  
Data set name of report template used to generate report |
# Table 97  LINEDATA2 field: Load file type (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT FLAGS</td>
<td>CHAR(4)</td>
<td>Single word defined as a series of <em>individual</em> bit flags. Flags are listed in order using the format [byte position:bit position].</td>
</tr>
<tr>
<td></td>
<td>FOR BIT DATA</td>
<td>If the bit flag is set to 1, the product takes the action listed, otherwise, no action is taken.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:1]–Separate data sets are output for each table’s data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:2]–URID header information is considered a table and is written to the load file separately</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:3]–Before and After images are written as 2 records</td>
</tr>
<tr>
<td></td>
<td>Column include flags: If flag is set to 1, the product includes the column in the load file; otherwise, the column is not included.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:4]–Connection type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:5]–Connection ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:6]–Correlation ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:7]–Authorization ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:8]–Plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:1]–URID LRSN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:2]–URID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:3]–URID member ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:4]–URID timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:5]–URID disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:6]–URID commit LRSN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:7]–URID commit point</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2:8]–Database name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:1]–Table space name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:2]–Table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:3]–Table sequence number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:4]–Table creator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:5]–Log RBA/LSRN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:6]–Change type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:7]–Update type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3:8]–Data row length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the bit flag is set to 1, the product takes the action listed, otherwise; no action is taken.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:1]–Product expands hexadecimal data in the load file to character format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:2]–Product writes load file using a delimited format, such as CSV or SDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:3]–Product expands (fills) VARCHAR columns in load file to full length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:4]–Product does not sort load file records</td>
</tr>
<tr>
<td></td>
<td>Column include flags: If flag is set to 1, the product includes the column in the load file; otherwise, the column is not included.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:5]–Timestamp of log record</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:6]–Partition number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4:7]–RID of log record</td>
</tr>
</tbody>
</table>
Table 97  LINEDATA2 field: Load file type (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>If the bit flag is set to 1, the product takes the action listed; otherwise, no action is taken.</td>
</tr>
<tr>
<td>[4:8]</td>
<td></td>
<td>[4:8]–Product allocates separate data sets for each table selected in the log scan, regardless of whether any log records exist for the table (generate empty files)</td>
</tr>
<tr>
<td>INCLUDEURID</td>
<td>INTEGER</td>
<td>1 – If all URID fields are included with each load record 2 – If no URID fields are included with each load record 3 – If only the URID value itself is included</td>
</tr>
<tr>
<td>UPDATEOPTS</td>
<td>INTEGER</td>
<td>1 – If only Before images for updates are included 2 – If only After images for updates are included 3 – If both Before and After images are included</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>INTEGER</td>
<td>Indicator that defines the format of dates within the data set: 0 – USA 1 – EUR 2 – ISO 3 – JIS</td>
</tr>
<tr>
<td>DELIMITER</td>
<td>CHAR(1)</td>
<td>Character used as a delimiter in a delimited format load file</td>
</tr>
<tr>
<td>CNTLFILETYPE</td>
<td>INTEGER</td>
<td>Reserved.</td>
</tr>
<tr>
<td>LOADOUTFMT</td>
<td>INTEGER</td>
<td>Indicator that defines the format of the output load file: 1 – Log Master native format 2 – UNLOAD PLUS format 3 – CSV (Comma Separated Value) format 4 – SDF (Standard Definition Format) format</td>
</tr>
<tr>
<td>LLRECTIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Reserved</td>
</tr>
<tr>
<td>LLPARTITION</td>
<td>SMALLINT</td>
<td>Reserved</td>
</tr>
<tr>
<td>LLRECRID</td>
<td>SMALLINT</td>
<td>Reserved</td>
</tr>
<tr>
<td>ABORTEDTRANS</td>
<td>INTEGER</td>
<td>0 – Include rollback value is No 1 – Include rollback value is Yes 2 – Include rollback value is Only</td>
</tr>
<tr>
<td>BIT FLAGS</td>
<td>CHAR(4) FOR BIT DATA</td>
<td>Single word defined as a series of individual bit flags. Flags are listed in order using the format [byte position:bit position]. If the bit flag is set to 1, the product takes the action listed; otherwise, no action is taken</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1:1]–Product creates load data file using a record format of VBS (variable, blocked, spanned)</td>
</tr>
<tr>
<td>INCLUDELOBS</td>
<td>BOOLEAN</td>
<td>0 – Output load file does not contain LOB data 1 – Output load file contains LOB data</td>
</tr>
<tr>
<td>INCLUDEXML</td>
<td>BOOLEAN</td>
<td>0 – Output load file does not contain XML data 1 – Output load file contains XML data</td>
</tr>
</tbody>
</table>
**LINEDATA2 field: DDL file type**

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies data definition language (DDL) output.

Table 98  **LINEDATA2 field: DDL file type**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| BIT FLAGS       | CHAR(4) FOR BIT DATA | Single word defined as a series of individual bit flags, describing various traits of the generated DDL. Flags are listed in order using the format [byte position:bit position]. If the bit flag is set to 1, the product takes the action listed; otherwise, no action is taken.  

  - [1:1]–Comments are generated and included in generated DDL (VERBOSE output)  
  - [1:2]–The product automatically executes the UNDO DDL after it is generated |
| COMMITCOUNT     | SMALLINT    | Number of URIDs (transactions) between COMMITS as specified through the online interface or in batch syntax                                                                                                 |
| UNDOABORT       | INTEGER     | Code indicating include rollback value for UNDO DDL:  

  - 0 – No  
  - 1 – Yes  
  - 2 – Only |
| MIGRATEABORT    | INTEGER     | Code indicating include rollback value for MIGRATE DDL:  

  - 0 – No  
  - 1 – Yes  
  - 2 – Only |

**LINEDATA2 field: Drop recovery outcopy file type**

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies the output is an output image copy created during an automated drop recovery action that uses the BMC Software product RECOVER PLUS.

Table 99  **LINEDATA2 field: Drop recovery outcopy file type (part 1 of 2)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td>CHAR(25)</td>
<td>Database containing the table space</td>
</tr>
<tr>
<td>TSNAME</td>
<td>CHAR(25)</td>
<td>Name of the table space contained in the image copy</td>
</tr>
</tbody>
</table>
### Table 99  LINEDATA2 field: Drop recovery outcopy file type (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| ICBACKUP    | CHAR(3)   | Indicator that defines the type of image copy parameters contained in the output data set:  
  - blank – LOCALSITE primary (first data set named with OUTCOPYDDN)  
  - LB – LOCALSITE backup (second data set named with OUTCOPYDDN)  
  - RP – RECOVERYSITE primary (first data set named with RECOVERYDDN)  
  - RB – RECOVERYSITE backup (second data set named with RECOVERYDDN) |
| DDNAME      | CHAR(9)   | DDNAME or DDNAME prefix |
| REGCOPY     | SMALLINT  | Indicator that defines if the image copy is registered in SYSCOPY  
  - 0 – No  
  - 1 – Yes |

### LINEDATA2 field: Drop recovery control parameters type

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies that the output is control parameters for an automated drop recovery action that uses the BMC Software product RECOVER PLUS.

### Table 100  LINEDATA2 field: Drop recovery control parameters type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| EXECDDL         | SMALLINT  | Indicator that determines if the generated DDL is executed after RECOVER PLUS performs a recovery:  
  - 0 – DDL not executed  
  - 1 – DDL executed |
| EXECPOSTRECSQL  | SMALLINT  | Indicator that determines if the product generates a job step to execute the generated SQL after RECOVER PLUS performs a recovery:  
  - 0 – No job step generated  
  - 1 – Job step generated |
| EXCRECOVER      | SMALLINT  | Indicator that determines if the product generates a job step to execute RECOVER PLUS:  
  - 0 – No job step generated  
  - 1 – Job step generated |
| EXCREBINDS      | SMALLINT  | Indicator that determines if the product generates a job step to execute rebinds after RECOVER PLUS performs a recovery:  
  - 0 – No job step generated  
  - 1 – Job step generated |
This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies that the output is a LOB VSAM file that the product uses to store LOB column values.

### Table 100  LINEDATA2 field: Drop recovery control parameters type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGONLY</td>
<td>SMALLINT</td>
<td>Indicator that determines if the product includes the RECOVER PLUS keyword LOGONLY in the generated recovery parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not include LOGONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Include LOGONLY</td>
</tr>
<tr>
<td>OUTCOPYBYPART</td>
<td>SMALLINT</td>
<td>Indicator that determines if the product generates parameters for separate image copies for each partition of a partitioned table space or each data set of a table space that is stored in multiple data sets:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Generate one image copy containing all partitions or data sets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Generate separate image copies</td>
</tr>
<tr>
<td>EXECCHECK</td>
<td>SMALLINT</td>
<td>Indicator that determines if the product generates a job step to execute checks after RECOVER PLUS performs a recovery:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – No job step generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Job step generated</td>
</tr>
</tbody>
</table>

### LINEDATA2 field: LOB VSAM file type

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies that the output is a LOB VSAM file that the product uses to store LOB column values.

### Table 101  LINEDATA2 field: LOB VSAM file type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOBCREATOR</td>
<td>CHAR(129)</td>
<td>Creator of auxiliary LOB table</td>
</tr>
<tr>
<td>LOBTABLENAME</td>
<td>CHAR(129)</td>
<td>Name of auxiliary LOB table</td>
</tr>
<tr>
<td>LOBCOLNAME</td>
<td>CHAR(129)</td>
<td>Name of LOB column in base and auxiliary tables</td>
</tr>
<tr>
<td>RELEASEFLAG</td>
<td>n/a</td>
<td>Indicates whether the product requests the operating system to release unused space when the LOB VSAM file is closed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – RELEASE not specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – RELEASE specified</td>
</tr>
<tr>
<td>ALLOCUNITS</td>
<td>n/a</td>
<td>Units that the product uses to allocate the LOB VSAM file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Tracks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Cylinders</td>
</tr>
<tr>
<td>INITIALDISP</td>
<td>n/a</td>
<td>Disposition of the LOB VSAM file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – OLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – NEW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – SHR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – MOD</td>
</tr>
<tr>
<td>COMPRESSFLAG</td>
<td>n/a</td>
<td>Indicates whether the LOB VSAM file is compressed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Compressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Not compressed</td>
</tr>
<tr>
<td>PRIMARYSPACE</td>
<td>INTEGER</td>
<td>Number of units of storage that the product requests when it first allocates the LOB VSAM file</td>
</tr>
</tbody>
</table>
Table 101  LINEDATA2 field: LOB VSAM file type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECONDARYSPACE</td>
<td>INTEGER</td>
<td>Number of units of storage that the product requests to expand the LOB VSAM file as needed.</td>
</tr>
<tr>
<td>VOLCOUNT</td>
<td>SMALLINT</td>
<td>Number of volumes used to store the LOB VSAM file</td>
</tr>
<tr>
<td>RETPD</td>
<td>SMALLINT</td>
<td>Retention period for LOB VSAM file</td>
</tr>
<tr>
<td>EXPDT</td>
<td>CHAR(9)</td>
<td>Expiration date for LOB VSAM file</td>
</tr>
<tr>
<td>DSNMODEL</td>
<td>CHAR(61)</td>
<td>Model data set name for LOB VSAM file</td>
</tr>
<tr>
<td>UNIT</td>
<td>CHAR(9)</td>
<td>Name of unit where LOB VSAM file is stored</td>
</tr>
<tr>
<td>STORCLASS</td>
<td>CHAR(9)</td>
<td>DFSMS storage class of LOB VSAM file</td>
</tr>
<tr>
<td>MGMTCLASS</td>
<td>CHAR(9)</td>
<td>DFSMS management class of LOB VSAM file</td>
</tr>
<tr>
<td>DATACLASS</td>
<td>CHAR(9)</td>
<td>DFSMS data class of LOB VSAM file</td>
</tr>
<tr>
<td>RECFM</td>
<td>CHAR(5)</td>
<td>Record format of LOB VSAM file</td>
</tr>
<tr>
<td>LRECL</td>
<td>INTEGER</td>
<td>Logical record length (in bytes) of LOB VSAM file</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>n/a</td>
<td>Type of data set used to store the LOB VSAM file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – PDS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – PDSE</td>
</tr>
<tr>
<td>DIRBLKS</td>
<td>INTEGER</td>
<td>Number of directory blocks that the product requests when it first allocates the LOB VSAM file</td>
</tr>
</tbody>
</table>

**LINEDATA2 field: XML VSAM file type**

This table shows the layout of the LINEDATA2 field in the Work ID Output File Definitions Table (Table 93 on page 418) when the LINETYPE value specifies that the output is an XML VSAM file that the product uses to store XML column values.

Table 102  LINEDATA2 field: XML VSAM file type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLCREATOR</td>
<td>CHAR(129)</td>
<td>Creator of table with XML column</td>
</tr>
<tr>
<td>XMLTABLENAME</td>
<td>CHAR(129)</td>
<td>Name table with XML column</td>
</tr>
<tr>
<td>XMLCOLNAME</td>
<td>CHAR(129)</td>
<td>Name of XML column</td>
</tr>
<tr>
<td>RELEASEFLAG</td>
<td>n/a</td>
<td>Indicates whether the product requests the operating system to release unused space when the XML VSAM file is closed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – RELEASE not specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – RELEASE specified</td>
</tr>
<tr>
<td>ALLOCUNITS</td>
<td>n/a</td>
<td>Units that the product uses to allocate the XML VSAM file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Tracks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Cylinders</td>
</tr>
<tr>
<td>INITIALDISP</td>
<td>n/a</td>
<td>Disposition of the XML VSAM file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – OLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – NEW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – SHR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – MOD</td>
</tr>
<tr>
<td>COMPRESSFLAG</td>
<td>n/a</td>
<td>Indicates whether the XML VSAM file is compressed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Compressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Not compressed</td>
</tr>
<tr>
<td>PRIMARYSPACE</td>
<td>INTEGER</td>
<td>Number of units of storage that the product requests when it first allocates the XML VSAM file</td>
</tr>
</tbody>
</table>
This table shows the layout of the OUTPUTDATA field in the Work ID Output File Definitions Table (Figure 93 on page 418) when the OUTPUTTYPE value specifies that the product writes output to a data set.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECONDARYSPACE</td>
<td>INTEGER</td>
<td>Number of units of storage that the product requests to expand the XML VSAM file as needed.</td>
</tr>
<tr>
<td>VOLCOUNT</td>
<td>SMALLINT</td>
<td>Number of volumes used to store the XML VSAM file</td>
</tr>
<tr>
<td>RETPD</td>
<td>SMALLINT</td>
<td>Retention period for XML VSAM file</td>
</tr>
<tr>
<td>EXPDT</td>
<td>CHAR(9)</td>
<td>Expiration date for XML VSAM file</td>
</tr>
<tr>
<td>DSNMODEL</td>
<td>CHAR(61)</td>
<td>Model data set name for XML VSAM file</td>
</tr>
<tr>
<td>UNIT</td>
<td>CHAR(9)</td>
<td>Name of unit where XML VSAM file is stored</td>
</tr>
<tr>
<td>STORCLASS</td>
<td>CHAR(9)</td>
<td>DFSMS storage class of XML VSAM file</td>
</tr>
<tr>
<td>DATACLASS</td>
<td>CHAR(9)</td>
<td>DFSMS management class of XML VSAM file</td>
</tr>
<tr>
<td>RECFM</td>
<td>CHAR(5)</td>
<td>Record format of XML VSAM file</td>
</tr>
<tr>
<td>LRECL</td>
<td>INTEGER</td>
<td>Logical record length (in bytes) of XML VSAM file</td>
</tr>
<tr>
<td>DSNTYPE</td>
<td>n/a</td>
<td>Type of data set used to store the XML VSAM file:</td>
</tr>
<tr>
<td>DIRBLKS</td>
<td>INTEGER</td>
<td>Number of directory blocks that the product requests when it first allocates the XML VSAM file</td>
</tr>
</tbody>
</table>

**OUTPUTDATA field: Data set type**

Table 103  OUTPUTDATA field: Data set type (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELEASEFLAG</td>
<td>SMALLINT</td>
<td>Code indicating whether space is released when the data set is closed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Do not release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Release</td>
</tr>
<tr>
<td>ALLOCUNITS</td>
<td>SMALLINT</td>
<td>Code indicating type of allocation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Tracks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Cylinders</td>
</tr>
<tr>
<td>INITIALDISP</td>
<td>SMALLINT</td>
<td>Code indicating initial data set disposition:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Old</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – New</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Share</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Mod</td>
</tr>
<tr>
<td>COMPRESSFLAG</td>
<td>SMALLINT</td>
<td>Code indicating whether output is compressed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Compress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Do not compress</td>
</tr>
<tr>
<td>PRIMARYSPACE</td>
<td>INTEGER</td>
<td>Primary DASD allocation</td>
</tr>
</tbody>
</table>
Table 103 OUTPUTDATA field: Data set type (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECONDARYSPACE</td>
<td>INTEGER</td>
<td>Secondary DASD allocation</td>
</tr>
<tr>
<td>VOLCOUNT</td>
<td>SMALLINT</td>
<td>Number of volumes to allocate</td>
</tr>
<tr>
<td>RETPD</td>
<td>SMALLINT</td>
<td>Retention period in days</td>
</tr>
<tr>
<td>EXPDT</td>
<td>CHAR(9)</td>
<td>Data set expiration date (null terminated)</td>
</tr>
<tr>
<td>DSNMODEL</td>
<td>CHAR(61)</td>
<td>Model for data set name, primary output file (null terminated)</td>
</tr>
<tr>
<td>MEMBERNAME</td>
<td>CHAR(9)</td>
<td>Member name of partitioned data set (null terminated).</td>
</tr>
<tr>
<td>MODELDSCB</td>
<td>CHAR(45)</td>
<td>Generation Data Group (GDG) model (null terminated)</td>
</tr>
<tr>
<td>UNIT</td>
<td>CHAR(9)</td>
<td>Unit name (null terminated)</td>
</tr>
<tr>
<td>STORCLASS</td>
<td>CHAR(9)</td>
<td>DFSMS storage class (null terminated)</td>
</tr>
<tr>
<td>MGMTCLASS</td>
<td>CHAR(9)</td>
<td>DFSMS management class (null terminated)</td>
</tr>
<tr>
<td>DATACLASS</td>
<td>CHAR(9)</td>
<td>DFSMS data class (null terminated)</td>
</tr>
<tr>
<td>LRECL</td>
<td>INTEGER</td>
<td>Logical record length</td>
</tr>
<tr>
<td>VOLSEQ</td>
<td>SMALLINT</td>
<td>Volume sequence number</td>
</tr>
<tr>
<td>USEDDNAMEOUTPUT</td>
<td>CHAR(1)</td>
<td>Y – generated as DDNAME output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N – not generated as DDNAME output</td>
</tr>
<tr>
<td>OUTPUTDDNAME</td>
<td>CHAR(9)</td>
<td>DDNAME to be used as output dd name</td>
</tr>
</tbody>
</table>

OUTPUTDATA field: SYSOUT type

This table shows the layout of the OUTPUTDATA field in the Work ID Output File Definitions Table (Table 93 on page 418) when the OUTPUTTYPE value specifies SYSOUT.

Table 104 OUTPUTDATA field: SYSOUT type (part 1 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENTPAGECOUNT</td>
<td>INTEGER</td>
<td>Number of pages per SYSOUT segment (0-99999)</td>
</tr>
<tr>
<td>COPIES</td>
<td>SMALLINT</td>
<td>Number of copies to print (0-255)</td>
</tr>
<tr>
<td>MODIFYTRC</td>
<td>SMALLINT</td>
<td>Modify character table number (0-3)</td>
</tr>
<tr>
<td>HOLD</td>
<td>SMALLINT</td>
<td>Code indicating hold status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Hold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Do not hold</td>
</tr>
<tr>
<td>SYSOUTCLASS</td>
<td>CHAR(1)</td>
<td>SYSOUT class</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>CHAR(9)</td>
<td>SYSOUT destination or remote destination ID (null terminated)</td>
</tr>
<tr>
<td>DESTUSERID</td>
<td>CHAR(9)</td>
<td>User ID associated with destination (null terminated)</td>
</tr>
<tr>
<td>CHARTABLE [0-3]</td>
<td>CHAR(5)</td>
<td>Four font tables (each entry is null terminated)</td>
</tr>
</tbody>
</table>
Work ID Step Table (ALPWSTP)

This table contains a row for each step defined within each work ID. Unless otherwise noted, all columns have the NOT NULL WITH DEFAULT attribute. For those columns that are overloaded, subsequent tables define the individual fields within the overloaded columns.

Table 105 shows columns that make up the unique index key in bold face type.

### Table 105  Work ID step table (ALPWSTP)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID associated with the work ID. This value can be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ defined by the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ obtained from the user’s TSO ID</td>
</tr>
<tr>
<td>WORKIDNAME</td>
<td>CHAR(18)</td>
<td>Unique name associated with the work ID</td>
</tr>
<tr>
<td>STEPNUM</td>
<td>SMALLINT</td>
<td>Sequential step number within the work ID</td>
</tr>
<tr>
<td>PHASEID</td>
<td>SMALLINT</td>
<td>Phase within the work ID</td>
</tr>
<tr>
<td>STEPTYPE</td>
<td>SMALLINT</td>
<td>Code indicating the type of step represented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Log mark</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Log scan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Execute SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – High-speed Apply JCL generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Repository maintenance</td>
</tr>
<tr>
<td>SPACELIST</td>
<td>VARCHAR(512)</td>
<td>For log mark steps only, a list of comma-delimited table space names</td>
</tr>
<tr>
<td>STEPDATA</td>
<td>LONGVARCHAR</td>
<td>Overloaded field containing different data depending on the STEPTYPE (for the possible formats, see Table 106 on page 433 through Table 110 on page 438)</td>
</tr>
</tbody>
</table>
**STEPDATA field: Log mark step type**

This table shows the layout of the STEPDATA field in the Work ID Step Table (Table 105 on page 432) when the STEPTYPE value specifies a log mark.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUIESCEFLAG</td>
<td>SMALLINT</td>
<td>0 – No quiesce is taken before the log mark</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Quiesce taken before log mark</td>
</tr>
<tr>
<td>MARKNAME</td>
<td>CHAR(41)</td>
<td>Name of the generated log mark (symbolic substitutions OK)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>CHAR(65)</td>
<td>Text description associated with the log mark (this field is a “C” format, null-terminated string)</td>
</tr>
<tr>
<td>TABLESPACESET</td>
<td>SMALLINT</td>
<td>Code indicating if QUIESCE requested for complete table space set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Table spaces only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Table space set (including RI table spaces, base table spaces, and LOB table spaces)</td>
</tr>
</tbody>
</table>

**STEPDATA field: Log scan step type**

This table shows the layout of the STEPDATA field in the Work ID Step Table (Table 105 on page 432) when the STEPTYPE value specifies a log scan.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROMTYPE</td>
<td>SMALLINT</td>
<td>Code indicating the type of start point to use for the log scan:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – RBA/LSRN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Log mark name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Date/time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Relative date/time</td>
</tr>
<tr>
<td>FROMVALUE</td>
<td>CHAR(50)</td>
<td>Value dependent on the FROMTYPE values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – RBA/LSRN, CHAR(10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Log mark name, CHAR(41)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELGENERATION, SHORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Date, CHAR(9) / Time, CHAR(13)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Relative date/time specification</td>
</tr>
<tr>
<td>TOTYPE</td>
<td>SMALLINT</td>
<td>Code indicating the type of end point to use for the log scan:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – RBA/LSRN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Log mark name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Date/time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – Current</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – Relative date/time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – Last archive</td>
</tr>
</tbody>
</table>
### Table 107  **STEPDATA field: Log scan step type (part 2 of 3)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| TOVALUE     | CHAR(50)  | Value dependent on the TOTYPE values:  
|             |           | 0 – RBA/LSRN, CHAR(10)  
|             |           | 1 – Log mark name, CHAR(41)  
|             |           | RELGENERATION, SHORT  
|             |           | 2 – Date, CHAR(9) / Time, CHAR(13)  
|             |           | 3 – Relative date/time specification  
| OPENTRANSFLAG | SMALLINT  | 0 – No open transaction processing is used  
|             |           | 1 – Log scan step requires open transaction processing  
| OPENTRANSL | INTEGER   | Handle ID of this log scan:  
|             |           | 0 – Log scan is not an ongoing process  
|             |           | nonzero – Log scan is an ongoing process and  
|             |           | the product uses this handle ID to search for  
|             |           | previous open URIDs  
| MARKNAME    | CHAR(41)  | Name of the log mark that the product generates at the same time as it scans the log  
| QUIESCEFLAG | SMALLINT  | 0 – No quiesce is taken before the log mark  
|             |           | 1 – Quiesce taken before log mark  
| DESCRIPTION | CHAR(65)  | Description of the log mark  
| REDOTYPE    | SMALLINT  | Code indicating the starting point for REDO information:  
|             |           | 0 – Last common point-in-time  
|             |           | 1 – Specified RBA/LSRN  
|             |           | 2 – Specified log mark  
|             |           | 3 – Last common quiesce point  
| REDORBA     | RESERVED  | RBA/LSRN value (when REDO TYPE is specified as RBA/LSRN)  
|             | CHAR(6)   |  
| REDOMARKID  | CHAR(40)  | REDO log mark name  
| REDORELGEN   | SMALLINT  | Relative generation number for REDO log mark  
| SORTNOHISTORY | SMALLINT  | Code indicating whether the product saves history information for this step of the current run of the current work ID:  
|             |           | 0 – Do not save history information  
|             |           | 1 – Save history information  
| LSSFILESZ   | SMALLINT  | Code indicating the selected method for obtaining or calculating the sort file size estimate:  
|             |           | -1 – None  
|             |           | 0 – Log bytes (total volume of log records read)  
|             |           | 1 – Use history  
|             |           | 2 – Estimate (use estimate value supplied by user)  
| FILESIZE    | INTEGER   | Estimated file size value entered by user  
| AVGRECLEN   | INTEGER   | Average record length entered by user  
| LIMITTYPE   | SMALLINT  | Code indicating the type of alternate limit (if any) specified for log scan:  
|             |           | 0 – No alternate limit specified (use TO point only)  
|             |           | 1 – Limit is number of log files  
|             |           | 2 – Limit is duration (days, hours, or both)  

---

434  [Log Master for DB2 Reference Manual](#)
STEPDATA field: Execute SQL step type

This table shows the layout of the STEPDATA field in the Work ID Step Table (Table 105 on page 432) when the STEPTYPE value specifies an execute SQL step.

Table 108  STEPDATA field: Execute SQL step type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOMALYANALYZE</td>
<td>SMALLINT</td>
<td>Code indicating if the product performs backout integrity checking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not perform backout integrity checking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Perform backout integrity checking</td>
</tr>
<tr>
<td>EXECUTESQL</td>
<td>SMALLINT</td>
<td>Code indicating if the product executes SQL statements in addition to backout integrity checking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not execute the SQL (backout integrity checking only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Execute the SQL</td>
</tr>
<tr>
<td>BINDOWNER</td>
<td>CHAR(8)</td>
<td>Character value to use as the bind owner</td>
</tr>
<tr>
<td>DECIMALPOINT</td>
<td>SMALLINT</td>
<td>Code indicating what character to use as the decimal point:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Use default from DSNHDECP module of DB2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Use decimal point</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Use comma</td>
</tr>
<tr>
<td>PLANNNAME</td>
<td>CHAR(8)</td>
<td>Character value to use as the plan name</td>
</tr>
<tr>
<td>FREEPLAN</td>
<td>SMALLINT</td>
<td>Code indicating if the product frees the plan after completion:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – Do not free the plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Free the plan</td>
</tr>
</tbody>
</table>
**STEPDATA field: High-speed Apply JCL generation step type**

This table shows the layout of the STEPDATA field in the Work ID Step Table (Table 105 on page 432) when the STEPTYPE value specifies an High-speed Apply JCL generation step.

Table 109  **STEPDATA field: High-speed Apply JCL generation step type (part 1 of 2)**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| STARTUP_INPUTTYPE       | CHAR(12)  | String indicating whether the input file contains SQL statements or logical log input:  
                          | SQL – SQL statements                                                        |
|                         |           | LogicalLog – logical log input                                              |
| STARTUP_PLANNAME        | CHAR(10)  | String to use as the name of a plan when the product binds packages for the target tables that are updated by the work ID |
| STARTUP_SSID            | CHAR(6)   | DB2 SSID (or group attachment name for data sharing) where input should be executed |
| STARTUP_FILENAME        | CHAR(45)  | Name of the input file that contains either SQL statements or logical log input |
| DISTTUNE_PARTCLUS       | SMALLINT  | Code indicating how the product distributes work between multiple agents when at least some of the target tables are partitioned tables:  
                          | 0 – product uses a single agent to process all of the partitions in a target table  
                          | 1 – product can use multiple agents to process different partitions in a target table |
| DISTTUNE_RICLUS         | SMALLINT  | Code indicating how the product distributes work between multiple agents when at least some of the target tables are subject to referential integrity (RI) relationships:  
                          | 0 – product uses a single agent to process all of the target tables in an RI relationship  
                          | 1 – product can use multiple agents to process different target tables in an RI relationship |
| CONFILE_FILENAME        | CHAR(45)  | Name of the conflict file (or model for multiple conflict files) where the product defers SQL statements or units of work for subsequent processing, examination, or research |
| CONFILE_SINGLEFILE      | SMALLINT  | Code indicating whether the product creates a single conflict file for all agents or one conflict file for each agent:  
                          | 0 – one conflict file for all agents                                         |
|                         |           | 1 – one conflict file for each agent                                       |
Table 109  STEPDATA field: High-speed Apply JCL generation step type (part 2 of 2)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| CONFLICT_RETRYFAIL           | CHAR(12)      | String indicating the conflict resolution action that the product takes when it encounters an SQL code for which an action of Retry is specified and all of the defined retry attempts fail:  
  Terminate – the product commits the current unit of work and terminates processing  
  Abort – the product rolls back the current unit of work and terminates processing |
| CONFLICT_RETRYLIMIT          | CHAR(7)       | String representing the type of units used for CONFLICT_RETRYVALUE:  
  Time – product retries until it reaches the number of seconds specified by RetryValue  
  Count – product retries until it reaches the number of attempts specified by RetryValue |
| CONFLICT_RETRYVALUE          | INTEGER       | Value indicating how often or how long the product retries executing an SQL statement in response to an SQL code for which an action of Retry is specified |
| BINDTUNE_STATEMENTCOUNT      | INTEGER       | Value indicating how many distinct statement types the product can add to a package or bind file before the product initiates a bind on the package        |
| BINDTUNE_MAXPACKAGES         | INTEGER       | Value indicating the maximum number of additional packages that the product can add to the base plan or package during execution                          |
| BINDTUNE_SYNC                | CHAR(5)       | Character string indicating whether the product can perform other processing at the same time as it binds plans or packages:  
  Yes – product must bind packages synchronously  
  No – product can perform other processing at the same time as it binds packages |
| COMMITTRIG_STATECOUNT        | INTEGER       | Value indicating how many SQL statements each agent processes before it issues a COMMIT statement                                                   |
| AGENT_MAXAGENTS              | INTEGER       | Value indicating the maximum number of agents that the product can use during one apply request                                                   |
| CONFLICT_RULES_LEN           | SMALLINT      | Value indicating the actual length of the conflict resolution rules contained in the CONFLICT_AND_OBJMAP_RULES field                                      |
| OBJMAP_RULES_LEN             | SMALLINT      | Value indicating the actual length of the object mapping specifications contained in the CONFLICT_AND_OBJMAP_RULES field                               |
| CONFLICT_AND_OBJMAP_RULES    | VARCHAR(3000) | Overloaded field containing different data depending on whether the work ID defines conflict resolution rules, object mapping specifications, or both types of data |
STEPDATA field: Repository maintenance step type

This table shows the layout of the STEPDATA field in the Work ID Step Table (Table 105 on page 432) when the STEPTYPE value specifies a Repository maintenance step.

Table 110  STEPDATA field: Repository maintenance step type

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETETYPE</td>
<td>SMALLINT</td>
<td>Code indicating which type of Repository data (rows in which table) the product deletes: 0 – Delete overtime rows (ALPOLDO table) 1 – Delete ongoing rows (ALPURID table) 2 – Both overtime and ongoing rows (both tables)</td>
</tr>
<tr>
<td>DELETEENDPOINTTYPE</td>
<td>SMALLINT</td>
<td>Code indicating technique used to delete Repository data: 0 – Age 1 – Date 2 – Before last archive 3 – All 4 – RBA/LRSN</td>
</tr>
<tr>
<td>WORKIDPATTERN</td>
<td>CHAR(27)</td>
<td>Character string that product compares against work ID value in ongoing table rows (ALPURID) to decide if row is deleted.</td>
</tr>
<tr>
<td>DELETEENDPOINTVALUE</td>
<td>CHAR(10)</td>
<td>Time value that product uses to decide if row is deleted (supplied by user). Value varies, depending on DELETEENDPOINTTYPE field: 0 – Number of days 1 – Date value 2 – Not valued 3 – Not valued 4 – RBA/LRSN value</td>
</tr>
</tbody>
</table>
Index

Symbols

& (symbolic substitutions)
  &DATE 325
  &GATN 325
  &SSID 325
  &SYSUID 323, 325
  &TABNAME 241, 247, 323, 324
  &TABOWN 241, 247, 323
  &TIME 325
  &WORKID 324
  &ZPREFIX 325
complete list of 323
in data set names 324
in output definitions 260
| | (concatenation operator in SQL) 212

A

aborted transactions
  including in generated SQL 221
  including in output load files 242
  including in output logical log files 202
  including in reports 192
above-the-bar storage 26
absolute generation, with log mark names 274
active log files
  define range to avoid reading 275
  order of reading 62
ACTIVITY DESC keyword, description 194
ACTIVITY TYPE keyword, description 194
adjusting for Daylight Saving Time 81
AFTER keyword
  description 291
  diagram 285
AGE keyword
  description 172
  diagram 164
ALIAS keyword
  description 290
  diagram 285
aliases
  in filters 290
  in logical log file 203, 351
ALL ACTIVITY keyword
  description 190
  diagram 182
ALL keyword
  description 172
  diagram 164
Allocation parameters, syntax diagram 262
ALPSOPTS load module 34
ALPADRF Repository table 387
ALPCOLS Repository table 388
ALPPDDLF Repository table 388
ALPDRLN Repository table 389
ALPDUMP data set 101
ALPFLIN
  LINEDATA field, catalog activity type 393
  Repository table 391
ALPFLTH Repository table 390
ALPFLTP Repository table 394
ALPFLIL
  LINEDATA field, input DB2 log type 415
  LINEDATA field, input image copy (uncataloged) type 416
  LINEDATA field, input logical log type 415
  Repository table 414
ALPXISQ Repository table 408
ALPLODF Repository table 394
ALPLOGF Repository table 395
ALPLRDMP data set 101
ALPMAIN load module 97
ALPMARK Repository table 397
ALPOFDSN installation option 75
ALPOFIL Repository table 417
ALPOFIL, LINEDATA2 field
  DDL file type 426
  drop recovery control parameters type 427
  drop recovery outcopy file type 426
  load file type 423
  LOB VSAM file type 428
  logical log file type 419
  report file type 421
  SQL file type 420
  XML VSAM file type 429
ALPOFIL, OUTPUTDATA field
  Data Set type 430
  SYSOUT type 431
ALPOLDO Repository table 398
ALPPRINT data set 100
ALPRPTF Repository table 400
ALPRTMPL Repository table 403
ALPSECH Repository table 404
ALPSQLC Repository table 405
ALPSQLF Repository table 406
ALPSQLX Repository table 408
ALPURID Repository table 399
ALPWHDR Repository table 409
ALPWHSIS Repository table 413
ALPWLSH Repository table 416
ALPWSTP Repository table 432
ALPWSTP, STEPDATA field
execute SQL step type 435
High-speed Apply JCL generation step type 436
log mark step type 433
log scan step type 433
repository maintenance step type 438
ALTER keyword
description 293
diagram 292
ALTERBUFFERPOOL keyword, description 300
alternate indexes
using overrides 209
with PRIMARY KEY 209
alternate installation options module 99
alternate limit on range of log scan 270
AMSGLEV installation option 46
analyze phase of processing, performance of 123
ANPCT installation option 54
APF authorizations 30
APFONLIN installation option 76
archive log files
deleting old objects based on 172
limit range of log scan to 270, 275
order of reading 62
ASCODED keyword
description 314
diagram 308
ATTEMPT COMPLETION keyword
description 122
diagram 119
Audit report, syntax 185
AUTH ID keyword
description 195, 287
diagram 285
authorization mechanisms, description 27
authorizations
APF 30
DB2 28
mechanisms, description 27
RACF 29
RACF ID and OPNDB2ID option 44
required 26
showing changes to 258
SQL execution 28
Automated Drop Recovery File Table (ALPADRF) in
Repository 387
automated drop recovery. See drop recovery
AVGRECLEN keyword
description 156
diagram 154
B
backing out transactions with UNDO SQL 215
backing up the Repository 385
Backout Integrity reports
key store memory with 144
syntax (detail) 183
syntax (summary) 183
BCPCT installation option 54
BEFORE keyword
description 291
diagram 285
BEFORE OLDEST ARCHIVE keyword
description 172
diagram 164
bind, dynamic. See dynamic bind of plans and packages
BINDADD authority 28
BINDOWN installation option 43
BINDOWNER keyword
description 306, 312
diagram 306, 308
BINDQUALIFIER installation option 43
BMC Software
related product, COPY PLUS for DB2 46, 125, 126, 385
related product, RECOVER PLUS 74, 118, 308, 317,
320, 427
related product, UNLOAD PLUS for DB2 69, 234
BMC Software, contacting 2
BMCSORT installation option 85
BMCSORT. See sorting
bootstrap data set (BSDS)
checkpoint records for Open Transaction report 188
input source 161
online interface and 30
BSDS keyword
description 161
diagram 161
buffering tool, VSAM 30
building Log Master jobs 94
BYPART keyword
description 314
diagram 308
BYPASSLL installation option 68
C
CA-ACF2 security product 27
cabinet copies, row completion and 125
calculating dictionary space 55
catalog activity
report 186
verbose DDL to show 258
Catalog activity definition, syntax diagram 292
Catalog object definition, syntax diagram 295
catalogObject variable
description 295
diagram 295
CA-Top Secret security product 27
CHANGED keyword
description 283
diagram 279
characters, wildcard 282
CHARS keyword
description 199
diagram 197
CHECK DATASET keyword
description 315
diagram 308
checkpoint records in BSDS, for Open Transaction report 188
CLASS keyword
description 198
diagram 197
cleaning up after a Log Master job 112
CLIST for product execution 33
CLUS record of logical log 361
CLUSTER keyword
description 142
diagram 139
Clustering Index Column Description Record (DCCL) of logical log 365
Clustering Index Information Record (DCXI) of logical log 364
CMPLGRNG, installation option 59
cntlDataSetName variable
description 161
diagram 161
cold start, unterminated URIDS from 132
Column include/exclude definition, syntax diagram 267
Column List Table (ALPCOLS) in Repository 388
COLUMN NAME keyword
description 246
diagram 232
columns
LOB columns and UNDO SQL 227
LOB columns, logging of 227
COLUMNS keyword
description 268
diagram 267
columns, order within logical log records 370
Comma Separated Value (CSV) format
in load files 234
in Summary reports 191
COMMAND keyword, description 195, 290
Commands, syntax for report 187
comment delimiter 108
COMMENT ON keyword
description 293
diagram 292
comments
in generated output 210, 230, 258
in Log Master SYSIN syntax 108
comments in SYSIN syntax 108
COMMIT FREQUENCY keyword
description 210, 258
diagram 208, 255
Commit report, syntax 186
compensated log records
in Rollback report 189
including in generated SQL 221
including in output load files 242
including in output logical log files 202
including in reports 192
completed transactions in output 182
completion processing. See row completion processing
components of logical log 332
COMPRESS keyword
description 266
diagram 262
compression dictionaries
access order for sources of 159, 168
access order for types of image copies 125
calculating memory for 55
DICTIONARYSPACE keyword 128
DICTSPC installation option 55
from encrypted image copies 125
image copies and 125
in Repository 168
INCLUDE DICTIONARY keyword 168
memory for 55, 128
performance reading 55
CONCAT OPERATOR keyword
description 212
diagram 208
concurrent image copies, row completion and 56
CONNECTION ID keyword
description 195, 287
diagram 285
CONNECTION TYPE keyword
description 195, 288
diagram 285
considerations, special. See special considerations for SQL output files
constants, string in generated SQL 212
control file format, logical log 332
Control File ID Record (DCNT) of logical log 335
control file, load 234
control library. See sample library
conventions
comments in SYSIN syntax 108
documentation 16
sort work data set names 104
syntax diagrams 114
COPIES keyword
description 198
COPY PLUS for DB2 (BMC Software product) 46, 125, 126, 385
CORRELATION ID keyword
description 195, 287
diagram 285
CREATE keyword
description 294
diagram 292
CREATE RBA keyword, description 195
CSTFILE installation option 57
CSTMOUNT installation option 58
CSTSEG installation option 58
CSV (Comma Separated Value) format
in load files 234
in Summary reports 191
CURRENT keyword
description 121, 275
diagram 119, 272
Current Path Record (XPTH) of logical log 343
current time, creating log mark at 305
customer support 3
customizing a report with a template 183, 322
CYLINDERS keyword
description 141, 145, 148, 264
diagram 139, 144, 147, 262

D

DASD DATASETS keyword
description 131
diagram 120
DASDDNSNS installation option 81
Data Capture Changes
settings for DB2 catalog tables 65
syntax for report 187
Data Change Header Description Record (XHDF) of logical log 345
Data Change Record (LLDF) of logical log 370
data file
external files for LOB values 246
external files for XML values 246
load 233
logical log 369
DATA keyword
description 142
diagram 139
data sets
ALPDUMP 101
ALPLRDMP 101
ALPPRINT 100
dynamically allocated by product 100
JCL in separate data set 107
key (for encrypted image copies) 46
migrating to archive 80, 130
output handling at termination 110
recalling from archive 78, 129
required by product 100
SQLCODES 101, 328
SQLPRINT 101, 307
SQLXLAT 101, 328
SYSIN 100
SYSMDUMP 101
SYSOUT 100
SYSUDUMP 101
data sharing
common end point for 60, 126
DDPRMBR keyword 81
filtering on log records from members 289
group attachment name 98
input DB2 log files 164
LOGSORT installation option 63
LOGSORT keyword 138
LOGTAPES installation option 64
LOGTAPES keyword 139
MINLOGPT keyword 60, 126
MTPRMBR keyword 79
number of early recall tasks and 79
OSRCHNCT installation option 77, 78
PER MEMBER keyword 129, 131
reading member log files 127
scan ranges for members 61
scanning members during row completion 59
scope of data set migration requests 81
scope of early recall requests 79
searching for log files 77, 78
SMSTASKS keyword 79
sorting log records 63, 138, 139, 196
specify input DB2 log files for 164
tape units for log files 64
USELGRNG keyword 61, 127
Data Sharing Member Record (DDSM) of logical log 336
DATABASE keyword
description 321
diagram 320
DATABASE NAME keyword
description 195, 288, 310
diagram 285, 308
databaseName variable
description 158
diagram 157
DATACLAS keyword
description 141, 146, 149, 266
diagram 139, 144, 147, 262
DATASET keyword
description 159
diagram 159
dataSetName variable
description 163, 260
diagram 162, 260
DATE keyword
description 172, 276
diagram 164, 272
&DATE symbolic substitution 325
date/time values
  Daylight Saving Time adjustment 81
  format in logical log 202
  format of 121
  TZRULE installation option 81
DATEFMT keyword
  description 121, 202
  diagram 119, 200
Daylight Saving Time, adjustment 81
DAYS keyword
  description 270
  diagram 269
DB2
  active log data sets 62
  authorizations needed 26
  BIND command 42
  BDS 161
  CHECK commands in drop recovery 315
  compression dictionaries 55, 128, 168
  DCC settings for catalog 65
  DSNHDECP module 306
  internal list of indexes 209
  internal time format 202
  load library for product use 94
  object names. See names of DB2 objects
  QUIESCE command 173, 305
  rebind commands in drop recovery 317
  security. See security
  specific log files as input to product 162
  subsystem ID (SSID) in JCL 98
  SYSIBM.SYSCOPY table. See SYSIBM.SYSCOPY table
    in DB2 Catalog
  SYSIBM.SYSINDEXES table 209
  SYSIBM.SYSLOGRNX table 59, 61, 127
  SYSIBM.SYSSYNONYMS table 203
  SYSIBM.SYSSXMLSTRINGS table 201
  table level security 44
  use of RACF ID 44
  UTILPLNS installation option 44
DB2 authorizations
  BINDADD authority 28
  DBADM authority 28
  DBCTRL authority 28
  DBMAINT authority 28
  DISPLAY privilege 28
  EXECUTE privilege 28
  IMAGECOPY privilege 28
  RACF ID 29
  showing changes to 258
  SYSADM authority 28
  SYSCtrl authority 28
DB2 catalog
  auditing 186, 258
  CATALOG ACTIVITY keyword 292
  CATALOG OBJECT keyword 295
  DB2CAT installation option 65
  DB2CATALOG keyword 174
defining DCC settings for 65
defining objects no longer in 326
DROPRECOVERY statement 308
EXECMODE installation option 66
execution mode 121
FLTRMTHD installation option 66
IMAGECOPY statement 157
including catalog information in output 174
missing image copies 157
old objects data set 122, 167, 326
OLD OBJECTS keyword 122
performance information, obtaining 123
PRIME FROM DB2 CATALOG keyword 168
processing 65, 66
quiet point (quiesce) 184, 272
recovering objects dropped from 308
REGISTER keyword 318
registering output image copies 318
reporting on catalog activity 186
REPOS DELETE keyword 171
REPOS UPDATE keyword 167
Repository updates from 167
row completion processing of 65
security changes in 186, 258
selecting catalog activity 292
selecting catalog objects 295
verbose DDL for catalog activity reporting 258
DB2 Column Description Record Part Two (DCL2) of logical log 354
DB2 command definition, syntax diagram 299
DB2 commands
  COMMANDS keyword 299
  including in logical log files 206
  selecting 299
DB2 log input definition, syntax diagram 162
DB2AUTH installation option 44
DB2CAT installation option 65
DB2CATALOG keyword
  description 174
  diagram 166
DB2Command variable, description 299
DB2LOG keyword
  description 162
  diagram 161
DBADM authority 28
DBCS installation option 77
DBCTRL authority 28
DBID keyword
  description 195, 288
  diagram 285
DBMAINT authority 28
DCCL record of logical log 365
DCL2 record of logical log 354
DCLI record of logical log 352
DCNT record of logical log 335
DCXI record of logical log 364
DD name, LOGSCAN output definition 260
DD statements
   ALPDUMP 101
   ALPLRDMP 101
   ALPPRINT 100
      in JCL 100
   sort work data sets 103
   SQLCODES 101, 328
   SQLPRINT 101, 307
   SQLXLAT 101
   STEPLIB 100
   SYSIN 100
   SYSDUMP 101
   SYSOUT 100
   SYSDUMP 101

DDL
   Catalog Activity report 186
   COMMIT frequency 210, 258
      creating MIGRATE DDL 256
      creating UNDO DDL 256
   DDL keyword 167
   DDLOBJ installation option 73
   EXECUTE keyword 256, 312
      generated in drop recovery 312
      immediate execution for Drop Recovery 312
      immediate execution of generated 256
      including DDL in generated SQL 218
      including DDL in logical log files 204
      inserting comments 258
      order of statements 257
   RECFM keyword 257
   RECREATE DATASET keyword 312
      restrictions on REDO DDL 218
      restrictions on UNDO DDL 256
      reversing order of UNDO DDL 257
      selecting based on type of 292
      sort work data sets for 105
      SQLID for execution of 259, 312
      syntax for creation 255
      types of DDL statement 292
      verbose DDL for catalog activity reporting 258
      VERBOISE keyword 258
   DDL file definition, syntax diagram 255
   DDL File Pointer Table (ALPDDLF) in Repository 388
   DDL keyword
      description 167
      diagram 164
   DDLOBJ installation option 73
   DDNAME keyword
      description 192
      diagram 181
      DDNAME output 260
   DDPRMBR installation option 81
   DDSM record of logical log 336
   DECIMAL POINT keyword
      description 211, 306
      diagram 208, 306
   default options (DOPTs). See installation options
default values
      INCLUDE ROLLBACK with reports 193
      installation options (list of) 35
      LRECL for output files 263
      online interface and POF 75
   DELETE keyword
      description 284
      diagram 279
   dependent RBA/LRSN value 379
   DESC keyword
      description 157, 173, 305
      diagram 156, 166, 305
   DEST keyword
      description 198
      diagram 197
   Detail report, syntax 187
   DEV keyword
      description 160, 164
      diagram 159, 162
   DFSMS
      concurrent image copies, row completion and 56, 125
      installation options for use with 52
      diagrams, syntax. See syntax diagrams
   DICL record of logical log 364
   dictionaries. See compression dictionaries
   DICTIONARYSPACE keyword
      calculating value for 55
      description 128
      diagram 119
   DICTSPC installation option 55
   DIRBLKS keyword
      description 265
      diagram 262
   DISPLAY system privilege 28
   displaying status of a Log Master job 109
   DIXI record of logical log 362
   DLCA record of logical log 355
   DLDS record of logical log 338
   DLGE record of logical log 341
   DLGI record of logical log 340
   DLIN record of logical log 369
   DLOB record of logical log 357
   DLUS record of logical log 339
   documentation
      information 16
related 15
syntax diagram conventions 114
DOPTs (default options). See installation options
DOTR record of logical log 342
double-byte character set
  DBCS installation option 77
  DCSSID in SQLCODES data set 327
entering constants 280
online interface and reports and 77
DROP keyword
description 294
diagram 292
drop recovery
  CHECK DATASET keyword 315
defining range of 272
  DROPRECOVERY statement 118, 308
  DSN1COPY DATASET keyword 315
generated DB2 commands 315, 317
generated DDL statements 312
generated RUNSTATS commands 316
generated SQL statements 316
information in Repository 387, 389, 427
  MIGRATE DATASET keyword 316
output image copies 317
overview 309
plans, rebinding copies 317
Range definition 272
  REBIND DATASET keyword 317
  REBUILD installation option 74
rebuilding indexes 74
  RECOVER DATASET keyword 313
  RECOVERY PLUS, parameters for 317
  RECOVER PLUS, without using 315
recovery of a database 310
recovering a dropped table 311, 320
recovering a table space 310
  RECREATE DATASET keyword 316
removing check pending status 315
report 317
  RUNSTATS DATASET keyword 316
syntax for DSN1COPY 315
  syntax for RECOVER PLUS 313
temporary objects 320
types of output 309
versioned tables and 309
Dropped Objects Table (ALPDRLN) in Repository 389
DROPRECOVERY statement
  ASCODED keyword 314
  BINDOWNER keyword 312
  BYPART keyword 314
  CHECK DATASET keyword 315
  DATABASE keyword 321
  DATABASE NAME keyword 310
  DSN1COPY DATASET keyword 315
  INDEP keyword 321
  LOGONLY keyword 313
  MIGRATE DATASET keyword 316
  OUTCOPY keyword 314
OUTCOPYDDN keyword 318
OWNER keyword 322
  REBIND DATASET keyword 317
  RECOVER DATASET keyword 313
  RECOVERYDDN keyword 319
  RECREATE DATASET keyword 312
  REGISTER keyword 318
  REPORT keyword 317
  RUNSTATS DATASET keyword 316
  SQLID keyword 312
  STOGROUP keyword 321
  syntax diagram 308
  TABLE NAME keyword 311
  TABLESPACE keyword 321
  TABLESPACE NAME keyword 310
DSN1COPY
  instead of RECOVER PLUS 315
  using for drop recovery 315
DSN1COPY DATASET keyword
description 315
diagram 308
DSNHDECP module of DB2 306
DSNSYMPRE installation option 51
DSNTYPE keyword
description 265
diagram 262
DSNUM keyword
description 159, 196
diagram 159
DSS1 record of logical log 343
DSS2 record of logical log 344
DTBB record of logical log 349
DTBC record of logical log 350
DTBI record of logical log 347
DTOB record of logical log 351
DTSI record of logical log 346
DUPLICATE DATASET keyword
description 146, 149
diagram 144, 147
duplicate rows in LOAD input, results with UNDO and REDO SQL 226
DURATION keyword
description 184
diagram 182
duration of ranges in Quiet Point report
  excluding shorter ranges 184
  using for quiesce point 185

Index 445
DWDK record of logical log 337
DWRK record of logical log 336
DXML record of logical log 359
DXSF record of logical log 361
DYNALLOC option, RETRY parameter 87
DYNALLOC installation option 85
dynamic bind of plans and packages 33

EARLY RECALL keyword
description 129
diagram 120
electronic documentation 15
encrypted image copies
COPY PLUS for DB2 46
row completion and 46, 125
end point
common for data sharing 60, 126
log scan 269
environment, software 26
ERLYRCL installation option 78
EST keyword
description 155
diagram 154
estimating costs for row completion processing 126
example of running product 95
EXCHANGE keyword
description 284
diagram 279
EXCLUDE INSERT COLUMN NAMES keyword
description 212
EXCLUDE keyword
description 268
diagram 267
EXCLUDE NULL INDICATOR keyword
description 235
diagram 231
excluding columns from output 267
EXEC statement 97
EXECMODE installation option 66
EXECSQL statement
BINDOWNER keyword 306
DECIMAL POINT keyword 306
FREE keyword 307
NOFREE keyword 307
PLAN NAME keyword 307
SQLDataSetName variable 306
SQLPRINT keyword 307
syntax diagram 306
executing
Log Master jobs 97, 109
online interface 33
SQL. See SQL execution
EXECUTION MODE keyword
description 121
diagram 119
EXPAND VAR keyword
description 202, 242
diagram 200, 231
EXPDT keyword
description 265
diagram 262
eXtensible Markup Language (XML). See XML
external DB2 security. See security, external
external files for LOB values, with load output 245, 265
external files for XML values, with load output 245, 265

FCB keyword
description 199
diagram 197
FCPCT installation option 54
FCUSE installation option 60
FILECOST keyword
description 126
diagram 119
FILSZ keyword
description 154
diagram 154
Filter Anchor Record (DFIL) of logical log 366
Filter Description Record (DFDS) of logical log 367
Filter Header Table (ALPFLTH) in Repository 390
Filter ID Record (DFID) of logical log 366
FILTER keyword
description 175
diagram 166
Filter Line Data Record (DLIN) of logical log 369
Filter Line Record (DFLN) of logical log 368
Filter Line Table (ALPFLIN) in Repository 391
FILTER METHOD keyword
description 123
diagram 119
FILTERREL keyword
description 122
diagram 119
filters
connecting filters with operators 122
DB2 catalog processing 123
differences from column include/exclude 268
FILTER keyword 175
FILTER METHOD keyword 123
FILTERREL keyword 122
FLTRMTHD installation option 66
multiple filter predicates 277
names 175
on records from data sharing members 289
performance of processing 123
predicate 279
predicate, synonyms for 298
processing options for 66, 123
RBA/LRSN values in 287
requirements for QUIESCE 174, 272
requirements for REDO 215
requirements for referential integrity 218
requirements for USELGRNG 127
synonyms in 298
syntax for creation 279
syntax for retrieval 175
Unicode characters in 280
USELGRNG keyword and 127
with REDO information 271
FlashCopy image copies 85, 134
FLTRMTHD installation option 66
format
CSV (Comma Separated Value) 191, 234
load file, records in 249
load file, syntax for 234
logical log control file 332
logical log data file 369
reports, alternate formats for Summary report 191
SDF (Standard Definition Format) 191, 234
segmented logical log records 376
syntax diagrams 17, 114
FORMAT keyword
description 191, 234
diagram 182, 231
FORMS keyword
description 199
diagram 197
FREE keyword
description 307
diagram 306
FROM keyword
description 269
diagram 269

G
&GATN symbolic substitution 325
GENERATE ALWAYS, generating SQL for ROWID or
IDENTITY columns 228
GENERATE EMPTY FILES keyword
description 247
diagram 232
GENERATE MASSDELETE keyword
description 131
diagram 120
GENMDEL installation option 68
GRANT keyword
description 294
diagram 292

H
Help, online 15
heuristic completion processing 60, 132
HEURISTIC FORWARDCOMPLETION keyword
description 132
diagram 120
high-level product syntax diagram 115
high-level qualifier (HLQ) 33
High-speed Apply Engine
for executing SQL statements 306
performance with SQL template 217
history changes
IGNOREFIELDS keyword 244
INCLUDE HISTORY keyword 223
including in generated SQL 223
including in output load files 244
HISTORY keyword
description 153
diagram 150
HLQ (high-level qualifier) 33
HOLD keyword
description 198
diagram 197

I
ICTYPE keyword
description 160
diagram 159
IGNOREFIELDS keyword
description 244, 245
image copies
access order for different types of 125
cabinet copies 125
DFSMS concurrent image copies 56, 125
encrypted image copies 46, 125
explicitly specifying 157
Instant Snapshot copies 125, 126
key data set for encrypted 46
overtime mode and 121
row completion and 56, 121, 157
syntax diagram of parameters for 159
Image Copy report, syntax 188
IMAGECOPY statement
databaseName variable 158
DATASET keyword 159
DEVT keyword 160
dsnm keyword 159
ICTYPE keyword 160
IMRBA keyword 159
REMAP keyword 161
SEQNO keyword 160
syntax diagram 157
TABLESPACE NAME keyword 158
tableSpaceName variable 158
VOLSER keyword 160
IMAGECOPY system privilege 28
IMAGESOURCE keyword
description 124
diagram 119
IMAGESRC installation option 56
improving Log Master performance 90
IMRBA keyword
description 159
diagram 159
IN keyword
description 283, 284, 294, 298, 304
diagram 279, 292, 295
INCLUDE COMMAND keyword
description 206
diagram 200
INCLUDE DDL keyword
description 204, 218
diagram 200, 214
INCLUDE DICTIONARY keyword
description 168
diagram 164
INCLUDE HISTORY keyword
description 223
diagram 214
INCLUDE INSERT COLUMN NAMES keyword
description 212
INCLUDE keyword
description 268
diagram 267
INCLUDE LOBS keyword
description 205, 222, 243
diagram 200, 214, 232
INCLUDE RI keyword
description 218
diagram 214
INCLUDE ROLLBACK keyword
default values with reports 193
description 192, 202, 221, 242
diagram 182, 200, 214, 232, 255
INCLUDE SYNONYMS keyword
description 203
diagram 200
INCLUDE TOTAL keyword
description 191
diagram 182
INCLUDE TRIGGER keyword
description 220
diagram 214
INCLUDE XML keyword
description 206, 223, 244
diagram 200, 214, 232
including columns in output 267
INDEP keyword
description 321
diagram 320
Index Column Description Record (DICL) of logical log 364
Index Information Record (DIXI) of logical log 362
indexes, used for PRIMARY KEY selection 209
indicator, null, in load file output 235
INFO COLUMNS keyword
description 236
diagram 231
informational columns in load file
object names in 237, 238
overview 236
inline LOB 227
input DB2 log files
explicitly specifying 162
heuristic forward completion 132
normal (default) processing 161
Repository and 170
resource demands 163
sorting 162
specifying for data sharing 164
input logical log files
explicitly specifying 161
ongoing processes and 378
segmented records in 376
INPUT statement
BSDS keyword 161
cntlDataSetNname variable 161
dataSetName variable 163
DB2LOG keyword 162
DEVT keyword 164
LLOG keyword 161
memberName variable 164
syntax diagram 161
INSERT keyword
description 284
diagram 279
installation
ALPSOPTS load module 34
APF authorizations 30
BINDOWN installation option 43
BINDQUALIFIER installation option 43
CLIST for execution 33
considerations for Log Master 31
DB2 authorizations needed 28
default options (DOPTs). See installation options
Installation System 34
macro for installation options 34
memory and 46
online interface considerations 30
operating environment 26
options (DOPTs). See installation options
plan names used 43
plans used to install 42
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

product options file. See POF
PUBLICPLAN installation option 43
RACF authorizations 29
related information provided during 87
SYS1.PARMLIB data set 30
upgrading Repository during 32
using an alternate installation options module 99

installation options
alphabetic list 35
alternate module 99
control of sort work data sets 102
controlling sort memory usage 99
DASD DDNS 81
default values of 35
descriptions of 42–85
DFSMS, using different options with 52
list of 35
LOCCPSEL 85
macro for 34
module name 99
module, in JCL 99
overriding 119, 137, 139, 150
product options file. See POF
quick reference list of 35
REMCPSEL 85
TZRULE 81
ZIP 84

Installation System, BMC Software 34
Instant Snapshot image copies (COPY PLUS for DB2) 125,
126
interface, online. See online interface
internal DB2 security. See security, native
ISPF profile and POF, relationship 75

J

JCL
alternate installation options module in 99
building jobs 94
comments in SYSIN input 108
DB2 subsystem ID (SSID) 98
DD statements in 100
element of running product 95
EXEC statement in 97
for Repository clean up 385
generated for drop recovery 309
group attachment name 98
in line syntax 100
JOB statement in 96
MAINT parameter 98
message level (msglevel) 99
online interface, created by 94
required elements to run 94
separate data set for 107
sort work data sets in 102
utility ID (utilID) 99
JOB statement, in Log Master JCL 96
JOBLIB statement 100

K

key data set (for encrypted image copies) 46
key store
ALPLRDMP data set 101
ANPCT installation option 54
BCPCT installation option 54
FCPCT installation option 54
distributing memory among 142
FCUSE installation option 60
KSMEMORY installation option 49
LRPCT installation option 53
memory for 139
MEMORY keyword 140
MEMPERCENT keyword 142
related memory buffer 55, 133
STOREOPTS statement 139
URPCT installation option 53
with Backout Integrity report 144
with logical log input 143
KEYDSNAM installation option 46
keywords, syntax
See syntax keywords
KSALLOCU installation option 50
KSCLOUT installation option 50
KSDACLS installation option 52
KSDATA installation option 51
KSMEMORY installation option 49
KSMGMT installation option 52
KSSPACE installation option 50
KSSTOR installation option 51
KSVOLS installation option 53

L

LABEL ON keyword
description 294
diagram 292
large objects. See LOBs
Large volume column definition
description 246
diagram 232
LAST ARCHIVE keyword
description 275
diagram 272
LASTRUN keyword
description 176
diagram 166
restrictions on 177

layout of records
load file 249
logical log data file 370
level of messages 99
LIKE keyword
description 264, 282

Index  449
diagram 262, 279
limit, alternate on range of log scan 270
LINEDATA field
   ALPFLIN, catalog activity type 393
   ALPIFIL, input DB2 log type 415
   ALPIFIL, input image copy (uncataloged) type 415,
       416
   ALPIFIL, input logical log type 415
LINEDATA2 field of ALPOFIL
   DDL file type 426
   drop recovery control parameters type 427
   load file type 423
   LOB VSAM file type 428
   logical log file type 419
   OUTCOPY file type 426
   report file type 421
   SQL file type 420
   XML VSAM file type 429
listing of sample JCL 95
LLDF record of logical log 370
LLOG keyword
   description 161, 166
   diagram 161, 164
LLOGAUTH installation option 45
load actions
   BYPASSLL installation option 68
   including in UNDO and REDO SQL 225
   INSERTs resulting from 225
   limits on SQL generated from 225
   LOAD REPLACE LOG NO 228
   LOAD REPLACE LOG YES 226
   LOAD RESUME LOG YES, logging of 225
load file
   external files for LOB values 245, 265
   external files for XML values 245, 265
   format 69, 234
   IGNOREFIELDS keyword 244
   INCLUDE LOBS keyword 243
   INCLUDE ROLLBACK keyword 242
   INCLUDE XML keyword 244
   including aborted transactions 242
   including compensated log records 242
   including history changes 244
   including LOB data 243, 245
   including overriding PERIOD definitions 245
   including XML data 244, 245
   informational columns 236
   null indicator in 235
   object names in informational columns 237, 238
   overall layout of record 249
   PERIODOVERRIDE keyword 245
   RECFM keyword 233
   record format 233
   sorting of records 248
   syntax for creation 231
   TEMPLATE keyword 245
Load file definition, syntax diagram 231
Load File Pointer Table (ALPLODF) in Repository 394
LOAD keyword
   description 167
   diagram 164
load library for DB2 94
load utilities, DB2 44, 161, 179, 385
LOADFMT installation option 69
loading data into tables 250
LOB Column Description Record (DLOB) of logical log 357
LOB CREATOR keyword
   description 246
   diagram 232
LOB, inline 227
LOB/XML Cluster Data Set Record (CLUS) of logical log 361
LOBALLOC installation option 69
LOBDATA installation option 69
LOBDUPDS installation option 70
LOBLIMIT installation option 70
LOBLIMIT keyword
   description 147
   diagram 144
LOBMGMT installation option 71
LOBOPTS statement
   CYLINDERS keyword 145
   DATACLAS keyword 146
   DUPLICATE DATASET keyword 146
   LOBLIMIT keyword 147
   MGMTCLAS keyword 146
   PREFIX keyword 146
   SPACE keyword 145
   STORCLAS keyword 146
   syntax diagram 144
   TRACKS keyword 145
   VOLUME keyword 146
LOBPREF installation option 71
LOBs (large objects)
   COLUMN NAME keyword 246
   DIRBLKS keyword 265
   DSNTYPE keyword 265
   external files for LOB values 245, 246, 265
   external files for XML values 245, 265
   format 69, 234
   IGNOREFIELDS keyword 244
   INCLUDE LOBS keyword 243
   INCLUDE ROLLBACK keyword 242
   INCLUDE XML keyword 244
   including aborted transactions 242
   including compensated log records 242
   including history changes 244
   including LOB data 243, 245
   including overriding PERIOD definitions 245
   including XML data 244, 245
   informational columns 236
   null indicator in 235
   object names in informational columns 237, 238
   overall layout of record 249
   PERIODOVERRIDE keyword 245
   RECFM keyword 233
   record format 233
   sorting of records 248
   syntax for creation 231
   TEMPLATE keyword 245
Large volume column definition 232, 246
LOB CREATOR keyword 246
LOB VSAM files 144, 205
LOBALLOC installation option 69
LOBDATA installation option 69
LOBDUPDS installation option 70
LOBLIMIT installation option 70
LOBMGMT installation option 71
LOBOPTS statement 144
LOGPREF installation option 71
LOBSPACE installation option 72
LOBSTOR installation option 72
LOBVOLS installation option 72
logging of 227
restriction with UNDO SQL 227
TABLE NAME keyword 246
TEMPLATE keyword 245
UNDO SQL for 227
LOBSPACE installation option 72
LOBSTOR installation option 72
LOBVOLS installation option 72
LOCCPSEL installation option 85
Log Bytes, syntax for report 189
LOG FILES keyword
description 270
diagram 269
Log Information End Record (DLGE) of logical log 341
Log Information Record (DLGI) of logical log 340
log mark
absolute generation 274
at current time 305
at TO value 173
creating during log scan 173
creating separately 305
CURRENT keyword and 275
in Range definition 274
LAST ARCHIVE keyword and 275
LOGMARK statement 305
MARKSCAN keyword 173
names 274
QUIESCE command 173, 305
relative generation 274
text description for 173, 305
with REDO 272
Log Master for DB2
building the EXEC statement 97
building the JOB statement 96
cleaning up after a job 112
data sets 100, 101
displaying job status 109
EXEC statement 97
high-level syntax diagram 115
installation 31
load module 97
logical log records 331
operating environment 26
Repository tables 382
restarting a job 110
row completion 124
sample JCL 95
software requirements 26
starting a job 109
syntax descriptions 114
syntax diagrams 114
terminating a job 110
log record buffer 133
log scan step
completed transactions within 182
specifying overtime processing 135
start point and end point for 269
LOGBYTES keyword
description 155
diagram 154
logging of LOB objects 227
logging of XML objects 227
logical log
aliases in 203
components 332
type file format 332
data file format 369
date and time formats in 202
DDLOBJ installation option 73
DECIMAL POINT keyword 211
dependent RBA and content of 379
EXPAND VAR keyword 202
format 332, 376
INCLUDE COMMAND keyword 206
INCLUDE DDL keyword 204
INCLUDE LOBS keyword 205
INCLUDE ROLLBACK keyword 202
INCLUDE SYNONYMS keyword 203
INCLUDE XML keyword 206
including aborted transactions 202
including compensated log records 202
including DB2 commands in 206
including DDL information 204
including LOB columns in 205
including synonyms and aliases 203
including XML columns in 201, 206, 332, 359, 361
including/excluding columns from 207
input files 161
input, key store memory with 143
LOGAUTH installation option 45
ongoing processes 378
order of columns in 370
order of records in 332, 369
output files 200
overall layout of data record 370
RECFM keyword 201
record format 201
record status 333
records 332
REDO information in 271
restriction on decimal point 211
segmented logical log records 376
sorting 369
synonyms (DB2) in 203
syntax for creation 200
Unicode characters in 334, 349, 353
XML data in 201, 206, 332, 359, 361
XMLSTRING control file 201, 332, 361
XMLSTRING keyword 201

Logical log definition, syntax diagram 200
Logical Log File Pointer Table (ALPLOGF) in Repository 395

logical log records
   Clustering Index Column Description Record (DCCL) 365
   Clustering Index Information Record (DCXI) 364
   Control File ID Record (DCNT) 335
   Current Path Record (XPTH) 343
   Data Change Header Description Record (XHDF) 345
   Data Change Record (LLDF) 370
   Data Sharing Member Record (DDSM) 336
   DB2 Column Description Record Part Two (DCL2) 354
   DB2 Column Information Record (DCLI) 352
   Filter Anchor Record (DFIL) 366
   Filter Description Record (DFDS) 367
   Filter ID Record (DFID) 366
   Filter Line Data Record (DLIN) 369
   Filter Line Record (DFLN) 368
   Index Column Description Record (DICL) 364
   Index Information Record (DIXI) 362
   LOGMARK statement
      DESC keyword 305
      logMarkName variable 305
      QUIESCE keyword 305
      syntax diagram 305
      logMarkName variable
         description 173, 272, 305
         diagram 166, 269, 305
      LOGONLY keyword
         description 313
         diagram 308
      LOGOPTS statement
         LOGSORT keyword 138
         LOGTAPES keyword 139
   LOGRECORD BUFSIZE keyword
      description 133
   LOGSCAN output definition, DD name 260
   LOGSCAN statement
      ACTIVITY DESC keyword 194
      ACTIVITY TYPE keyword 194
      AFTER keyword 291
      AGE keyword 172
      ALIAS keyword 290
      ALL ACTIVITY keyword 190
      ALL keyword 172
      ALTER keyword 293
      ALTERBUFFERPOOL keyword 300
      AUTH ID keyword 195, 287
      basic syntax 164
      BEFORE keyword 291
      BEFORE OLDEST ARCHIVE keyword 172
      catalogObject variable 295
      CHANGED keyword 283
      CHARS keyword 199
      CLASS keyword 198
      COLUMN NAME keyword 246
      COLUMNS keyword 268
      COMMAND keyword 195, 290
      COMMIT FREQUENCY keyword 210, 258
      COMPRESS keyword 266
      CONCAT OPERATOR keyword 212
      CONNECTION ID keyword 195, 287
      CONNECTION TYPE keyword 195, 288
      CORRELATION ID keyword 198
      CORRELATION ID keyword 195, 287
      CREATE keyword 294
      CREATE RBA keyword 195
      CURRENT keyword 275
      COLUMNS keyword 264
      DATABASE NAME keyword 195, 288
      DATACLAS keyword 266
      dataSetName variable 260
      DATE keyword 172, 276
      DATEFMT keyword 202
      DAYS keyword 270
      DB2CATALOG keyword 174
      DB2Command variable 299
      DBID keyword 195, 288
      DDL keyword 167
      DDM NAME keyword 192
      DECIMAL POINT keyword 211
      DELETE keyword 284
      DESC keyword 173
      DEST keyword 198
      DIRBLK S keyword 265
      DNSTYPE keyword 265
      DROP keyword 294
      DSNMkeyword 196
      DURATION keyword 184
TABLESPACESET keyword 173, 305
TEMPLATE keyword 183, 217, 245, 322
TIME keyword 276
TO keyword 269
TRACKS keyword 264
TRANSACTIONS keyword 210
TYPE keyword 183
UCS keyword 199
UNDO keyword 215, 256
UNIT keyword 265
UNIT OF RECOVERY keyword 197, 287
UPDATE ALL COLUMNS keyword 211
UPDATE keyword 284
UPDATE TYPE keyword 284
UPDATES keyword 235
URID FIELDS keyword 240
URID TABLE keyword 241
USE keyword 176
value variable 280
VERBOSE keyword 210, 258
VOLUME keyword 164, 266
WHERE keyword 175
WITH ALL FIELDS keyword 197
WORKID keyword 172
WRITER keyword 199
XML CREATOR keyword 246
XMLSTRING keyword 201
LOGSORT installation option 63
LOGSORT keyword
description 138
diagram 137
LOGTAPES installation option 64
LOGTAPES keyword
description 139
diagram 137
long names. See names of DB2 objects
LRECL keyword
default values for output 263
description 199, 262
diagram 197, 262
LRPCT installation option 53
LRSN
dependent LRSN value 379
keyword, description 273
keyword, diagram 272
values in filter specification 287
LUW (logical unit of work) 289
LUW NAME keyword
description 289
diagram 285
LUW NETWORK ID keyword
description 289
diagram 285
LUW UNIQUE VALUE keyword
description 289
diagram 285

M

macro for installation options 34
MAINT parameter, in JCL 98
maintaining the Repository 385
MARK keyword
description 274
diagram 272
Mark Table (ALPMARK) in Repository 397
MARKSCAN keyword
description 173
diagram 166
mass delete action
completion within segmented table space 58, 126
generating in output 68, 131
UNDO SQL for segmented table space 215, 228
utilities that cause 44
MAXURIDS keyword
description 184
diagram 182
MAXVOL keyword
description 265
diagram 262
MEMBER ID keyword
description 196, 289
diagram 285
MEMBER NAME keyword
description 196, 289
diagram 285
member variable
description 261
diagram 260
memberName variable
description 164
diagram 162
MEMLIMIT parameter 26, 27, 96
memory
ANPCT installation option 54
BCPCT installation option 54
compression dictionaries, for 55
controlling sort memory with installation options 99
DICTIONARYSPACE keyword 128
DICTSPC installation option 55
FCPCT installation option 54
FCUSE installation option 60
for compression dictionaries 55, 128
for input/output processing 47, 151
for sorting 46, 150
installation considerations 46
key store. See key store
KSMEMORY installation option 49
LOGRECORD BUFSIZE keyword 133
LRPCT installation option 53
MEMLIMIT parameter in SMF 96
MEMORY keyword 140
MEMPERCENT keyword 142
performance and 140, 142
QBLRBUF installation option 55
REGION parameter in JCL 96
RESINV installation option 47
RESINV keyword 151
SMCORE installation option 46
SMCORE keyword 151
SORTOPTS statement 150
STOREOPTS statement 139
URPCT installation option 53
MEMORY keyword
description 140
diagram 139
MEPERCENT keyword
description 142
diagram 139
message level (msglevel), in JCL 99
MGMTCLAS keyword
description 142, 146, 149, 266
diagram 139, 144, 147, 262
MIGRATE DATASET keyword
description 316
diagram 308
MIGRATE installation option 80
MIGRATE keyword
description 130, 215, 256
diagram 120, 214, 255
migration
difference between MIGRATE and REDO SQL 216
MIGRATE DATASET keyword 316
MIGRATE DDL 256
MIGRATE keyword 215
MIGRATE SQL 215
MIGRATE SQL in drop recovery 316
migrating changes with SQL translation 101, 328
referential integrity defaults for SQL 218
RUNSTATS commands in drop recovery 316
SQLXLAT data set 101
migration of data sets
MIGRATE keyword 130
MIGRWAIT installation option 80
to archive 80, 130
waiting for migration requests 80
MIGRWAIT installation option 80
MINLOGPT
installation option 60
keyword, description of 126
keyword, in diagram 119
MOD keyword
description 261
diagram 260
MODIFY keyword
description 199
diagram 197
MTPRMBR installation option 79

N
names of DB2 objects
in load file informational columns 237, 238
in SYSIN syntax 109
shortened names 238
truncation in online interface 78
naming convention for sort work data sets 104
naming data sets with symbolic substitutions 323
native DB2 security. See security, native
NEW keyword
description 261
diagram 260
NOFREE keyword
description 307
diagram 306
NOHOLD keyword
description 198
diagram 197
NONE keyword
description 154
diagram 154
NOSCAN keyword
description 168
diagram 164
NOSORT keyword
description 248
diagram 232
NOT keyword
description 278, 283, 294, 298, 304
diagram 277, 279, 292, 295
null indicator, in load file output 235
NULL keyword
description 283
diagram 279

O
OBID keyword
description 196, 289
diagram 285
Object Activity Summary report
alternate formats for 191
subtotals and report totals in 191
syntax for 190
object names. See names of DB2 objects
OBJECT SET keyword
description 290
diagram 285
OBJECT TYPE keyword, description of 196
objects over time. See overtime mode
OLD keyword
description 261
old objects data set

OLD OBJECTS keyword
description 122
diagram 119

Old Objects Table (ALPOLDO) in Repository 398

ONGOING HANDLE keyword
description 176
diagram 166

ONGOING keyword (for Repository)
description 171
diagram 164

ongoing processing
and row completion processing 179
and versions of a table 179
dependent RBA/LRSN and 379
handleID variable 176
limit range of 270
logical log files with 378
ONGOING HANDLE keyword 176
online interface and 94
outputs at termination 110
overview of 179
performance of 179
purging data from Repository 178
Repository 110
Repository, deleting open transactions from 171
rerunning an ongoing job 176
resetting start point of 178
updating Repository 179
work IDs with 156

online Help 15

online interface
ALPOFDSN installation option 75
APFONLIN installation option 76
assigning handle ID 176
CLIST for execution 33
columns for generated WHERE clauses 209
creating JCL 94
DBCS installation option 77
default values for (POF) 75
default work ID name 157
filters 175
long names and 78
OSRCHCNT installation option 77
Repository 94
running 33
search for log files 77
SYSOUT defaults 197
TRCHARS installation option 78
TRPOS installation option 78
truncation of DB2 object names 78
truncation of names in 78

Open Transaction Record (DOTR) of logical log 342
Open Transaction report, syntax 188
Open Unit of Recovery Table (ALPURID) in Repository 399

operator variable
description 280
diagram 279

OPNDB2ID installation option 44

OPTION statement
ATTEMPT COMPLETION keyword 122
basic syntax 119
CURRENT keyword 121
DASD DATASETS keyword 131
DATEFMT keyword 121
DICTIONARYSPACE keyword 128
EARLY RECALL keyword 129
EXECUTION MODE keyword 121
FILECOST keyword 126
FILTER METHOD keyword 123
FILTERREL keyword 122
GENERATE MASSDELETE keyword 131
HEURISTIC FORWARDCOMPLETION keyword 132
IMAGESOURCE keyword 124
LOGRECORD BUFSIZE keyword 133
MIGRATE keyword 130
MINLOGPT keyword 126
OLD OBJECTS keyword 122
OVERTIME keyword 121
PER MEMBER keyword 129, 131
PROC COLD START URIDS keyword 132
PROCESS PITS keyword 128
REPOS keyword 121
RESOURCE SELECTION keyword 134
SMSTASKS keyword 129
syntax diagram 119
USELGRNG keyword 127
WAIT keyword 130
ZIIP keyword 135

OR LIMIT keyword
description 270
diagram 269

ORDER BY definition
sorting reports 194
synonyms for fields 298
syntax diagram 194

order of
columns in logical log data record 370
DDL statements 257
EXEC statement parameters 97
logical log records 332, 369
reading active/archive log files 62
reading compression dictionaries 159, 168
reading specific DB2 log files 62
reports, information within 194
Repository update jobs 136, 170
security checking 44
Index 457

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

selecting index for primary key 209
orderingField variable
description 194
diagram 194
OSRCHCNT installation option 77
OUTCOPY keyword
description 314
diagram 308
OUTCOPYDDN keyword
description 318
diagram 317
Output definition, syntax diagram 260
output files (data sets)
completed transactions in 182
default names in online interface 75
external files for LOB values 245, 265
external files for XML values 245
handling at termination 110
LOB VSAM files 205
Unicode characters in 229
XML VSAM files 206
OUTPUTDATA field of ALPOFIL
Data Set type 430
SYSOUT type 431
OVERTIME keyword
description 121
diagram 119
OVERTIME keyword (for Repository)
description 171
diagram 164
overtime mode
DB2CATALOG keyword 174
EXECMODE installation option 66
EXECUTION MODE keyword 121
obtaining structure definitions 167
OLD OBJECTS keyword 122
overview 135
processing overhead 135
Repository, deleting objects from 171
Repository, order of update jobs 136, 170
Repository, updates to 167
sequence number in load files 238
updating Repository 121, 135
Work ID Header Table (ALPWHDR) in Repository 410
Work ID Record (DWRK) of logical log 337
working with 135
OWNER keyword
description 322
diagram 320
P
packages, dynamic binding of during execution 33
parameters
MEMLIMIT 26
system 26
parameters used in JCL
DB2 subsystem ID (SSID) 98
group attachment name 98
installation options module 99
MAINT 98
message level (msglevel) 99
on the EXEC statement 97
utility ID (utilID) 99
parameters, allocation of data set 262
PARMLIB TSO command 30
PARMOPTS keyword
description 152
diagram 150
PARMSTOR installation option 48
PARMUNIT installation option 48
PART keyword
description 281
diagram 279
partitions in a table space, selecting 281
PER MEMBER keyword
description 129, 131
diagram 120
performance
analyze phase of processing 123
DB2 catalog information 123
filter processing 123
High-speed Apply Engine with SQL template file 217
key store memory and 49, 142
log record buffer and 133
ongoing processing 179
reading compression dictionaries 55
reading DB2 log files 127
recalling data sets from archive 129
Repository maintenance 385
row completion processing 59, 126
row completion processing in initial log scan 132
row completion processing of DB2 catalog 65
sort actions 46, 154
sort work data sets and 102
versioning events 179
performance, improving 90
PERIOD definitions
including in output load files 245
PERIODOVERRIDE keyword 245
PIT. See Point in Time Recovery plan
authority to bind 28, 306, 312
BINDOWN installation option 43
BINDOWNER keyword 306, 312
BINDQUALIFIER installation option 43
dynamic binding of 33
execution of generated SQL 307
FREE keyword 307
freeing SQL execution plan 307
informational column 237
PLAN keyword 297
PLAN NAME keyword 237, 287, 307
PLANALPB installation option 42
PLANALPO installation option 42
PLANEQL installation option 42
PUBLICPLAN installation option 43
rebinding in drop recovery 317
removing check pending status in drop recovery 315
selecting catalog activity for plans 297
selecting log records based on 287
sorting reports by 196
synonyms for 298
UTILPLNS installation option 44
PLAN NAME keyword
description 196, 287, 307
diagram 285, 306
PLANALPB installation option 42
PLANALPO installation option 42
PLANEQL installation option 42
POF (product options file) 75
Point in Time (PIT) recovery
generating REDO SQL 272
processing log records from 128
PREFIX keyword
description 146, 149
diagram 144, 147
PRIMARY KEY keyword
description 196, 209
diagram 208
selecting index for 209
PRIME FROM DB2 CATALOG keyword
description 168
diagram 164
PROCESS COLD START URIDS keyword
description 132
diagram 120
PROCESS PITS keyword
description 128
diagram 119
product options file. See POF
product support 3
PSID keyword
description 196, 288
diagram 285
publications, related 15
PUBLICPLAN installation option 43
PURGE keyword
description 178
diagram 166
purging data from an ongoing log scan 178
QUIESCE keyword
description 173, 185, 305
diagram 166, 182, 305
Quiet Point report
almost quiet ranges 184
excluding shorter ranges 184
quietes with 185
sorting on duration 185
syntax 184
R
RACF
authorizations 29
ID 29
OPNDB2ID installation option 44
RACF (IBM Resource Access Control Facility)
security exit 27
Range definition, syntax diagram 272
range description, limiting by time or log files 270
RBA
dependent RBA value 379
keyword, description 273
keyword, diagram 272
values in filter specification 287
RBA/LRSN keyword
description 172
diagram 164
REBIND DATASET keyword
description 317
diagram 308
RECOVER DATASET keyword
description 313
diagram 308
REBUILD installation option 74
RECOVER PLUS (BMC Software product)
drop recovery 118, 308
drop recovery without 315
DSNI1COPY, instead of 315
INDEP OUTSPACE syntax 320
parameters for 317
rebuilding indexes 74
Repository information 427
Q
QBLRBUF installation option 55
QUALIFY keyword
description 209
diagram 208
recovery

of dropped objects. See Drop Recovery point for REDO information 271
processing Point in Time (PIT) records 128
recovery parameters in drop recovery 317
REDO SQL and 215
RECOVERYDDN keyword
description 319
diagram 317

RECREATE DATASET keyword
description 312
diagram 308

REDO
action of filter with 271
difference between REDO and MIGRATE SQL 216
duplicate rows in LOAD input 226
from log mark 272
from RBA/LRSN 272
generating from LOAD RESUME actions 225
overview of 215
quiesce point 272
recovery point 271
REDO FROM keyword 271
REDO SQL 215
referential integrity defaults for 218
restriction on REDO DDL 218
REDO FROM keyword
description 271
diagram 269

REDO keyword
description 190, 215
diagram 181, 182, 214
reducing the size of the Repository 385
referential integrity, including updates resulting from 218
REGION parameter in JCL 96
REGISTER keyword
description 318
diagram 317
related publications 15
relative generation, with log mark names 274
RELEASE keyword
description 265
diagram 262
REMAP keyword
description 161
diagram 159
REMCPSSEL installation option 85
RENAME keyword
description 294
diagram 292
reorganization utilities, DB2 44, 161, 179, 385
Report definition, syntax diagram 181, 322
Report File Pointer Table (ALPRPTF) in Repository 400
REPORT keyword
description 166, 317
diagram 164, 308

Report Template Table (ALPRTMPL) in Repository 403
reports
almost quiet ranges in Quiet Point report 184
alternate formats for Summary reports 191
Audit 185
Backout Integrity (detail) 183
Backout Integrity (summary) 183
Catalog Activity 186
Commands 187
Commit 186
completed transactions in 182
CSV format for Summary reports 191
customizing 183, 322
Data Capture Changes 187
date formats for 121
Detail 187
double-byte characters in 77
Drop Recovery 317
duration of ranges in Quiet Point report 184, 185
Image Copy 188
INCLUDE ROLLBACK keyword 192
including aborted transactions 192
including compensated log records 192
including/excluding columns in 268
Log Bytes 189
Object Activity Summary 190
Open Transaction 188
ORDER BY Definition 191
ORDER BY definition 194
QUIESCE keyword 184
quiesce with Quiet Point report 185
Quiet Point 184
Rollback 189
SDF format for Summary reports 191
selecting type of 183
sort work data sets for 105
sorting 191, 194
subtotals and report totals in 191
Summary 189
Summary with BYPASSLL option 68
syntax for creation 181
templates for 183, 322
TYPE keyword 183
URID fields in Summary reports 197
REPOS DELETE keyword
description 171
diagram 164
REPOS keyword
description 121
diagram 119
REPOS UPDATE keyword
description 167
diagram 164
Repository
backing up 385
cleanup 385
compression dictionaries in 159, 168
DB2CATALOG keyword 174
deleting data from 171, 178
deleting objects from 171, 385
handle ID 176
INCLUDE DICTIONARY keyword 159, 168
input DB2 log files and 170
JCL for cleanup 385
log marks stored in 173
maintaining 167, 385
ongoing processing 110, 178, 179
online interface and 94
operational considerations 169
order of Repository update jobs 136, 170
over time mode required 121
overview 382
performance when using 385
privileges 385
purging ongoing data from 178
reducing size of 385
relationship of tables to table spaces 383
REPOS DELETE keyword 171
REPOS keyword 121
REPOS UPDATE keyword 167
restrictions 385
scan range when deleting objects from 171
SQL for cleanup 385
table relationships 384
table spaces in 383
updating 110, 121, 167, 174
updating during installation 32
updating during ongoing processing 178, 179
updating during overtime processing 135, 136, 170
updating during termination 110
updating from DB2 catalog 167
Repository tables
Automated Drop Recovery File Table (ALPADRF) 387
Column List Table (ALPCOLS) 388
DDL File Pointer Table (ALPDDLF) 388
Dropped Objects Table (ALPDRLN) 389
Filter Header Table (ALPFLTH) 390
Filter Line Table (ALPFLIN) 391
Filter Pointer Table (ALPFLTP) 394
Load File Pointer Table (ALPLODF) 394
Logical Log File Pointer Table (ALPLOGF) 395
Mark Table (ALPMARK) 397
Old Objects Table (ALPODO) 398
Open Unit of Recovery Table (ALPURID) 399
Report File Pointer Table (ALPRPTF) 400
Report Template Table (ALPRTMPL) 403
Security Exception History Table (ALPSECH) 404
SQL Codes Handler Table (ALPSQHC) 405
SQL File Pointer Table (ALPSQFL) 406
SQL Index Table (ALPIXSQ) 408
SQL Name Translation Table (ALPSQLX) 408
Work ID Header Table (ALPWHDR) 409
Work ID History Table (ALPWHIS) 413
Work ID Input File Definitions Table (ALPFSFL) 414
Work ID Log Scan History Table (ALPWLISH) 416
Work ID Output File Definitions Table (ALPOFIL) 417
Work ID Step Table (ALPWSTP) 432
Repository tables (field detail)
LINEDATA field, ALPFLIN, catalog activity type 393
LINEDATA field, ALPFSFL, input DB2 log type 415
LINEDATA field, ALPFSFL, input image copy (uncataloged) type 415, 416
LINEDATA field, ALPFSFL, input logical log type 415
LINEDATA2 field, DDL file type 426
LINEDATA2 field, drop recovery control parameters type 427
LINEDATA2 field, drop recovery out copy file type 426
LINEDATA2 field, load file type 423
LINEDATA2 field, LOB VSAM file type 428
LINEDATA2 field, logical log file type 419
LINEDATA2 field, report file type 421
LINEDATA2 field, SQL file type 420
LINEDATA2 field, XML VSAM file type 429
OUTPUTDATA field, data set type 430
OUTPUTDATA field, SYSOUT type 431
STEPDATA field, execute SQL step type 435
STEPDATA field, High-speed Apply JCL generation step type 436
STEPDATA field, log mark step type 433
STEPDATA field, log scan step type 433
STEPDATA field, repository maintenance step type 438
required JCL to run product 94
requirements, software for product 26
RERUN keyword
description 177
diagram 166
RESET keyword
description 178
diagram 166
resetting
ongoing log scan 178
run sequence number 178
RESINV installation option 47
RESINV keyword
description 151
diagram 150
RESOURCE SELECTION keyword
description 134
restarting a Log Master job 110
restrictions
completion from segmented mass delete action 126
creating UNDO DDL 256
executing generated SQL files 225, 229
explicit control of sort work data sets 105
LASTRUN keyword 177
LOB columns and UNDO SQL 227
logical log decimal point 211
mass deletes and UNDO SQL 215, 228
Repository 385
Log Master for DB2 Reference Manual
RETPD keyword
   description 266
diagram 262
RETRY parameter of DYNALLOC option 87
retrying dynamic allocation of SORTWK 87
REVOKE keyword
   description 294
diagram 292
RI. See referential integrity
Rollback report, syntax 189
row completion processing
   access order for types of image copies 125
FILECOST keyword 126
from cabinet copies 125
from DFSMS concurrent image copies 56, 125
from encrypted image copies 46, 125
heuristic forward completion 60, 132
IMAGESOURCE keyword 124
of DB2 catalog 65, 174, 255, 293, 295
ongoing log scans and 179
performance of 59, 126
summary of 124
with specific DB2 log files 163
run sequence number
   displayed in message 176
   not reset with log scan 178
   RERUN keyword 177
   USE keyword 176
running
   Log Master jobs 97, 109
   online interface 33
RUNSEQ. See run sequence number
RUNSTATS DATASET keyword
   description 316
diagram 308
S
   sample JCL for running product 95
   sample library, from installation tape 87, 385
Scan range definition
   not required when 171
   syntax diagram 269
SDF (Standard Definition Format)
   in load files 234
   in Summary reports 191
Search condition definition, syntax diagram 277
security
   AMSGLEV installation option 46
   auditing changes to 186, 258
   authorizations needed 26
   bind authority 306, 312
   DB2 table level 44
   DB2AUTH installation option 44
   external 44
   LLOGAUTH installation option 45
mechanisms 27
   native 44
   OPNDB2ID installation option 44
   order of checking 44
Security Exception History Table (ALPSECH) in
Repository 404
See names of DB2 objects
See Unicode characters in
SEGMENT keyword
   description 199
diagram 197
segmented
   logical log records, format 376
   table space, mass delete action within 58, 126, 215, 228
Selectable field definition, syntax diagram 285
selecting an alternate PRIMARY KEY 209
SEPARATE DATASETS keyword
   description 247
diagram 232
SEQNO keyword
   description 160
diagram 159
shortened names of DB2 objects 238
SHR keyword
   description 261
diagram 260
size of product Repository 385
SMCORE
   installation option 46
   keyword, description of 151
   keyword, in diagram 150
SMSTASKS
   installation option 79
   keyword, description of 129
   keyword, in diagram 120
software environment 26
Sort file size parameters, syntax diagram 154
sort routine
   passing estimates to 154
   sort work data sets and 103
sort work data sets
   control through installation options 102
   defining 102
   explicit creation 103
   names 104
   number of 105
   required number 105
   with BMCSORT 103
   with sort routine 103
SORTDYN
   installation option 49
   keyword, description of 152
   keyword, in diagram 150
software environment 26
sorting
   allocating sort work data sets 102
   DD statements 103
   load file records 248
   log records from INPUT DB2LOG 162
logical log records 369
memory allocation with SMCORE 46
number of sort work data sets 105
performance of 46, 102, 154
Quiet Point report by duration 185
records from data sharing members 196
related installation options 102
reports with ORDER BY definition 194
sort file size estimate for a job 152, 154
sort file size estimate for a log scan 174
sort file size, estimating techniques 154
sort work data sets 105
SORTOPTS statement
   AVGRECLEN keyword 156
   EST keyword 155
   FILSZ keyword 154
   HISTORY keyword 153
   LOGBYTES keyword 155
   NONE keyword 154
   PARMOPTS keyword 152
   RESINV keyword 151
   SMCORE keyword 151
   SORDYN keyword 152
   syntax diagram 150
   USE HISTORY keyword 155
   WORKOPTS keyword 153
SPACE keyword
   description 141, 145, 264
   diagram 139, 144, 147, 262
special considerations for SQL output files
duplicate rows in LOAD input 226
LOAD RESUME LOG YES 225
LOB columns, logging of 227
SQL files 225, 229
UNDO SQL and LOB columns 227
UNDO SQL of a mass delete action 228
specific DB2 log files
   as input 162
   order of reading 62
   Repository and 170
   specifying for data sharing 164
   specifying the EXEC statement 97
   specifying the JOB statement 96
SQL
   | | (concatenation operator) 212
columns in generated WHERE clauses 209
   COMMIT frequency 210
   considerations for output files 225, 229
decimal point in 211, 306
difference between REDO and MIGRATE 216
   EXECSQL statement 306
   executing. See SQL execution
   for Repository clean up 385
   generated 306
   generated string constants 212
   generating mass deletes 68
   immediate execution of generated 215
   in drop recovery 316
   including aborted transactions 221
   including all columns in updates 211
   including comments in 210
   including compensated log records 221
   including DDL in 218
   including history changes 223
   including LOB data 222
   including RI activity in 218
   including trigger activity 220
   including XML data 223
   limits for ROWID or IDENTITY columns 228
   mass deletes in 68, 131
   MIGRATE SQL 215
   output file 208
   recovery point for REDO 271
   REDO SQL 215
   referential integrity 218
   responding to SQL codes 328
   reversing deletes and updates of LOB columns 227
   reversing mass deletes 215, 228
   selecting an alternate PRIMARY KEY 209
   sort work data sets for 105
   special considerations for output 225, 229
   SQL codes 328
   SQL file definition 208
SQLCODES data set 101, 328
SQLPRINT data set 101
SQLXLAT data set, description 101
SQLXLAT data set, information in Repository 408
SQLXLAT data set, syntax diagram 328
string constants in generated 212
   template for 217
   translating names in 328, 408
   translation data set 101
   trigger activity 220
   UNDO SQL 215
   updates eliminated in 267
SQL Codes Handler Table (ALPSQLC) in Repository 405
SQL execution
   authorizations 28
   CONCAT OPERATOR keyword 212
   concatenation operator for 212
   EXECSQL statement 306
   EXECUTE keyword 215, 256
   generated SQL 306
   immediate execution 215, 256
   installation options for 42, 43
   limits on 225, 229
   plan for generated SQL 307
   responding to SQL codes 328
   results of 101
   special considerations 225, 229
   SQLID for 212
   SQLID keyword 212, 259, 312
   SQLPRINT data set 101, 307
SQL file definition, syntax diagram 208
SQL File Pointer Table (ALPSQLF) in Repository 406
SQL Index Table (ALPIXSQ) in Repository 408
SQL keywords
  CONCAT OPERATOR keyword 212
  EXECUTE keyword 215
  GENERATE MASSDELETE keyword 131
  INCLUDE HISTORY keyword 223
  INCLUDE LOBS keyword 222
  INCLUDE RI keyword 218
  INCLUDE ROLLBACK keyword 221
  INCLUDE TRIGGER keyword 220
  INCLUDE XML keyword 223
  MIGRATE DATASET keyword 316
  QUALIFY keyword 209
  RECFM keyword 217
  REDO FROM keyword 271
  SQL keyword, description 166
  SQL keyword, diagram 164
  SQLID keyword 212, 259, 312
  TEMPLATE keyword 217
  UPDATE ALL COLUMNS keyword 211
  VERBOSE keyword 210
SQL Name Translation Table (ALPSQLX) in Repository 408
SQL type/output definition, syntax diagram 214
SQLCODES data set
  DD statement 101
  syntax diagram 328
SQLDataSetName variable
description 306
  diagram 306
SQLID keyword
description 212, 259, 312
  diagram 208, 255, 308
SQLPRINT data set 101, 307
SQLPRINT keyword
description 307
  diagram 306
SQLXLAT data set
  DD statement 101
  information in Repository 408
  syntax diagram 328
SSID
  in JCL 98
  keyword, description of 197, 289
  &SSID symbolic substitution 325
Standard Definition Format (SDF)
in load files 234
  in Summary reports 191
start point of a log scan 269
starting a Log Master job 109
STATEMENTS keyword
description 210
  diagram 208
statements, syntax
  DROPRECOVERY 308
  EXECSQL 306
  IMAGECOPY 157
  INPUT 161
  LOBOPTS 144
  LOGMARK 305
  LOGOPTS 137
  LOGSCAN 164
  OPTION 119
  SORLOPTS 150
  STOREOPTS 139
  WORKID 156
  XMLLOPTS 147
STEPDATA field of ALPWSTP
  execute SQL step type 435
  High-speed Apply JCL generation step type 436
  log mark step type 433
  log scan step type 433
  repository maintenance step type 438
STELIB DD statement 100
STOGROUP keyword
description 321
  diagram 320
storage, virtual 26
storage. See memory
STOREOPTS statement
  CLUSTER keyword 142
  CYLINDERS keyword 141
  DATA keyword 142
  DATACLAS keyword 141
  MEMORY keyword 140
  MEMPERCENT keyword 142
  MGMTCLAS keyword 142
  SPACE keyword 141
  STORCLAS keyword 141
  syntax diagram 139
  TRACKS keyword 141
  VOLUME keyword 142
string constants in generated SQL 212
stringPattern variable
description 282
  diagram 279
wildcard characters in 282
substitute characters, in Unicode translation 229, 230
substitutions, symbolic 323
subsystem identifier (SSID), in JCL 98
SUBSYSTEM NAME keyword
description 289
  diagram 285
subtotals and report totals in Summary reports 191
SUMMARY keyword
description 184
  diagram 181
Summary report
  all URID fields in 197
  alternate formats for 191
  subtotals and report totals in 191
syntax 189
   with BYPASSLL option 68
support, customer 3
symbolic names 323
symbolic substitutions. See & (symbolic substitutions)
SYNONYM keyword
   description 290
   diagram 285
synonyms
   DB2 synonyms in filters 288
   DB2 synonyms in logical log files 203, 351
   for product syntax keywords 298
   in filters 298
syntax
   comments in SYSIN input 108
   descriptions 114
   diagrams. See syntax diagrams
   format for diagrams 17, 114
   inline in JCL 100
   keywords. See syntax keywords
   object names in 109
   synonyms for 298
syntax diagrams
   Allocation parameters, LOGSCAN statement 262
   Catalog activity definition, LOGSCAN statement 292
   Catalog object definition, LOGSCAN statement 295
   Column include/exclude definition, LOGSCAN statement 267
   conventions 114
   DB2 command definition, LOGSCAN statement 299
   DB2 log input definition, INPUT statement 162
   DDL file definition, LOGSCAN statement 255
   DROPRECOVERY statement 308
   EXECSQL statement 306
   Filter predicate, LOGSCAN statement 279
   high-level product syntax 115
   Image copy parameters, IMAGECOPY statement 159
   IMAGECOPY statement 157
   INPUT statement 161
   Large volume column definition, LOGSCAN statement 232
   Load file definition, LOGSCAN statement 231
   LOBOPTS statement 144
   Logical log definition, LOGSCAN statement 200
   LOGMARK statement 305
   LOGOPTS statement 137
   LOGSCAN statement 164
   Old objects data set 327
   OPTION statement 119
   ORDER BY definition, LOGSCAN statement 194
   Output definition, LOGSCAN statement 260
   Range definition, LOGSCAN statement 272
   Recover parameters, DROPRECOVERY statement 317
   Report definition, DROPRECOVERY statement 322
   Report definition, LOGSCAN statement 181
   Scan range definition, LOGSCAN statement 269
   Search condition definition, LOGSCAN statement 277
   Selectable field definition, LOGSCAN statement 285
Sort file size parameters, SORTOPTS statement 154
SORTOPTS statement 150
SQL file definition, LOGSCAN statement 208
SQL type/output definition, LOGSCAN statement 214
SQLCODES data set 328
SQLXLAT data set 328
STOREOPTS statement 139
SYSOUT definition, LOGSCAN statement 197
Temporary objects definition, DROPRECOVERY statement 320
WORKID statement 156
XMLOPTS statement 147
syntax keywords, descriptions of
   for appearance in diagrams, see syntax keywords, in diagrams
   ACTIVITY DESC 194
   ACTIVITY TYPE 194
   AFTER 291
   AGE 172
   ALIAS 290
   ALL 172
   ALL ACTIVITY 190
   ALTER 293
   ALTERBUFFERPOOL 300
   ASCODED 314
   ATTEMPT COMPLETION 122
   AUTH ID 195, 287
   AVGRECLEN 156
   BEFORE 291
   BEFORE OLDEST ARCHIVE 172
   BINDOWNER 306, 312
   BDS 161
   BYPART 314
   catalogObject variable 295
   CHANGED 283
   CHAR 199
   CHECK DATASET 315
   CLASS 198
   CLUSTER 142
cntlDatasetName variable 161
   COLUMN NAME 246
   COLUMN 268
   COMMAND 195, 290
   COMMENT ON 293
   COMMIT FREQUENCY 210, 258
   COMPRESS 266
   CONCAT OPERATOR 212
   CONNECTION ID 195, 287
   CONNECTION TYPE 195, 288
   COPIES 198
   CORRELATION ID 195, 287
   CREATE 294
   CREATE RBA 195
   CURRENT 121, 275
   CYLINDERS 141, 145, 148, 264
   DASD DATASETS 131
   DATA 142
   DATABASE 321
XMLSTRING 201
ZIP 135
syntax keywords, in diagrams
  for text descriptions of syntax, see syntax keywords,
  descriptions of
AFTER 285
AGE 164
ALIAS 285
ALL 164
ALL ACTIVITY 182
ALTER 292
ASCODED 308
ATTEMPT COMPLETION 119
AUTH ID 285
AVGRECLEN 154
BEFORE 285
BEFORE OLDEST ARCHIVE 164
BINDOWNER 306, 308
BSDS 161
BYPART 308
catalogObject variable 295
CHANGED 279
CHARS 197
CHECK DATASET 308
CLASS 197
CLUSTER 139
cntlDataSetName variable 161
COLUMN NAME 232
COLUMNS 267
COMMENT ON 292
COMMIT FREQUENCY 208, 255
COMPRESS 262
CONCAT OPERATOR 208
CONNECTION ID 285
CONNECTION TYPE 285
COPIES 197
CORRELATION ID 285
CREATE 292
CURRENT 119, 272
CYLINDERS 139, 144, 147, 262
DASD DATASETS 120
DATA 139
DATABASE 320
DATABASE NAME 285, 308
databaseName variable 157
DATACLAS 139, 144, 147, 262
DATASET 159
dataSetName variable 162, 260
DATE 164, 272
DATEDATE 119, 200
DAYS 269
DB2CATALOG 166
DB2LOG 161
DBID 285
DDL 164
DDNAME 181
DECIMAL POINT 208, 306
DELETE 279
DESC 156, 166, 305
DEST 197
DEV 159, 162
DICTIONARYSPACE 119
DIRBLKS 262
DROP 292
DSNCPYdataset 308
DSNTYPE 262
DSNUM 159
DUPLICATE DATASET 144, 147
DURATION 182
EARLY RECALL 120
EST 154
EXCHANGE 279
EXCLUDE 267
EXCLUDE NULL INDICATOR 231
EXECUTION MODE 119
EXPAND VAR 200, 231
EXPDT 262
FCB 197
FILECOST 119
FILSZ 154
FILTER 166
FILTER METHOD 119
FILTERREL 119
FORMAT 182, 231
FORMS 197
FREE 306
FROM 269
GEN. EMPTY FILES 232
GENERATE MASSDELETE 120
GRANT 292
HEURISTIC FORWARDCOMPLETION 120
HISTORY 150
HOLD 197
ICTYPE 159
IMAGESOURCE 119
IMRBA 159
IN 279, 292, 295
INCLUDE 267
INCLUDE COMMAND 200
INCLUDE DDL 200, 214
INCLUDE DICTIONARY 164
INCLUDE HISTORY 214
INCLUDE LOBS 200, 214, 232
INCLUDE RI 214
INCLUDE ROLLBACK 182, 200, 214, 232, 255
INCLUDE SYNONYMS 200
INCLUDE TOTAL 182
INCLUDE TRIGGER 214
INCLUDE XML 200, 214, 232
INDEP 320
INFO COLUMNS 231
INSERT 279
LABEL ON 292
LAST ARCHIVE 272
LASTRUN 166
LIKE 262, 279
LLOG 161, 164
UCS 197
UNDO 214, 255
UNIT 262
UNIT OF RECOVERY 285
UPDATE 279
UPDATE ALL COLUMNS 208
UPDATE TYPE 279
UPDATES 231
URID FIELDS 231
URID TABLE 231
USE 166
USE HISTORY 154
USELGRNG 119
userID variable 156
value variable 279
VERBOSE 208, 255
VOLSER 159
VOLUME 139, 144, 147, 162, 262
WAIT 120
WHERE 166
WORKID 164
workID variable 156
WORKOPTS 150
WRITER 197
XML CREATOR 232
XMLLIMIT 147
XMLSTRING 200
ZIIP 120

syntax statements. See statements, syntax
SYS1.PARMLIB data set 30
SYSADM authority 28
SYSCOPY table of DB2. See SYSEBMS.SYSCOPY table in
DB2 Catalog
SYSCTRL authority 28
SYSSERR DD statement. See ALPDUMP
SYSEBMS.SYSCOPY table in DB2 Catalog
DCC attribute and 65
image copies in 157
PIT ranges and 128
Quiet Point report 184
REDO from last quiesce action 272
selecting activity in 297
utility updates to 44
SYSEBMS.SYSINDEXES table in DB2 Catalog 209
SYSEBMS.SYSSLGRNX table in DB2 Directory 59, 61, 127
SYSEBMS.SYSSYNONYMS table in DB2 Catalog 203
SYSEBMS.SYSXMLSTRINGS table in DB2 Catalog 201
SYSIN
comments in Log Master syntax 108
DD statement 100
Log Master syntax in separate data set 107
SYSEBMS.SYSINDEXES table of DB2, product use of 209
SYSEBMS.SYSSLGRNX table of DB2, product use of 59, 61, 127
SYSEBMS.SYSSDUMP data set 101
SYSEBMS.SYSSOUT data set 100
SYSEBMS.SYSSOUT definition, syntax diagram 197
SysStrings Base Record (DSS1) of logical log 343
SysStrings TransTable Record (DSS2) of logical log 344
SYSEBMS.SYSSYNONYMS table of DB2, product use of 203
SYSEBMS.SYSSDUMP data set 101
SYSEBMS.SYSXMLSTRINGS table of DB2, product use of 201

T

Table Byte Count Record (DTBB) of logical log 349
Table Information Record (DTBI) of logical log 347
TABLE keyword
description 268
diagram 267
TABLE NAME keyword
description 197, 246, 288, 311
diagram 232, 285, 308
table name, shortened 238
Table Record Count Record (DTBC) of logical log 350
TABLE SEQUINNUM field, in load files 238
Table Space Information Record (DTSI) of logical log 346
table spaces
in Repository 383
LOB, logged or not logged 227
mass delete within segmented, risks 58, 126, 215, 228
recovering with automated drop recovery 310
selecting a partition in 281
selecting log records from 288
selecting on activity affecting 297
XML, logged or not logged 227
Table Synonym and Alias Record (DTOB) of logical log 351
TABLESPACE keyword
description 321
diagram 320
TABLESPACE NAME keyword
description 158, 197, 288, 310
diagram 157, 285, 308
tableSpaceName variable
description 158
diagram 157
TABLESPACESET keyword
description 173, 305
diagram 166, 305
&TABNAME symbolic substitution 324
TAPES installation option 74
technical support 3
TEMPLATE keyword
description 183, 217, 245, 322
diagram 181, 214, 232
templates
external files for LOB/XML data in load files 245
reports, customizing with 183, 322
SQL, improve performance with 217
Temporary objects definition, syntax diagram 320
terminating a Log Master job 110
termination processing, Log Master jobs 110
time format in logical log 202
TIME keyword
description 276
diagram 272
&TIME symbolic substitution 325
time value, limit range by 270
TO keyword
creating a log mark at 173
description 269
diagram 269
totals, report and subtotals in Summary reports 191
TRACKS keyword
description 141, 145, 148, 264
diagram 139, 144, 147, 262
UNLOAD PLUS for DB2 (BMC Software product) 69, 234
UNDO keyword
description 215, 256
diagram 214, 255
Undo keyword
description 215, 256
diagram 214, 255
Unid characters in
column data, substitute characters 229
Unicode characters in
column data, substitute characters 229
UNDO DDL 256
UNDO SQL 215
UNDO SQL and LOB columns 227
UNDO keyword
description 215, 256
diagram 214, 255
Unicode characters in
column data, substitute character 230
column data, substitute characters 229
UNDO keyword
description 215, 256
diagram 214, 255
Unicode characters in
column data, substitute character 230
column data, substitute characters 229
unit of recovery identifier. See URID
UNIT OF RECOVERY keyword
description 197, 287
diagram 285
UNITWAIT installation option 64
UNLOAD PLUS for DB2 (BMC Software product) 69, 234
UPDATE ALL COLUMNS keyword
description 211
diagram 208
UPDATE keyword
description 284
diagram 279
UPDATE TYPE keyword
description 284
diagram 279
UPDATE keyword
description 235
diagram 231
updates, eliminated from SQL 267
updating Repository
during installation 32
during ongoing log scan 179
order of jobs 136, 170
URID (unit of recovery identifier)
cold start and 132
comments in DDL 258
comments in SQL 210
COMMIT frequency 210, 258
fields (all fields in Summary reports) 197
fields (extra columns in load files) 240, 250
filter specification of 287
layout of extra columns and table 250
ONGOING HANDLE keyword 176
RBA/LRSN values of 287
separate load file output 241
sorting based on 197
synonyms for 299
threshold (limit) on changes in 67
UNIT OF RECOVERY keyword 197
unterminated from cold start 132
URID FIELDS keyword 240
URID TABLE keyword 241
URIDTHR installation option 67
VERBOSE keyword 258
with ongoing processing 179
URID FIELDS keyword
description 240
diagram 231
URID TABLE keyword
description 241
filter specifications 280
logical log records 334, 349, 353
object names, substitute characters 229, 230
output 229
string comparisons 282
V

value variable
   description 280
diagram 279

variable blocked files
   DDL files 257
   load files 233
   logical log files 376
   SQL files 217

verbose DDL 258

VERBOSE keyword
   description 210, 258
diagram 208, 255

verbose SQL 210

versions of DB2 tables
   drop recovery and 309
   ongoing processing and 179

eentral storage 26

virtual storage. See memory

VOLSER keyword
   description 160
diagram 159

VOLUME keyword
   description 142, 146, 149, 164, 266
diagram 139, 144, 147, 162, 262

VSAM
   buffering tool, use with product 30
   files for large objects (LOBs) 144, 205
   files for XML data 147, 206

W

WAIT keyword
   description 130

diagram 120

waiting for data set migration requests 80

WHERE keyword
   description 175
diagram 166

wildcard characters for string comparisons 282

wildcard characters, in string comparisons 282

WITH ALL FIELDS keyword 197

WKSTOR installation option 48

WKUNIT installation option 48

Work ID Description Record (DWKID) of logical log 337

Work ID Header Table (ALPWHDR) in Repository 409

Work ID History Table (ALPWHIS) in Repository 413

Work ID Input File Definitions Table (ALPIFIL) in Repository 414

Work ID Log Scan History Table (ALPWLSH) in Repository 416

Work ID Output File Definitions Table (ALPOFIL) in Repository 417

Work ID Record (DWRK) of logical log 336

Work ID Step Table (ALPWSTP) in Repository 432

WORKID keyword (for Repository)
   description 172
diagram 164

WORKID statement
   DESC keyword 157
syntax diagram 156
   userID variable 156

workID variable
   description 157
diagram 156

&WORKID symbolic substitution 324

XML

COLUMN NAME keyword 246

external files for XML values 245, 246

INCLUDE XML keyword 206, 223, 244

including in generated SQL 223

including in logical log files 201, 206, 332, 359

including in output load files 244

Large volume column definition 232, 246

logging of 227

TABLE NAME keyword 246

TEMPLATE keyword 245

XML CREATOR keyword 246

X

XHDF record of logical log 345

See memory
XML VSAM files 147, 206
XMLOPTS statement 147
XMLPREF installation option 73
XMLSTRING control file 201, 332, 361

XML Column Description Record (DXML) of logical log 359

XML CREATOR keyword
   description 246
   diagram 232

XML String Data Set Name Record (DXSF) of logical log 361

XMLLIMIT keyword
   description 150
   diagram 147

XMLOPTS statement
   CYLINDERS keyword 148
   DATACLAS keyword 149
   DUPLICATE DATASET keyword 149
   MGMTCLAS keyword 149
   PREFIX keyword 149
   STORCLAS keyword 149
   syntax diagram 147
   TRACKS keyword 148
   VOLUME keyword 149
   XMLLIMIT keyword 150

XMLPREF installation option 73
XMLSTRING control file of logical log 201, 332, 361
XMLSTRING keyword
   description 201
   diagram 200

XPTH record of logical log 343
XTYP record of logical log 333

Z

ZIIP installation option 84
ZIIP keyword
   description 135
   diagram 120
&ZPREFIX symbolic substitution 325