ALTER and CHANGE MANAGER®
for DB2®
User Guide, Volume 2

Supporting
Version 11.1 Administrative Assistant for DB2
Version 11.1 ALTER for DB2
Version 11.1 CHANGE MANAGER for DB2
Version 10.2 Database Administration for DB2

June 2013
**Contacting BMC Software**

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

**United States and Canada**

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

**Outside United States and Canada**

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

© Copyright 1998-2013 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.

DB2, DFSMS, DFSMShsm, IBM, MVS, OS/390, QMF, RACF, REXX, z/OS, and zSeries are the trademarks or registered trademarks of International Business Machines Corporation.

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 UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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
- Related publications ................................................. 23
- Conventions ......................................................... 24
- Syntax statements. .................................................. 25
  - Syntax diagrams ............................................... 26

### Part 1 Managing objects with ALTER and CHANGE MANAGER 29

<table>
<thead>
<tr>
<th>Chapter 1 Migrating database environments</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before you begin ..................................</td>
<td>32</td>
</tr>
<tr>
<td>Overview .............................................</td>
<td>33</td>
</tr>
<tr>
<td>Migrating data structures ........................</td>
<td>33</td>
</tr>
<tr>
<td>Deciding how to migrate data structures .....</td>
<td>37</td>
</tr>
</tbody>
</table>
  - Using migrate-type work IDs .................. | 37 |
  - Using outbound migrate profiles ............. | 38 |
  - Using Data Manipulation Language ............ | 39 |
| Defining the scope of a migration ........... | 40 |
  - Defining the scope with a profile .......... | 40 |
  - Specifying the data structures for a migration | 41 |
  - Identifying changes to data structures .... | 45 |
| Specifying change rules for a migration .. | 46 |
  - Specifying change rules for a migrate-type work ID | 47 |
  - Specifying change rules for an outbound migrate profile | 48 |
  - Specifying values for change rule attributes | 49 |
  - Resolving change rules ......................... | 53 |
| Specifying migrate options .................... | 57 |
| Analyzing the effects of a migration .......... | 58 |
  - Analyzing requests using work IDs .......... | 59 |
  - Analyzing requests using migrate profiles  | 60 |
  - Selecting the run type ......................... | 60 |
  - Overriding the default processing options | 61 |
  - Validity and performance checks ............. | 73 |
  - Reviewing the results of Analysis .......... | 78 |
| Executing changes ................................. | 82 |
  - Specifying options for Execution .......... | 82 |
  - Controlling authorizations .................... | 94 |
  - Processing a worklist ........................... | 98 |
Reviewing the results of a migration ............................................. 103
Restart methods ........................................................................... 106
Accomplishing your goals ............................................................... 113
Creating a migrate-type work ID .................................................... 114
Creating a migrate-type work ID in batch mode ................................. 115
Defining migrate options for a migrate-type work ID ....................... 116
Creating change rules for a migrate-type work ID ............................ 117
Converting an external SQL procedure to a native stored procedure .... 118
Specifying the data structures in a migrate-type work ID .................. 119
Migrating the auxiliary objects for a LOB column ............................. 121
Migrating data structures by using DML ......................................... 123
Assigning a unique name by importing DML ................................. 128
Defining primary and secondary quantities by importing DML ............ 129
Setting a secondary quantity to a factor of a primary quantity by importing DML .......................................................... 131
Changing table spaces and indexes to DEFINE NO by importing DML .... 132
Migrating a partitioned table space to a range-partitioned table space by importing DML ....................................................... 134
Migrating a single-table table space to a partition-by-growth table space by importing DML ....................................................... 136
Migrating tables from an explicit database to an implicit database by importing DML .......................................................... 137
Migrating tables that contain inline LOB columns by importing DML .... 138
Migrating stored procedures by importing DML ............................... 140
Migrating data structures by using a migrate-type work ID .................. 141
Creating a stop list ........................................................................ 144
Creating an outbound migrate profile .......................................... 146
Creating scope rules for an outbound migrate profile ........................ 148
Specifying locations in an outbound migrate profile ........................ 149
Creating change rules for a migrate profile .................................... 151
Analyzing a migrate-type work ID and generating a worklist ............. 153
Creating a receive-type work ID ..................................................... 155
Executing a migrate-type worklist .................................................. 158
Restarting a worklist ..................................................................... 164
Starting a worklist over .................................................................. 166
Where to go from here ................................................................. 168

Chapter 2 Taking a snapshot of a database environment 169

Before you begin ......................................................................... 170
Overview ...................................................................................... 170
Using baselines ............................................................................ 173
Establishing baselines ................................................................. 173
Determining what type of baseline to establish ................................. 174
Creating baseline profiles ............................................................. 175
Creating structure-only baselines ................................................. 180
Using a DB2 catalog to create a catalog baseline ............................. 180
Using a DDL file to create a DDL baseline ..................................... 181
Using a migrate-type worklist to create a DDL baseline ................... 182
Creating full-recovery baselines .................................................... 183
Chapter 4 Altering data structures in a database environment 307

Before you begin ........................................ 310
Overview ........................................ 310
Altering data structures .................................. 314
Specifying changes to data structures ................. 314
  Identifying changes to data structures .............. 316
  Specifying changes to auxiliary objects for a LOB column 317
  Converting table spaces .................................. 320
  Considerations for changing data structures .......... 326
Importing files .......................................... 328
  Deciding what type of file to import .................. 329
  Deciding the type of work ID to use ................. 332
  Validity checks ......................................... 334
  Reviewing the results of the import .................. 335
Estimating space ....................................... 336
  Estimating space for a table space ................... 337
  Estimating space for an index ....................... 341
Estimating space requirements based on user-specified values 344
Analyzing changes ................................... 345
  Overriding the default processing options .......... 346
  Validity and performance checks .................... 347
  Reviewing the results of Analysis .................. 347
Generating JCL ....................................... 348
Executing changes ..................................... 349
  Specifying options for Execution .................... 349
  Controlling authorizations ............................ 352
  Processing a worklist .................................. 352
  Reviewing the results of your changes ............... 353

Results of a comparison ................................ 261
Diagnostic log ......................................... 262
CDL file ........................................ 262
Generating a CDL file from an alter-type work ID .......... 263
Applying the CDL ..................................... 264
Improving the performance of a comparison ............ 264
Accomplishing your goals ................................ 266
  Comparing a DB2 catalog to a DB2 catalog .......... 266
  Using a profile to scope a catalog to a catalog comparison 272
  Comparing a DB2 catalog to a DDL file ............ 276
  Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist 280
  Comparing a baseline to a baseline .................. 283
  Comparing a baseline to a DB2 catalog .......... 287
  Comparing a DB2 catalog to a baseline .......... 290
  Comparing a baseline to a DDL file .......... 294
  Generating a CDL file from a work ID .......... 297
  Using a script to perform a comparison ........... 300
Where to go from here ................................ 306

Reviewing the results of Analysis .......................... 347
Validity and performance checks ........................ 347
Overriding the default processing options ............... 346
Reviewing the results of Analysis .......................... 347
Generating JCL ....................................... 348
Executing changes ..................................... 349
  Specifying options for Execution .................... 349
  Controlling authorizations ............................ 352
  Processing a worklist .................................. 352
  Reviewing the results of your changes ............... 353
Accomplishing your goals .............................................................. 354
Creating an alter-type work ID ...................................................... 357
Creating an alter-type work ID in batch mode .............................. 358
Sorting work IDs ......................................................................... 359
Deleting a work ID ....................................................................... 360
Creating an inbound migrate profile in CHANGE MANAGER ....... 361
Replicating multiple work IDs ....................................................... 362
Modifying a work ID with values in a user-defined table .............. 366
Changing column definitions for imported DDL ............................ 370
Deleting multiple work IDs ............................................................ 373
Deleting sync table entries for multiple work IDs ....................... 376
Importing a CDL, DDL, or DML file in CHANGE MANAGER ....... 379
Importing a DDL file in ALTER ..................................................... 381
Specifying changes to data structures .......................................... 383
Creating a primary constraint for a table ..................................... 384
Creating a unique constraint for a table ....................................... 386
Editing the columns for a unique constraint ............................... 390
Editing a primary key ................................................................... 392
Naming a primary key .................................................................. 395
Changing the attributes of an index .............................................. 397
Creating indexes for ROWID GENERATED ALWAYS columns by
importing DML ........................................................................... 400
Adding a column to a table ........................................................... 402
Dropping a column from a table ................................................... 404
Changing the data type and length of a column ......................... 405
Moving multiple tables using Quick Edit ..................................... 406
Changing data structures by using an existing work ID ............... 408
Receiving DDL to create data structures ...................................... 411
Changing data structures by using DML ...................................... 415
Adding columns to the DB2 PLAN_TABLE table by importing DML 420
Deleting aliases by importing DML .............................................. 422
Copying a column by importing DML .......................................... 423
Updating index partitions for a database by importing DML .......... 424
Adding WITH RESTRICT ON DROP to tables by importing DML ..... 426
Updating table spaces or indexes with existing data sets by importing DML 427
Creating synonyms for selected tables by importing DML .......... 428
Converting VCAT-defined partitions to STOGROUP-defined partitions by
importing DML ........................................................................... 429
Creating a materialized query table (MQT). ................................. 431
Creating an application-period temporal table ............................ 433
Creating a system-period temporal table ..................................... 435
Creating a history table for a system-period temporal table .......... 437
Creating a history table from an existing history table .................. 440
Creating a history table from a base table ................................. 443
Creating an external stored procedure ....................................... 446
Creating a native SQL stored procedure ..................................... 447
Creating a new version of a native SQL stored procedure .......... 449
Updating options for a native SQL stored procedure by importing DML 451
Updating parameters for a native SQL stored procedure by importing DML 452
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimating the space requirements for a table space</td>
<td>453</td>
</tr>
<tr>
<td>Estimating the space requirements for an index</td>
<td>460</td>
</tr>
<tr>
<td>Estimating space requirements for a table space based on user-specified values</td>
<td>464</td>
</tr>
<tr>
<td>Estimating space requirements for an index based on user-specified values</td>
<td>466</td>
</tr>
<tr>
<td>Setting space estimation parameters by importing DML</td>
<td>467</td>
</tr>
<tr>
<td>Estimating the primary and secondary quantities in table space and index partitions by importing DML</td>
<td>470</td>
</tr>
<tr>
<td>Setting a secondary quantity in table space and index partitions by importing DML</td>
<td>471</td>
</tr>
<tr>
<td>Creating objects for the SHRLEVEL CHANGE HSSC process</td>
<td>473</td>
</tr>
<tr>
<td>Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC process</td>
<td>476</td>
</tr>
<tr>
<td>Converting table spaces to partition-by-growth table spaces by importing DML</td>
<td>479</td>
</tr>
<tr>
<td>Converting a nonpartitioned table space to an index-controlled partitioned table space</td>
<td>480</td>
</tr>
<tr>
<td>Converting a nonpartitioned table space to a table-controlled partitioned table space</td>
<td>486</td>
</tr>
<tr>
<td>Converting a nonpartitioned table space to a range-partitioned table space</td>
<td>490</td>
</tr>
<tr>
<td>Converting a nonpartitioned table space to a partition-by-growth table space</td>
<td>491</td>
</tr>
<tr>
<td>Converting a partitioned table space to a nonpartitioned table space</td>
<td>493</td>
</tr>
<tr>
<td>Changing a partitioned table space to a range-partitioned table space by importing DML</td>
<td>495</td>
</tr>
<tr>
<td>Converting a table-controlled partitioned table space to an index-controlled partitioned table space</td>
<td>497</td>
</tr>
<tr>
<td>Converting table-controlled partitioned table spaces to range-partitioned table spaces by importing DML</td>
<td>498</td>
</tr>
<tr>
<td>Converting a range-partitioned table space to an index-controlled partitioned table space</td>
<td>500</td>
</tr>
<tr>
<td>Converting an index-controlled partitioned table space to a table-controlled partitioned table space</td>
<td>500</td>
</tr>
<tr>
<td>Converting index-controlled partitioned table spaces to table-controlled partitioned table spaces by importing DML</td>
<td>504</td>
</tr>
<tr>
<td>Converting an index-controlled partitioned table space to a range-partitioned table space</td>
<td>505</td>
</tr>
<tr>
<td>Converting index-controlled partitioned table spaces to range-partitioned table spaces by importing DML</td>
<td>506</td>
</tr>
<tr>
<td>Converting a partitioned table space to a partition-by-growth table space</td>
<td>507</td>
</tr>
<tr>
<td>Converting a partition-by-growth table space to an index-controlled partitioned table space</td>
<td>510</td>
</tr>
<tr>
<td>Converting a partition-by-growth table space to a table-controlled partitioned table space</td>
<td>512</td>
</tr>
<tr>
<td>Converting a partition-by-growth table space to a range-partitioned table space</td>
<td>513</td>
</tr>
<tr>
<td>Changing a single-table table space to a partition-by-growth table space by importing DML</td>
<td>514</td>
</tr>
<tr>
<td>Changing tables in an explicit database to an implicit database by importing DML</td>
<td>515</td>
</tr>
</tbody>
</table>
Converting explicit databases and implicit table spaces to implicit databases
and table spaces by importing DML .................................................. 517
Changing the encoding scheme for a table space by importing DML .... 518
Adding a ROWID and a LOB column to create a base table ............... 519
Creating auxiliary objects for a partitioned table space .................... 525
Creating auxiliary objects for a single partition in a partitioned table space . 531
Liking a base table and creating auxiliary objects for a table space .... 536
Creating auxiliary objects for a nonpartitioned table space ............... 538
Converting a VARCHAR data type to a LOB data type .................... 543
Creating a new database and new table spaces for tables by importing DML . 548
Changing multi-table table spaces to single-table table spaces by importing
DML .................................................................................................. 550
Moving all empty tables in a database to one table space by importing DML . 552
Setting table spaces and indexes to DEFINE NO by importing DML .... 554
Updating table spaces or indexes defined with DEFINE NO by importing
DML .................................................................................................. 557
Updating the length of an inline LOB column by importing DML ....... 558
Analyzing an alter-type work ID and generating a worklist in CHANGE
MANAGER ......................................................................................... 560
Executing an alter-type worklist ....................................................... 562
Generating a multi-step job ............................................................. 564
Where to go from here .......................................................................... 567

Part 2  Managing environments with ALTER and CHANGE
MANAGER ........................................................................................... 569

Chapter 5  Maintaining database environments ................................. 571
Before you begin .................................................................................. 571
Overview .............................................................................................. 572
Migrating changes to data structures .................................................. 572
  Acting as the sending subsystem .................................................... 573
  Acting as the receiving subsystem .................................................. 575
Maintaining applications ....................................................................... 577
  Implementing a strategy ................................................................. 578
Recovering data structures ................................................................. 579
Feeding back changes to data structures .......................................... 580
Using data modeling tools ................................................................... 581
  Importing DDL from a data modeling tool ...................................... 582
  Maintaining catalog information .................................................. 583
Accomplishing your goals .................................................................... 583
  Migrating data structure changes to other subsystems .................. 584
  Merging subsystems by using ALTER .......................................... 585
  Receiving data structure changes from a strong sender ................. 587
  Receiving data structure changes from a strong receiver ............... 590
  Restoring data structures with current data ................................. 595
  Recovering data structures with current data ............................... 596
Chapter 6  Managing your PeopleSoft environment with CHANGE MANAGER  609

Overview ................................................................. 609
Working in a PeopleSoft environment ................................. 610
Meeting the challenges of a PeopleSoft environment ............... 611
  Taking a snapshot of the initial environments ...................... 611
  Creating copies of an existing environment ....................... 613
  Optimizing the design of an environment ......................... 618

Part 3  Using tools and utilities with ALTER and CHANGE MANAGER  623

Appendix A  Using scripting tools to automate change management  625

Overview .................................................................. 626
Using task IDs ............................................................ 627
  Creating a task ID ...................................................... 628
  Listing task IDs ........................................................ 630
  Editing a task ID ...................................................... 631
  Executing a task ID .................................................... 632
  Viewing the execution status of a task ID ......................... 639
  Browsing a task ID .................................................... 640
  Deleting a task ID ...................................................... 641
Processing CM/PILOT worklists ........................................ 642
  Restarting a CM/PILOT worklist .................................. 642
  Starting a CM/PILOT worklist over .............................. 644
Using scripts .................................................................. 646
  Creating a script ....................................................... 648
  Listing scripts .......................................................... 659
  Editing a script ......................................................... 660
  Listing the steps for a script ........................................ 661
  Executing a user-defined script .................................... 662
  Browsing a script ...................................................... 662
  Deleting a script ....................................................... 663
  Creating a task ID from a script .................................... 664
  Listing task IDs for a script ........................................ 665
  Copying a script supplied by BMC ................................. 665
  Copying a user-defined script ...................................... 666
Using applications ......................................................... 666
  Creating an application .............................................. 668
  Listing applications .................................................. 669
  Editing an application ............................................... 669
  Browsing an application ............................................. 670
  Deleting an application .............................................. 671
Contents 13

Creating a task ID from an existing application ..................................... 671
Listing task IDs for an application ...................................................... 672

Appendix B Using utilities with ALTER and CHANGE MANAGER 673
Overview ......................................................................................... 674
Using the unload utilities ................................................................. 674
  Using the BMC BASIC UNLOAD utility ........................................... 675
  Using the BMC UNLOAD PLUS utility ............................................ 676
  Using the IBM UNLOAD utility ...................................................... 686
  Using the BMC RECOVER PLUS utility ......................................... 689
  Using the EXPORT and IMPORT commands .............................. 691
  Unloading Unicode, LOB, and XML data ..................................... 694
Using the load utilities ............................................................... 694
  Using the BMC LOADPLUS utility ................................................ 695
  Using the IBM LOAD utility .......................................................... 697
  Loading Unicode, LOB, and XML data ....................................... 697
  Migrating tables with ROWID columns ....................................... 698
Using the check utilities ............................................................. 698
  Using the BMC CHECK PLUS utility ............................................ 698
  Using the IBM CHECK DATA utility ............................................ 698
Using the reorg utilities .............................................................. 699
  Using the BMC REORG PLUS utility ............................................ 700
  Using the IBM REORG utility ....................................................... 701
  Generating SHRLEVEL reorgs ..................................................... 701
  Dynamically allocating reorg data sets ...................................... 703
  Considerations for using the utilities ......................................... 703
Using the recover and rebuild utilities ........................................ 705
  Using the BMC RECOVER PLUS utility ..................................... 706
  Using the IBM REBUILD INDEX utility ...................................... 706
  Considerations for using the utilities ......................................... 706
Using the copy utilities .............................................................. 707
  Using the BMC COPY PLUS utility ............................................. 708
  Using the IBM COPY utility ........................................................ 708
  Copying and registering the data sets ....................................... 708
  Creating image copies ............................................................... 709
  Dynamically allocating copy data sets ..................................... 712
Using the statistics utilities .......................................................... 715
  Collecting statistics ................................................................... 715
  Updating statistics ..................................................................... 718

Appendix C Using the skeleton library compiler 719
Overview ......................................................................................... 719
Compiling SLIBs ............................................................................ 720
Changing SLIBs ............................................................................ 721
  Testing changes using ISPF file tailoring ................................. 722
  Compiling changed SLIBs .......................................................... 723
Processing SLIBs .......................................................................... 723
  Generating the SLIB report ....................................................... 723
### Appendix D  Using worklist parallelism

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of worklist parallelism</td>
<td>727</td>
</tr>
<tr>
<td>Considerations for enabling worklist parallelism</td>
<td>729</td>
</tr>
<tr>
<td>Controlling XIM execution</td>
<td>733</td>
</tr>
<tr>
<td>Determining the status of XIM</td>
<td>734</td>
</tr>
<tr>
<td>Starting XIM</td>
<td>735</td>
</tr>
<tr>
<td>Inactivating XIM initiators</td>
<td>736</td>
</tr>
<tr>
<td>Shutting down XIM</td>
<td>736</td>
</tr>
<tr>
<td>Activating XIM initiators</td>
<td>737</td>
</tr>
<tr>
<td>Modifying MVS image variables</td>
<td>737</td>
</tr>
<tr>
<td>Troubleshooting the execution of XIM</td>
<td>739</td>
</tr>
<tr>
<td>Processing a worklist in parallel</td>
<td>740</td>
</tr>
</tbody>
</table>

### Glossary

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glossary</td>
<td>743</td>
</tr>
</tbody>
</table>

### Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>765</td>
</tr>
</tbody>
</table>
Figures

Workflow for the migration process (part 1 of 2) ........................................ 35
List of DB2 data structures ................................................................. 42
Unattached objects .............................................................................. 43
Work ID change rule .......................................................................... 47
Specifying a range of values ............................................................ 49
Using the maximum value plus one to specify all values ................. 50
Specifying a new value ...................................................................... 51
Specifying a default value with a change rule ................................. 52
Specifying a <BLANK> value with a change rule ............................... 53
Subsequent change rules ................................................................. 54
Application of change rules .............................................................. 55
Analysis Worklist Options panel ....................................................... 62
Analysis Worklist Commands Options panel—migrate-type work ID .... 63
Analysis Input Stream Parameters panel—migrate-type Work ID .......... 67
Explicit view column list ................................................................. 68
Implicit view column list .................................................................. 69
Analysis Input Stream Parameters panel—alter-type work ID ............ 72
Execution JCL Processing Interface panel—foreground ....................... 86
Execution JCL Processing Interface panel—batch ................................ 87
Execution Override Options panel—migrate-type work ID ................. 88
Execution Status panel .................................................................... 100
Auxiliary Objects List panel ............................................................. 122
DML for a MIGRATE statement ....................................................... 124
DML for assigning a unique name ................................................... 128
DML for setting primary and secondary quantities ............................ 130
Example DML for setting SQTY ........................................................ 131
DML to migrate table spaces and indexes ......................................... 133
DML to migrate table-controlled partitioned table spaces to range-partitioned table spaces ......................................................... 134
DML to migrate table-controlled partitioned table spaces to range-partitioned table spaces by using a JOIN statement .......................... 134
DML to migrate index-controlled partitioned table spaces to range-partitioned table spaces ......................................................... 135
DML to migrate a table space to a partition-by-growth table spaces ........ 136
DML to migrate tables from an explicit database to an implicit database .... 137
DML to migrate tables from an explicit database to an implicit database using a JOIN ................................................................. 137
DML to migrate tables with CLOB columns ...................................... 139
DML to migrate all dependent objects of an inline CLOB column ......... 139
DML to migrate stored procedures .................................................. 140
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of the NEWWORKID keyword</td>
<td>156</td>
</tr>
<tr>
<td>Use of the REPLACEWORKID keyword</td>
<td>157</td>
</tr>
<tr>
<td>Establishing a baseline</td>
<td>170</td>
</tr>
<tr>
<td>Workflow for the baseline process</td>
<td>172</td>
</tr>
<tr>
<td>Scope rules from a baseline profile</td>
<td>179</td>
</tr>
<tr>
<td>DDL file</td>
<td>182</td>
</tr>
<tr>
<td>ALUIN input stream for a baseline</td>
<td>184</td>
</tr>
<tr>
<td>Browse Baseline panel</td>
<td>188</td>
</tr>
<tr>
<td>ALUIN input stream for a baseline report</td>
<td>190</td>
</tr>
<tr>
<td>Execution Pre- and Post-Processing Interface panel</td>
<td>206</td>
</tr>
<tr>
<td>Baseline Deletion List panel</td>
<td>215</td>
</tr>
<tr>
<td>Workflow for the comparison process</td>
<td>222</td>
</tr>
<tr>
<td>Comparing a DB2 catalog to a DB2 catalog</td>
<td>224</td>
</tr>
<tr>
<td>Comparing a DB2 catalog to a DDL file or a migrate-type worklist</td>
<td>227</td>
</tr>
<tr>
<td>Comparing a DDL file to a DDL file</td>
<td>229</td>
</tr>
<tr>
<td>Comparing a baseline to a baseline</td>
<td>231</td>
</tr>
<tr>
<td>Comparing a baseline to a DB2 catalog</td>
<td>232</td>
</tr>
<tr>
<td>Comparing a DB2 catalog to a catalog baseline</td>
<td>233</td>
</tr>
<tr>
<td>Comparing a baseline to a DDL file or migrate-type worklist</td>
<td>235</td>
</tr>
<tr>
<td>Keywords used in a DDL file to a DB2 catalog comparison</td>
<td>239</td>
</tr>
<tr>
<td>Keywords used for a DB2 catalog to a DB2 catalog comparison</td>
<td>240</td>
</tr>
<tr>
<td>Keywords used for a migrate-type worklist to a DB2 catalog comparison</td>
<td>240</td>
</tr>
<tr>
<td>Keywords used for a baseline to a baseline comparison</td>
<td>242</td>
</tr>
<tr>
<td>Compare Catalog to Catalog panel</td>
<td>244</td>
</tr>
<tr>
<td>Keywords used in ALUIN input stream for catalog-to-catalog comparison</td>
<td>245</td>
</tr>
<tr>
<td>Comparing two DB2 catalogs</td>
<td>246</td>
</tr>
<tr>
<td>Compare Object Attributes panel</td>
<td>251</td>
</tr>
<tr>
<td>Use of ALTER, CREATE, and DROP in the ALUIN input stream</td>
<td>255</td>
</tr>
<tr>
<td>Use of CDLDEP in the ALUIN input stream</td>
<td>256</td>
</tr>
<tr>
<td>Example ALUIN input stream</td>
<td>260</td>
</tr>
<tr>
<td>Compare Catalog to Catalog panel</td>
<td>267</td>
</tr>
<tr>
<td>ALUIN input stream—DB2 catalog to DB2 catalog</td>
<td>271</td>
</tr>
<tr>
<td>ALUIN input stream—DB2 catalog to DB2 catalog with profiles</td>
<td>275</td>
</tr>
<tr>
<td>ALUIN input stream—DB2 catalog to a DDL file</td>
<td>279</td>
</tr>
<tr>
<td>ALUIN input stream—DDL file to a DDL file</td>
<td>282</td>
</tr>
<tr>
<td>ALUIN input stream—baseline to a baseline</td>
<td>286</td>
</tr>
<tr>
<td>ALUIN input stream—baseline to a DB2 catalog</td>
<td>289</td>
</tr>
<tr>
<td>ALUIN input stream—DB2 catalog to a baseline</td>
<td>293</td>
</tr>
<tr>
<td>ALUIN input stream—baseline to a DDL file</td>
<td>296</td>
</tr>
<tr>
<td>ALUIN input stream—convert work ID to CDL</td>
<td>299</td>
</tr>
<tr>
<td>Workflow for the alter process (part 1 of 2)</td>
<td>312</td>
</tr>
<tr>
<td>Mixed List panel</td>
<td>315</td>
</tr>
<tr>
<td>Auxiliary Objects List panel</td>
<td>318</td>
</tr>
<tr>
<td>Tablespace Estimation panel (nonpartitioned)</td>
<td>338</td>
</tr>
<tr>
<td>Index Space Estimation panel (partitioned)</td>
<td>341</td>
</tr>
<tr>
<td>Tablespace Estimation panel</td>
<td>345</td>
</tr>
<tr>
<td>Execution Pre- and Post-Processing Interface panel</td>
<td>350</td>
</tr>
<tr>
<td>Example of name template</td>
<td>351</td>
</tr>
<tr>
<td>DML for replicating a work ID</td>
<td>363</td>
</tr>
</tbody>
</table>
DML for replicating a database .................................................. 364
DML for applying changes from a user-defined table ................. 367
DML for changing column definitions ........................................ 371
DML for deleting work IDs ...................................................... 375
DML for deleting sync table entries .......................................... 377
Mixed List panel ................................................................. 385
Constraint Key Mixed List panel .............................................. 386
New constraint key column ................................................... 388
Mixed List panel ................................................................. 389
Constraint Key Mixed List panel .............................................. 390
Edited constraint key columns .............................................. 391
Constraint Key Mixed List panel .............................................. 393
Constraint Key Mixed List panel .............................................. 394
Mixed List panel ................................................................. 396
Index Key Mixed List panel ................................................... 398
Index Key Mixed List panel—column added ............................. 399
DDL to create a view ............................................................ 401
Table Quick Edit Confirmation dialog ..................................... 407
DML for updating database .................................................... 417
SQL to create temporary table ............................................... 420
DML to add columns to the PLAN_TABLE ............................... 421
DML for deleting aliases ....................................................... 423
DML for copying a column .................................................... 424
DML for updating index parts ............................................... 425
DML to add WITH RESTRICT ON DROP ................................ 426
DML to locate indexes with existing data sets ........................... 427
DML to create synonyms ....................................................... 428
DML to convert VCAT-defined partitions to STOGROUP-defined partitions ... 430
LIKE the table space with an existing history table .................... 440
LIKE the existing history table .............................................. 441
Edit the base table ............................................................ 442
LIKE the table space with an existing history table .................... 443
LIKE the existing history table .............................................. 444
Edit the base table ............................................................ 445
DML to update an option ...................................................... 452
DML to update a parameter .................................................. 453
Tablespace Estimation panel (nonpartitioned) .......................... 454
Average Row Length panel .................................................... 455
Tablespace Estimation panel (partitioned) ............................... 456
Space Estimation Attribute Propagation panel ......................... 457
Tablespace Estimation panel (partition-by-growth table space) .... 458
Mixed List panel ................................................................. 459
Index Space Estimation panel (partitioned) .............................. 461
Space Estimation Attribute Propagation panel ......................... 462
Mixed List panel ................................................................. 463
DML to update a table .......................................................... 468
DML to update a table partition .............................................. 468
DML to update an index partition .......................................... 468
DML to update a table by using the JOIN command ................. 469

Figures 17
DML to update a table partition by using the JOIN command ................................. 469
DML to update an index partition by using the JOIN command ............................. 469
DML to estimate the primary and secondary quantities ...................................... 471
Example DML for setting SQTY ................................................................. 472
DML statements to create and modify the target set of objects .......................... 474
DML statements to modify and migrate objects ................................................. 477
DML to convert to partition-by-growth table spaces ......................................... 479
Index Key Mixed List panel .............................................................................. 483
Index Key Mixed List panel .............................................................................. 484
Partition-by-growth dialog .............................................................................. 492
DML to change table-controlled partitioned table spaces to range-partitioned table spaces ................................................................. 495
DML to change table-controlled partitioned table spaces to range-partitioned table space by using a JOIN ................................................................. 495
DML to change index-controlled partitioned table spaces to range-partitioned table spaces ................................................................. 496
DML to convert to range-partitioned table spaces ............................................. 499
Preserve Limitkey Values panel ....................................................................... 501
Table Detail panel ........................................................................................... 502
DML to convert to table-controlled partitioned table spaces .............................. 504
DML to convert to range-partitioned table spaces ............................................. 506
Partition-by-growth dialog .............................................................................. 508
DML to change a table space to a partition-by-growth table space ...................... 514
DML to change tables from an explicit database to an implicit database .......... 516
DML to change tables from an explicit database to an implicit database .......... 516
DML to convert to implicit databases and table spaces ..................................... 517
DML to change the encoding scheme for a table space .................................... 518
Index Key Mixed List panel .............................................................................. 523
Index Key Mixed List panel .............................................................................. 524
Auxiliary Objects List—partitioned table space .............................................. 530
Auxiliary Objects List—single partition ........................................................... 535
Auxiliary Objects List—nonpartitioned table space ......................................... 542
DML for creating a database and table spaces ................................................. 548
DML to change multi-table table spaces to single-table table spaces............... 551
DML to move empty tables in a database ......................................................... 553
DML to update table spaces and indexes with DEFINE NO .............................. 555
DML to update table spaces and indexes with DEFINE NO based on cardinality .......................................................................................... 556
DML to locate table spaces defined with DEFINE NO ....................................... 557
DML to set the length of an inline CLOB to the system default ......................... 558
DML to change an inline CLOB column to a non-inline CLOB column ............. 558
DML to change non-inline LOB columns to inline LOB columns ..................... 559
DML to change the length of LOB columns in universal table spaces ............... 559
DML to set the length of the CLOB column based on the maximum length ...... 559
Creating copies of an environment ................................................................ 614
Optimizing the design of the environment ....................................................... 619
Creating additional job steps in the Execution interface ................................. 636
TASKID Execution Status panel ..................................................................... 639
AEXIN input stream for restarting a worklist .................................................... 644

18 ALTER and CHANGE MANAGER for DB2 User Guide, Volume 2
<table>
<thead>
<tr>
<th>Figures</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEXIN input stream for starting over a worklist</td>
<td>645</td>
</tr>
<tr>
<td>Script steps</td>
<td>646</td>
</tr>
<tr>
<td>Script Selection List panel</td>
<td>647</td>
</tr>
<tr>
<td>Baseline Step Options panel</td>
<td>650</td>
</tr>
<tr>
<td>Step Baseline Profile Options panel</td>
<td>651</td>
</tr>
<tr>
<td>Compare Step Options panel</td>
<td>652</td>
</tr>
<tr>
<td>Step Migrate Profile Options panel</td>
<td>653</td>
</tr>
<tr>
<td>Import Step Options panel</td>
<td>654</td>
</tr>
<tr>
<td>Step Migrate Profile Options panel</td>
<td>655</td>
</tr>
<tr>
<td>Analysis Step Options Panel</td>
<td>656</td>
</tr>
<tr>
<td>Execution Step Options panel</td>
<td>657</td>
</tr>
<tr>
<td>Baseline Report Step Options panel</td>
<td>658</td>
</tr>
<tr>
<td>Replicate WORKID Step Options panel</td>
<td>659</td>
</tr>
<tr>
<td>Application Action Menu</td>
<td>667</td>
</tr>
<tr>
<td>Analysis Utility Options panel</td>
<td>675</td>
</tr>
<tr>
<td>Analysis LOB DATA MOVER Commit Options panel</td>
<td>680</td>
</tr>
<tr>
<td>Analysis Utility Dataset Options panel</td>
<td>682</td>
</tr>
<tr>
<td>Analysis Unload / Copy Options panel—UNLOAD PLUS</td>
<td>684</td>
</tr>
<tr>
<td>Analysis Unload / Copy Options panel—RECOVER PLUS</td>
<td>690</td>
</tr>
<tr>
<td>Analysis Unload / Copy Options panel—Copy Migration</td>
<td>693</td>
</tr>
<tr>
<td>Analysis Reorg Options panel</td>
<td>700</td>
</tr>
<tr>
<td>Analysis Image Copy Options panel</td>
<td>707</td>
</tr>
<tr>
<td>Analysis Statistics Options panel</td>
<td>715</td>
</tr>
<tr>
<td>Processing flow of the SLIB compiler</td>
<td>720</td>
</tr>
<tr>
<td>Sample runtime report</td>
<td>724</td>
</tr>
<tr>
<td>Parallel worklist processing</td>
<td>729</td>
</tr>
<tr>
<td>Result of STATUS command</td>
<td>734</td>
</tr>
<tr>
<td>Result of QUIESCE command</td>
<td>736</td>
</tr>
<tr>
<td>XIMACMI member</td>
<td>738</td>
</tr>
</tbody>
</table>
Tables

Mixed List actions ................................................................. 44
Example work ID change rule ............................................... 47
<BLANK> support for attributes ............................................. 52
Ordering of change rules ..................................................... 55
Change rules for storage group and VCAT partition volume lists ........................................ 56
Execution return codes ....................................................... 98
Execution status fields ........................................................ 100
Sync table columns ............................................................ 101
Conditions for writing sync records ...................................... 102
Output files for worklist parallelism ...................................... 103
Reestablished sync values .................................................... 107
Restarting utilities ............................................................ 109
Migration tasks ................................................................. 113
Worklist commands ........................................................... 182
Load requirements for full recovery ...................................... 185
Deleting baselines using DELETEAGE .................................... 195
Deleting baselines using RETAINMAX ................................... 195
DB2 catalog table indexes .................................................... 197
Baseline tasks ................................................................. 198
Types of comparisons ......................................................... 235
Scope of a comparison ........................................................ 236
DD Names used in the comparison JCL .................................. 258
Compare return codes ......................................................... 262
DB2 catalog table indexes .................................................... 264
Comparison tasks ............................................................. 266
Mixed List actions ............................................................. 315
Change-level indicators ....................................................... 316
Supported types of conversions ............................................ 321
Converting to table-controlled partitioning ......................... 322
Referencing physical and logical partition numbers ............... 324
Specifying <DEFLT> for the database or table space name ........ 325
Working with implicit table spaces ....................................... 326
Import input rules for CDL and DDL files ............................. 334
Import return codes ......................................................... 335
Attributes for estimating the table space ............................... 338
Attributes for modifying the table space ............................... 339
Attributes for estimating index space ................................... 342
Attributes for modifying the index space ............................... 343
Name template characters .................................................... 351
Comparing a baseline to a catalog ........................................ 351
About this book

The ALTER and CHANGE MANAGER for DB2 User Guide, Volume 2 explains all of the components, objects, tasks, and functions of the ALTER and CHANGE MANAGER products. This book also provides examples of how to use the products.

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

- IBM® DB2 Universal Database for z/OS® and OS/390®
- job control language (JCL)
- Interactive System Productivity Facility (ISPF)

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

**NOTE**

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

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

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:

- Link to the BMC Documentation Center (https://webapps.bmc.com/infocenter/index.jsp) to browse documentation sets

- View BMC Quick Course Demos (short overviews of selected product concepts, tasks, or features), which are included in the BMC Documentation Center
Read individual product documents (books and notices) within the “A – Z Supported Product List”

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

### Conventions

This book uses the following special conventions:

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

- **Variable text in path names, system messages, or syntax is displayed in *italic* text:**

  - `testsy/instance/filename`

- **The symbol => connects items in a menu sequence. For example, Actions => Create Test instructs you to choose the Create Test command from the Actions menu.**

- **Change bars signify changes that clarify or correct existing information, or that provide new information corresponding to product changes. This book does not use change bars to denote editorial and formatting changes or typographical errors that have been fixed, unless these updates significantly affect your use of the information.**

The following symbols represent components and data structures in processes:

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="example.png" alt="globe" /> <img src="example.png" alt="folder" /></td>
<td>represents a collection of tables, such as the DB2 catalog, or internal storage. The symbol is also used to represent baselines.</td>
</tr>
<tr>
<td><img src="example.png" alt="disk" /></td>
<td>represents files that usually reside on DASD, although they might be allocated on a tape unit.</td>
</tr>
</tbody>
</table>
The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2 | KEYWORD3] KEYWORD4={YES | NO} fileName...
```

The following table explains conventions for syntax statements and provides examples:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items in italic type represent variables that you must replace with a name or value. If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words.</td>
<td>alias&lt;br&gt;databaseDirectory&lt;br&gt;serverHostName</td>
</tr>
</tbody>
</table>
| Brackets indicate a group of optional items. Do not type the brackets when you enter the option. A comma means that you can choose one or more of the listed options. You must use a comma to separate the options if you choose more than one option. | [tableName, columnName, field]<br>[-full, -incremental, -level] (Unix)
The following figure shows the standard format for syntax diagrams:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item.</td>
<td>`{DBDName</td>
</tr>
<tr>
<td>A vertical bar means that you can choose only one of the listed items. In the example, you would choose either commit or cancel.</td>
<td>{commit</td>
</tr>
<tr>
<td>An ellipsis indicates that you can repeat the previous item or items as many times as necessary.</td>
<td>columnName . . .</td>
</tr>
</tbody>
</table>

**Syntax diagrams**

The following figure shows the standard format for syntax diagrams:
The following guidelines provide additional information about syntax diagrams:

- Read diagrams from left to right and from top to bottom, along the primary path line.

- Required items are on the primary path line.

- Optional items are below the primary path line.

- An underlined item is a default value.

- If you must select an item from two or more required items, the first item in the stack is shown on the primary path line.

- If you can choose from two or more optional items, they are vertically stacked and the entire stack is below the primary path line.

- A recursive (left-pointing) arrow above or below a single item or a stack of items indicates that you might repeat the single item or choose more than one item in the stack. If the items must be separated by commas, the recursive arrow contains a comma.

- Usually, you can select each possible path only once. However, in some cases, a path might be repeated (usually with varying parameters). Statements that allow paths to be repeated are noted in the description of the specific statement, and the repeatable items are flagged with a dagger (†) character.

- Sub-diagrams are indicated by variable names in boldface. When a portion of a syntax diagram cannot fit onto a single page, or within the boundary of a column, a variable name represents a sub-diagram that is defined separately from the main diagram.

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

- In general, commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum portion of the command or keyword might appear in uppercase with the remainder of the word in lowercase (for example, CANcel).
The following conventions apply to variables:

— Variables typically appear in lowercase and are always italicized.

— If a variable is represented by two or more words, underscores connect the words (for example, `databaseName` and `userID`).

— If a variable name occurs more than once, subscripted numbers are used to distinguish one occurrence from another when references would otherwise be ambiguous.

The use of `<DEFLT>` indicates that the system-defined default for that value should be used.

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 either 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.
Managing objects with ALTER and CHANGE MANAGER

This part presents the following topics:

Chapter 1
Migrating database environments ........................................ 31

Chapter 2
Taking a snapshot of a database environment ............................ 169

Chapter 3
Comparing database environments ....................................... 219

Chapter 4
Altering data structures in a database environment .................... 307
This chapter presents the following topics:

Before you begin .................................................. 32
Overview .......................................................... 33
Migrating data structures ....................................... 33
Deciding how to migrate data structures .................. 37
  Using migrate-type work IDs ............................... 37
  Using outbound migrate profiles ......................... 38
  Using Data Manipulation Language .................... 39
Defining the scope of a migration .......................... 40
  Defining the scope with a profile ....................... 40
  Specifying the data structures for a migration ....... 41
  Identifying changes to data structures ................. 45
Specifying change rules for a migration .................. 46
  Specifying change rules for a migrate-type work ID 47
  Specifying change rules for an outbound migrate profile 48
  Specifying values for change rule attributes .......... 49
  Resolving change rules ................................. 53
Specifying migrate options .................................. 57
Analyzing the effects of a migration ....................... 58
  Analyzing requests using work IDs .................... 59
  Analyzing requests using migrate profiles ............ 60
  Selecting the run type ................................. 60
  Overriding the default processing options .......... 61
  Validity and performance checks ..................... 73
  Reviewing the results of Analysis .................... 78
Executing changes ............................................. 82
  Specifying options for Execution .................... 82
  Controlling authorizations ............................. 94
  Processing a worklist .................................. 98
  Reviewing the results of a migration ................. 103
  Restart methods .................................... 106
Accomplishing your goals .................................... 113
  Creating a migrate-type work ID .................... 114
  Creating a migrate-type work ID in batch mode .... 115
  Defining migrate options for a migrate-type work ID 116
Before you begin

Before you migrate (or copy) your data structures, you should review the chapters in the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*. 

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating change rules for a migrate-type work ID</td>
<td>117</td>
</tr>
<tr>
<td>Converting an external SQL procedure to a native stored procedure</td>
<td>118</td>
</tr>
<tr>
<td>Specifying the data structures in a migrate-type work ID.</td>
<td>119</td>
</tr>
<tr>
<td>Migrating the auxiliary objects for a LOB column</td>
<td>121</td>
</tr>
<tr>
<td>Migrating data structures by using DML</td>
<td>123</td>
</tr>
<tr>
<td>Assigning a unique name by importing DML</td>
<td>128</td>
</tr>
<tr>
<td>Defining primary and secondary quantities by importing DML</td>
<td>129</td>
</tr>
<tr>
<td>Setting a secondary quantity to a factor of a primary quantity by importing DML</td>
<td>131</td>
</tr>
<tr>
<td>Changing table spaces and indexes to DEFINE NO by importing DML</td>
<td>132</td>
</tr>
<tr>
<td>Migrating a partitioned table space to a range-partitioned table space by importing DML</td>
<td>134</td>
</tr>
<tr>
<td>Migrating a single-table table space to a partition-by-growth table space by importing DML</td>
<td>136</td>
</tr>
<tr>
<td>Migrating tables from an explicit database to an implicit database by importing DML</td>
<td>137</td>
</tr>
<tr>
<td>Migrating tables that contain inline LOB columns by importing DML</td>
<td>138</td>
</tr>
<tr>
<td>Migrating stored procedures by importing DML</td>
<td>140</td>
</tr>
<tr>
<td>Migrating data structures by using a migrate-type work ID.</td>
<td>141</td>
</tr>
<tr>
<td>Creating a stop list</td>
<td>144</td>
</tr>
<tr>
<td>Creating an outbound migrate profile</td>
<td>146</td>
</tr>
<tr>
<td>Creating scope rules for an outbound migrate profile</td>
<td>148</td>
</tr>
<tr>
<td>Specifying locations in an outbound migrate profile</td>
<td>149</td>
</tr>
<tr>
<td>Creating change rules for a migrate profile</td>
<td>151</td>
</tr>
<tr>
<td>Analyzing a migrate-type work ID and generating a worklist</td>
<td>153</td>
</tr>
<tr>
<td>Creating a receive-type work ID</td>
<td>155</td>
</tr>
<tr>
<td>Executing a migrate-type worklist</td>
<td>158</td>
</tr>
<tr>
<td>Restarting a worklist</td>
<td>164</td>
</tr>
<tr>
<td>Starting a worklist over</td>
<td>166</td>
</tr>
<tr>
<td>Where to go from here</td>
<td>168</td>
</tr>
</tbody>
</table>
Overview

The ALTER and CHANGE MANAGER products enable you to create data structures on a DB2 subsystem from structures that already exist on the same subsystem or on a different subsystem. This process of creating data structures is called migration. In the migration process, you create a new set of data structures, using the existing data structures as a template. The migration process enables you to:

- copy an application’s data structures from a development subsystem to a test subsystem
- copy an initial version (or perhaps a major update) of an application’s data structures to one or more production subsystems
- migrate an entirely new set of structures, because the new version of an application’s data structures is substantially different from prior versions
- migrate data with the data structures
- use migration as the preferred method of installation when you install the initial version of an application

This chapter describes the migration process and the tasks that you can perform to accomplish the following goals:

- migrate data structures, data, or both within the same subsystem
- migrate data structures, data, or both to a different subsystem

Migrating data structures

The process of migrating data structures and data within the same subsystem or to a different subsystem is virtually the same. The migration process consists of the following tasks:

- creating a migrate-type work ID, an outbound migrate profile, or both to define the scope of the migration, migration options, and the change rules for the migration
- specifying the data structures and their dependents
- analyzing the requests in the work ID
- generating a worklist
Migrating data structures

- generating the execution job control language (JCL)
- executing the worklist

When the worklist is executed, the product unloads the data, creates the new data structures, loads the data, and runs any other utilities. When you migrate data and data structures to a different subsystem, the worklist is executed in two phases:

- Phase 1 executes on the sending subsystem and unloads the data.
- Phase 2 executes on the receiving subsystem and creates the new data structures, loads any migrated data, and runs any utilities.

The workflow in Figure 1 illustrates the migration process.

NOTE
For more information, view the Quick Course Migrating in Same Subsystem. You must have a BMC Support ID to view the Quick Course.
Figure 1  Workflow for the migration process (part 1 of 2)

Start

Create a migrate-type Work ID

Do you want to reuse all information that you specify for a migration? yes

Define the scope, change rules, and locations in an outbound migrate profile

no

Define the migrate options in a migrate-type Work ID

Do you want to use change rules? yes

Define change rules in an outbound migrate profile

no

Define the scope of the migration by specifying the data structures

Do you want to reuse the change rules? yes

Define change rules in a migrate-type Work ID

no

Analyze the Work ID and generate a worklist
Figure 1  Workflow for the migration process (part 2 of 2)

Are you generating a worklist to use in a comparison?

- yes: Go to comparison process
- no:

Are you migrating only data structures to a different subsystem?

- yes: Generate execution JCL
- no: Execute the worklist

Generate execution JCL
(See the documentation for additional steps for storing data sets.)

Execute the worklist
(If you are migrating to the same subsystem, a single-phase worklist is used to migrate data structures and data or data structures only.)

Did you execute the worklist for a receive-type Work ID?

- yes: Do the subsystems share DASD?
- no: Are you migrating data and data structures to a different subsystem?

- yes: Create a receive-type Work ID on the receiving subsystem
- no:

Stop

FTP the worklist, unloaded data, or both to the receiving subsystem
Deciding how to migrate data structures

To perform a migration, you must first create a migrate-type work ID. The migrate-type work ID is the basic unit of work in the migration process. After you create the migrate-type work ID, you can choose to use options and rules in the work ID, an outbound migrate profile, an SQL-like data manipulation language (DML), or a combination of all three to migrate your data structures, data, or both.

- The migrate-type work ID enables you to group your migration requests into one work unit.
- The outbound migrate profile, which can only be used in CHANGE MANAGER, enables you to select a set of objects and customize the changes to the objects. The outbound migrate profile can be defined one time and then used repeatedly for different processes.
- The DML that is used in the CM/PILOT component of CHANGE MANAGER enables you to migrate multiple data structures in a single statement.

Using migrate-type work IDs

Migrate-type work IDs can contain migrate options, change rules, and specifications. The migrate options define the default dependent objects that should be included in the migration. The change rules allow you to have the product automatically change the value of an attribute of an existing data structure during the migration process. Specifications are used to select parent objects, as well as dependent objects, to be migrated.

Migrate-type work IDs that contain migrate options and change rules are typically used for single-use or one-time migrations. That is, the information that is contained in a migrate-type work ID cannot be reused.

**TIP**

To create a migrate-type work ID, see page 114.

**NOTE**

For more information, view the Quick Course Creating Work IDs. You must have a BMC Support ID to view the Quick Course.
Using outbound migrate profiles

Instead of using a migrate-type work ID to select the objects and define the change rules, you can use an outbound migrate profile in CHANGE MANAGER to specify the scope of the migration and the change rules. You can also specify locations in an outbound migrate profile to generate multiple worklists. Each specified location can have change rules that tailor its data structures. You can then reuse the profile in future migrations with different migrate-type work IDs.

**TIP**
To create an outbound migrate profile, see page 146.

If you regularly migrate data structures (for example, some installations move data structures from development into test once per week), an outbound migrate profile’s scope automates the object selection process. The outbound migrate profile’s scope can automatically select the objects to be migrated, thus eliminating the task of manually selecting them with Specification. In addition, you need to select the objects to migrate only once. You can use the same migrate profile scope when you establish a baseline that contains the application’s data structures. Doing so ensures consistency between the migrated objects and the objects in the baseline by avoiding the inadvertent omission of objects from either the baseline or the migration. (For more information about establishing baselines, see Chapter 2, “Taking a snapshot of a database environment.”) You can also use an outbound migrate profile to migrate only data by defining the scope rules to select a specific database and to include dependent objects and data. You can also exclude SQL.

**NOTE**
If you use an outbound migrate profile’s scope, any migrations that you request through Specification, as well as those that are stored in the migrate-type work ID, are ignored. You cannot use the scope from both the work ID and the profile.

**TIP**
To create scope rules for an outbound migrate profile, see page 148.

Change rules in the outbound migrate profile can automate object attribute changes for existing objects. If multiple locations are specified, each one can have change rules that tailor the worklist to the specific requirements for each location. Change rules are commonly used to change the owner of a table or the name of the database.
Locations can specify different sets of change rules for multiple receiving locations. For example, at many installations, a central development center must support several production sites. In these cases, you must migrate the application's data structures to several receiving locations. If you support multiple locations, you can use the locations defined for an outbound migrate profile to generate a worklist for each one. You can also use the same worklist for multiple receiving locations. For example, if your production systems are identical, you can simply copy the worklist to each subsystem, or even access the same worklist through shared DASD.

**TIP**
To create change rules for a migrate profile, see page 151.

**TIP**
To specify locations in an outbound migrate profile, see page 149.

**NOTE**
For more information, view the Quick Course Creating Migrate Profiles. You must have a BMC Support ID to view the Quick Course.

### Using Data Manipulation Language

The CM/PILOT component of CHANGE MANAGER provides several scripts that allow you to use an SQL-like DML to specify data structure changes or migrations. This function is similar to the function of the Specification component of CHANGE MANAGER, except that you can quickly change or migrate multiple objects at the same time. The CM/PILOT panels for these scripts help you to create DML statements by providing input fields for each action (update, delete, migrate, and replicate) and each data structure type.

A DML statement can contain keywords, a WHERE clause, and a SET clause.

- The INCLUDE keyword specifies the migrate options or the types of dependent data structures that you want to migrate.
- A WHERE clause defines the data structures that you want to change, or the scope.
- A SET clause specifies changes to the attributes of the data structures or to the data structures.
Defining the scope of a migration

DML uses syntax that is slightly different than DB2 DML. For information about the DML used by CM/PILOT, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

Error checking on the DML is performed when you process a CM/PILOT worklist. The product checks the following items:

- data structure types
- actions (update, delete, migrate)
- keywords
- SET clauses
- WHERE clauses

Any warnings or errors are described in the DML section of the DML report. The DML report is included as part of the diagnostic output. It contains information about the data structures that are affected by the DML statements. You can review the report and evaluate the changes before executing the changes.

---

**TIP**

To migrate data structures by using DML, see page 123.

---

Defining the scope of a migration

You can define the scope of a migration either by using an outbound migrate profile or by explicitly specifying data structures.

Defining the scope with a profile

In an outbound migrate profile, you use scope rules and migrate options to define the set of data structures and the dependencies of those data structures to include in the migration.

Use the following guidelines when you create scope rules:

- In CHANGE MANAGER, stored procedures are not dependent upon any other DB2 object. Thus, when you create a scope rule for an object that is modified by a stored procedure or that calls a stored procedure, you cannot include the stored procedure as a dependency. To include stored procedures in the scope, you must specify them explicitly.
Auxiliary objects for a LOB column are dependent upon databases, table spaces, and tables.

— If you select a database or a table space object, you must include the dependent tables or auxiliary objects.

— If you select a table object, you must include the dependent auxiliary objects.

To migrate auxiliary objects for a LOB column, migrate the auxiliary objects and the base table.

Authorizations are the only dependency on sequence objects.

To include the parent objects of tables in the scope, include the dependent database and table space. If the parent object is an implicit object, CHANGE MANAGER ignores the object.

Specifying the data structures for a migration

If you do not define the scope of a migration by using an outbound migrate profile, you can define the scope in a work ID by specifying the names of the data structures that you want to appear in a mixed list.

From the mixed list, you can specify the objects that you want to migrate for a migrate-type work ID. Figure 2 shows an example of a mixed list. The list displays different types of DB2 data structures.

The objects in a mixed list are displayed in hierarchical order by object type. For example, table spaces that are associated with a database appear under the database, as do the tables that are associated with a table space. The products retrieve data that can use any encoding scheme from the DB2 catalog. When the products execute SQL that uses an ORDER BY clause against the DB2 catalog, the query uses a Unicode collating sequence to sort data and typically uses the same sequence to display the data on panels and in reports. However, when the products sort the data retrieved from the DB2 catalog queries, the products display sorted data on panels and in reports in an EBCDIC collating sequence.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key. You can specify user options for displaying the name, truncation characters, and autotab characters. To set the panel attributes, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*. 
Figure 2  List of DB2 data structures

For object-type abbreviations that Specification uses in the Mixed List panel and for the data types for the object names, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

In CHANGE MANAGER, stored procedures and sequences are not dependent upon any other DB2 object and are displayed in the Unattached Objects section at the end of the mixed list (as shown in Figure 3).
As a result, if you want to migrate stored procedures that read or modify another object or that are called by another object, you must explicitly specify the stored procedures for the migration. Orphaned auxiliary indexes, which are auxiliary indexes that are not associated with an existing auxiliary table, are also displayed in the Unattached Objects section. (Orphaned auxiliary table spaces, however, are displayed in the hierarchy.)

You can use the FIND and ZOOM commands in the mixed list to find specific data structures and the dependents for those data structures.

- You can use the FIND command to quickly locate a specific data structure. Type \texttt{FI} on the \textit{Command} line and specify a character string on which to search. For example, you can type \texttt{FI EMP} and press Enter. The cursor displays at the first occurrence of a match.

- You can use the ZOOM command in the Mixed List panel to list all of the dependents for an object in a hierarchical fashion. Type \texttt{Z} in the \textit{Act} column adjacent to the object that you want to zoom. ZOOM displays all of the dependent objects, which includes dependent materialized query tables (MQTs).

\textbf{NOTE}

This command can take several minutes to process if the object contains dependent views.
You can use action codes (or **Command** line commands) on the mixed list to indicate the type of action that you want to perform. Some of the actions that you can perform are described in **Table 1**.

**Table 1  Mixed List actions**

<table>
<thead>
<tr>
<th>Act</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Migrate</td>
<td><em>(migrate-type work ID only)</em> specifies the creation of data structures on the destination subsystem that are identical to those on the origin subsystem. The work ID migrate options control which dependent objects are included in the migration. You cannot use the M action code on an orphaned auxiliary object.</td>
</tr>
<tr>
<td>MO</td>
<td>Migrate Options</td>
<td><em>(migrate-type work ID only)</em> for the selected object, overrides the default dependencies specified for the migrate-type work ID through the work ID migrate options. You can use the MO action code to mark an object for migration and to specify dependencies.</td>
</tr>
<tr>
<td>QE</td>
<td>Quick Edit</td>
<td>marks the selected tables for moving to a different database, table space, or both</td>
</tr>
<tr>
<td>USE</td>
<td>Use</td>
<td>marks the selected table space as the table space into which tables flagged QE are moved</td>
</tr>
<tr>
<td>U</td>
<td>Undo</td>
<td>reverses the action</td>
</tr>
</tbody>
</table>

You can also use the following command-line commands on a mixed list:

- *(migrate-type work ID only)* MIGALL marks all of the data structures, except orphaned auxiliary objects, for migration.

- DROPALL excludes all of the data structures that are marked for migration from being included in the migration. This command is the equivalent of typing D in the Act column for all data structures.

- UNDOALL reverses the action on all data structures. This command is the equivalent of typing U in the Act column for all data structures.

When you indicate the type of action that you want to perform, consider the following information:

- You can migrate auxiliary objects for a LOB column by migrating the base table that contains the LOB column and including auxiliary objects as dependents. You cannot edit, like, or drop auxiliary objects for a LOB column in a mixed list.
To exclude a dependent object or an auxiliary table or index from a migration, type **D** in the Act column adjacent to the object. For example, if you want to migrate a database and all of its dependent objects except for one table space, type **D** in the Act column adjacent to the table space. The table space is excluded from the migration.

You can create, like, edit, and drop external stored procedures and native SQL stored procedures.

**NOTE**

You can only migrate the following types of stored procedures:

- external SQL stored procedures
- stored procedures that contain a UDT data type or an SQL table function that defines a parameter for a transition table (TABLE LIKE ... AS LOCATOR syntax)
- all stored procedures on a DB2 subsystem that is in one of the following modes: version 8 NFM, version 9 CM or ENFM, or version 10 CM8 or ENFM8

BMC displays these types of procedures in the Mixed List.

For more information about the commands that you can use in a mixed list, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**TIP**

To specify the data structures in a migrate-type work ID, see page 119.

---

**Identifying changes to data structures**

When you specify a change to an object, an asterisk (*) and a change-level indicator appears on a list panel adjacent to the Act column. The M change-level indicator signifies that you selected to migrate an object. This indicator is not applicable to auxiliary objects.
When you request a change to a data structure, only the definition of the data structure and the requested changes are saved in the Change Definition (CD) tables. The specified changes do not occur in the DB2 catalog until you analyze the changes, generate a worklist, and execute the worklist.

### Specifying change rules for a migration

You can use change rules to modify the attributes of the data structures. Change rules compare the object type, attribute, and current value of the attribute, along with any name qualifiers, with those in the migration. If a match is found, the new value specified for the attribute is used for the object on the destination subsystem.

The following types of change rules exist:

- **Change-type** is used to change the value of a data structure’s attribute. For example, you can change the buffer pool of a table space from BP1 to BP3. You can also change the names of objects to enforce the naming conventions of the receiving location.

- **Include-type** is used to include volumes in a storage group, table space, or index. This type of change rule enables you to tailor a set of data structures for the DASD requirements of the receiving location.

- **Exclude-type** is used to exclude volumes from a storage group, table space, or index.

- (migration only) **Force-type** is used to force storage group defined table spaces and indexes to be created by using VCAT catalog names on the receiving subsystem instead of storage group names. You can only use this rule with storage-group defined table spaces or indexes and you must specify the VCAT attribute.

- (import and comparison only) **Suppress-type** is used in an inbound migrate profile to filter out the changes specified in a Change Definition Language (CDL) file when you use the Import component to import the CDL. You can also filter out specific changes from appearing in a CDL file by using suppress-type change rules in an outbound migrate profile with the Compare component.

You can specify change rules for either migrate-type work IDs or outbound migrate profiles.
Specifying change rules for a migrate-type work ID

In a migrate-type work ID, you can specify change rules to automate changes to the attributes of existing data structures and to make the data structures on the receiving subsystem different from those on the sending subsystem.

Figure 4 shows an example of a work ID change rule. The object is a database and the attribute specified to change is the name of the database. The current value is TEST* and the new value is PROD* (for more information about wildcard characters, see “Specifying values for change rule attributes” on page 49).

Figure 4  Work ID change rule

When this change rule is applied, all databases whose names match the pattern TEST* in the migration will be changed to have the prefix PROD on the destination subsystem, as shown in Table 2.

Table 2  Example work ID change rule

<table>
<thead>
<tr>
<th>Current Attribute Value</th>
<th>New Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST001</td>
<td>PROD001</td>
</tr>
<tr>
<td>TESTDB3</td>
<td>PRODDB3</td>
</tr>
<tr>
<td>TES01</td>
<td>Unchanged</td>
</tr>
</tbody>
</table>
Specifying change rules for an outbound migrate profile

You can streamline and automate changes to sets of object structures by defining change rules for migrate profiles. The change rules are applied to the existing objects that are defined for the scope of the profile or that are defined in Specification.

Change rules allow common changes to be easily specified and repeated. For example, in many DB2 installations, the owner or creator of a table changes when it is moved from a test system into production. You can use a profile to perform the changes necessary to migrate the test system’s structures into production. This profile can include a change rule that changes all table creators from one value to another.

If you support multiple production subsystems, some of the subsystems might have different requirements than others. Instead of performing the migration multiple times, you can use the change rules in an outbound migrate profile to tailor the worklists for each receiving location.

For example, your company might have naming conventions that include the department name as the owner of DB2 objects. Therefore, a table for the Denver sales office might be called DENVER.TABLE, while the same table that is used in Atlanta might be named ATLANTA.TABLE. You can use the change rules in an outbound migrate profile to automate the name changes. If you use that profile every time you migrate to Atlanta or Denver, the name change is automatically performed. (If you are not using an outbound migrate profile and are migrating to a single location, you can use change rules in a migrate-type work ID to perform the same function.)

If you have multiple locations or plan to migrate schema or changes more than once, you can create a profile and reuse it each time. If multiple locations that are defined in a profile use the same change rules, you do not have to define the change rules twice. You can define the change rules for one location, and then use that location as a reference location for the second location. For instance, if Denver and Toronto locations have similar needs, you can define rules for Denver and use it as a reference location for Toronto. Toronto then uses the change rules defined for Denver.

To create change rules for a migrate-type work ID, see page 117.

To create change rules for a migrate profile, see page 151.
Specifying values for change rule attributes

When you specify the current value or a new value of the attribute, you can use numeric and nonnumeric values, wildcard characters, and keywords. For a list of the attributes that you can use in a change rule, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Specifying numeric values

The current value that you specify and the actual value of the attribute must match exactly for numeric values of most attributes. You can specify the current value exactly or as a range of values for the following table space and index attributes that have numeric values:

- DSSIZE
- FREEPAGE
- PCTFREE
- PRIQTY
- SECQTY
- SEGSIZE

To specify a range of values, use a less than (<) or greater than (>) symbol with the current value of the preceding numeric attributes. For example, you can change all primary quantity values for table spaces that have a current value of less than 400 to a new value of 500. Figure 5 shows an example.

Figure 5 Specifying a range of values

<table>
<thead>
<tr>
<th>Act Opt Obj Attribute</th>
<th>Current Attribute Value</th>
<th>New Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TOP</td>
</tr>
<tr>
<td>C TS PRIQTY</td>
<td>&lt;400</td>
<td>500</td>
</tr>
</tbody>
</table>

If you want to change all of the values for one of the preceding numeric attributes, specify the maximum valid value plus one with a less than symbol. For example, specify <100 to include all values less than and equal to the maximum valid value for the PCTFREE attribute. The maximum valid value for PCTFREE is 99. Because you cannot use wildcard characters with numeric values, this type of range specification is equivalent to using a wildcard character. Figure 6 on page 50 shows an example.
Specifying values for change rule attributes

Figure 6   Using the maximum value plus one to specify all values

<table>
<thead>
<tr>
<th>Act Opt</th>
<th>Obj Attribute</th>
<th>Current Attribute Value</th>
<th>New Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C TS PCTFREE</td>
<td>&lt;100</td>
<td>10</td>
</tr>
</tbody>
</table>

For STOGROUP-defined objects, you can specify -1 for the current or new attribute values for the PRIQTY and SECQTY attributes. (If you specify -1, DB2 uses a default value for allocating the primary space.) The products consider -1 to be a special value. As such, you cannot combine the value with greater than (>), or less than (<) symbols, <DEFLT> or <CURRENT> keywords, or arithmetic operators (+, -, *, /). In addition, when you specify -1 for the current value, you cannot include the <CURRENT> keyword or an arithmetic operator for the new value. The new value must be numeric.

Specifying a new value based on a current value

You can specify new numeric values for the attributes of table spaces and indexes by using the following operations:

- + (Add)
- - (Subtract)
- * (Multiply)
- / (Divide)

The number that you specify for the operand must be a decimal or unsigned integer value.

For example, Figure 7 shows a change rule for table spaces that increases PRIQTY values that are less than 400 by one third. Current values are specified using the <CURRENT> keyword.

**WARNING**

'<CURRENT>' is a reserved keyword. Do not use it for other purposes.
Specifying special numeric values

For several attributes that have numeric values, you cannot use the greater than (>) or less than (<) symbols to specify a range of values. In addition, you cannot specify the new value with the <DEFLT> or <CURRENT> keywords, or with arithmetic operators (+, -, *, /). The following attributes have these restrictions:

- ASUTIME
- LOCKMAX
- PIECESIZE

You can specify an asterisk (*) for the current value for these attributes. The asterisk indicates that you want to change any current value.

Specifying nonnumeric values

You can specify exact values or use wildcard characters to specify a partial match for nonnumeric values. You can use an underscore (_) as a literal instead of a wildcard by preceding it with a back slash (\). The Current Attribute Value must have exactly the same number of wildcard characters in the same order if the New Attribute Value contains wildcard characters. The Current Attribute Value can use any combination of wildcards if the New Attribute Value does not contain wildcards. For more information about wildcard characters, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

Specifying default values

You can use the <DEFLT> keyword to specify default values for the buffer pool and STOGROUP attributes of table spaces and indexes, and for the WLM_ENV attribute of stored procedures. The actual string <DEFLT> is used as the attribute value if you specify <DEFLT> for a current value. The default value of the destination subsystem defines the attribute value if you specify <DEFLT> for a new value.

Figure 8 shows an example of a change rule that specifies to use the default value of the destination subsystem for the STOGROUP attribute for table spaces.
Specifying values for change rule attributes

Figure 8  Specifying a default value with a change rule

<table>
<thead>
<tr>
<th>Act Opt</th>
<th>Obj Attribute</th>
<th>Current Attribute Value</th>
<th>New Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WARNING**
Analysis builds the CREATE statement without a USING clause in the worklist if you specify <DEFLT> for the new value of the STOGROUP attribute of a table space or index. DB2 then uses the default storage group for the parent database of the object. The primary and secondary quantities will also revert to default values, which might be too small. <DEFLT> is a reserved keyword. Do not use it for other purposes.

Specifying blank values

You can specify a blank value by using the keyword <BLANK> for the attributes that are shown in Table 3.

Table 3  <BLANK> support for attributes

<table>
<thead>
<tr>
<th>Object</th>
<th>Editproc</th>
<th>Fieldproc</th>
<th>Label</th>
<th>Validproc</th>
<th>VCAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Table</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Global Temporary</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>View</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alias</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All objects&lt;sup&gt;a&lt;/sup&gt;</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<sup>a</sup> All objects indicates all of the objects that are listed in this table in which the attribute is supported.

The attribute that does not have a value defined is changed to the new value if you specify <BLANK> for a current value. Figure 9 shows an example of a change rule that specifies the CHKDATA validation procedure for a table.

**WARNING**
The procedure or label is removed from the object if you specify <BLANK> for a new value. <BLANK> is a reserved keyword. Do not use it for other purposes.
Specifying nonstandard characters

In a double-byte character set (DBCS), katakana and other lowercase characters are nonstandard characters. ALTER and CHANGE MANAGER require that these nonstandard characters be contained within double quotation marks, for example, “s”.

**NOTE**

Nonstandard characters that are not contained within double quotation marks will be automatically converted to uppercase letters by ALTER and CHANGE MANAGER.

Specifying nonprintable or nonviewable characters

You must specify nonprintable and nonviewable characters (such as null and control characters) in limit keys, view text, trigger text, or check constraint text in an external hexadecimal format. For more information, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*.

Specifying name qualifiers

Name qualifiers enable you to specify particular DB2 objects to which you can apply change rules. You can specify wildcard characters with name qualifiers. For more information about naming conventions for DB2 objects, see the IBM documentation.

Resolving change rules

The change rule list is built by retrieving the change rules from the rule table in the following order:

1. suppress-type rules
2. include-type rules
3. force-type rules

4. exclude-type rules

5. change-type rules

The products apply the rules according to the following guidelines:

- Within each type of rule, the objects are ordered so that the generic change rules (***) appear last in the list. As a result, change rules for specific objects are applied before generic rules.

- In the rule list, the first rule that applies to an attribute of an object is the only rule that is used for that attribute. Subsequent rules in the list that apply to that attribute of that object are ignored. However, subsequent rules in the list that apply to that attribute for other objects are used.

For example, suppose that a rule changes the name of column TB1.A to TB1.B, as shown in Figure 10.

**Figure 10  Subsequent change rules**

```
<table>
<thead>
<tr>
<th>Act</th>
<th>Opt</th>
<th>Obj</th>
<th>Attribute</th>
<th>Current Attribute Value</th>
<th>New Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TB</td>
<td>COLNAME</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>TB</td>
<td>COLNAME</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>
```

The second rule in the example is not applied to TB1.B. The second rule can, however, cause TB2.B to be changed to TB2.C.

- Rules are applied in descending order by current value and new value when they specify the same option, object, attribute, and name qualifier. If the new value is the same as the current value for an object, the rule is not applied. However, subsequent rules in the list that apply to the same attribute of the object are used.

For example, suppose that SECQTY equals zero (0). The first rule in Figure 11 on page 55 does not change the value for SECQTY because 0 * 2 equals 0 and the current value is the same as the new value. Because the second rule in the example changes the value for SECQTY to 10, the second rule is applied.
Resolving change rules

Figure 11 Application of change rules

<table>
<thead>
<tr>
<th>Act</th>
<th>Opt</th>
<th>Obj</th>
<th>Attribute</th>
<th>Current value</th>
<th>New value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TS</td>
<td>SECQTY</td>
<td>&lt;04</td>
<td>&lt;CURRENT&gt;*2.00</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>TS</td>
<td>SECQTY</td>
<td>&lt;05</td>
<td>&lt;CURRENT&gt;+10</td>
<td></td>
</tr>
</tbody>
</table>

Table 4 shows an example in which order dependency is significant.

Table 4 Ordering of change rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Option</th>
<th>Object</th>
<th>Attribute</th>
<th>Current value</th>
<th>New value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>TS</td>
<td>STOGROUP</td>
<td>SGOLTEST</td>
<td>SGTSPROD</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>TB</td>
<td>DATABASE</td>
<td>DBOLTEST</td>
<td>DBTBPROD</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>SG</td>
<td>NAME</td>
<td>SGO_TEST</td>
<td>SGO_PROD</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>DB</td>
<td>NAME</td>
<td>DBO_TEST</td>
<td>DBO_PROD</td>
</tr>
<tr>
<td>5</td>
<td>C</td>
<td>DB</td>
<td>STOGROUP</td>
<td>SGD_TEST</td>
<td>SGD_PROD</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>**</td>
<td>NAME</td>
<td>SG*TEST</td>
<td>SG*PROD</td>
</tr>
<tr>
<td>7</td>
<td>C</td>
<td>**</td>
<td>NAME</td>
<td>DB*TEST</td>
<td>DB*PROD</td>
</tr>
</tbody>
</table>

As shown in Table 4, Rules 1, 3, and 6 would work for table space storage group partitions named SGOLTEST. Because of the way the rules are ordered, rule 1 applies, changing SGOLTEST to SGTSPROD. A storage group named SGOBTEST is changed to SGOBPROD (rule 3). A storage group named SGOTEST is changed to SGOPROD (rule 6). Rule 5 would not apply to any of these names. Rules 2, 4, and 7 would work for tables in database DBOLTEST. The rule ordering would cause rule 2 to be applied, changing DBOLTEST to DBTBPROD. A database named DBOBTEST is changed to DBOBPROD (rule 4). A database named DBTTEST is changed to DBTPROD (rule 7).
Resolving change rules for storage group and VCAT partition volume lists

To resolve change rules for storage group and VCAT partition volume lists, the products apply the rules in the order shown in Table 5.

Table 5 Change rules for storage group and VCAT partition volume lists

<table>
<thead>
<tr>
<th>Order</th>
<th>Rule</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>include</td>
<td>applied first by using the original volume list to determine which additional volumes should be included&lt;br&gt;Any volumes that are added as a result of the include rules are kept apart from the original list. This action prevents included volumes from causing the inclusion of other volumes.</td>
</tr>
<tr>
<td>2</td>
<td>exclude</td>
<td>applied to the original volume list&lt;br&gt;For CREATE commands, at least one volume must be left after the exclude rules have been applied. This volume can be from the original list or can be an included volume. For ALTER commands, all volumes might be excluded. If this exclusion occurs and no other changed attributes exist for that object and no included volumes exist, the ALTER command is suppressed.</td>
</tr>
<tr>
<td>3</td>
<td>change</td>
<td>applied to the original volume list&lt;br&gt;Change rules are not applied to the included volumes. Only one rule is applied per volume. If multiple rules match a given volume name, only the first rule in the list is applied.</td>
</tr>
</tbody>
</table>

Propagating name changes for auxiliary objects

Analysis automatically propagates name changes for a dependent attribute when a parent attribute is changed. To propagate name changes for auxiliary objects for a LOB column, the products apply the following rules:

- If the name of a database that contains a LOB table space is changed, the database name in the LOB table space is changed.
- If the name of a LOB base table space is changed, the name of the base table that the auxiliary table references is changed.
- If the owner or the name of a base table is changed, the name of the base table that the auxiliary table references is changed.
- If the column name of a base table is changed, the column name in the base table that the auxiliary table references is changed.
If the owner or the name of the auxiliary table is changed, the name of auxiliary table that the auxiliary index references is changed.

Specifying migrate options

In a migrate-type work ID, you specify the installation options for a migration by selecting the types of dependent data structures that you want to migrate. (You select the parent objects in Specification.) You define the work ID migrate options when you create or edit a work ID, or specify the scope of an outbound migrate profile.

Consider the following information when you specify your options for a migration:

- If you have multi-table table spaces in a database, select the table spaces for migration.
- To ensure that the proper utilities are used when you have tables that are defined with referential constraints, mark all of the dependent foreign keys for migration before you generate the worklist.
- To ensure that data is unloaded with the ORDER BY clause of a SELECT statement for those tables that have a clustering index, select the indexes for migration.
- To migrate auxiliary objects for a large object (LOB) column, select to migrate the auxiliary objects and the base table. The base table can be selected as a dependent of a table space or database. If you choose to migrate data as a dependent of a table that contains a LOB column, the LOB data is not migrated.
- If you are migrating only the data structures, do not migrate data. By default, the product migrates data. To omit the data, deselect the Data option on the Analysis Worklist Command Options panel.
- By default, the product migrates data from clone tables. To omit clone data, deselect the Clone Data option on the Analysis Worklist Command Options panel.
- To migrate the parent objects of tables, include the dependent database and table space. If the parent object is an implicit object, CHANGE MANAGER ignores the object.
- If you are migrating only data into a table space on the receiving subsystem that has a different structure than the table space on the sending subsystem, alter the structure of the table space on the sending subsystem in the work ID. Analysis assumes that the structures on the sending and receiving subsystems are identical, and generates the LOAD statements based on the structure of the sending subsystem.
To migrate only the data for a single table in a segmented table space, select to migrate the table on the Mixed List panel. On the Migrate Options Overrides panel, select to migrate data and table spaces.

To migrate only the data for a single table at the table space level for a segmented table space, select to migrate the table space on the Mixed List panel. Specify to delete the tables in the table space that you do not want to migrate. On the Migrate Options Overrides panel, select to migrate data and tables.

To migrate only the data for more than one table at the table space level for a segmented table space, select to migrate the table space on the Mixed List panel. On the Migrate Options Overrides panel, select to migrate data and tables.

To migrate only the data for an entire table space at the table space level for a partitioned table space, select to migrate the table space on the Mixed List panel. On the Migrate Options Overrides panel, select to migrate data and tables.

TIP
To define migrate options for a migrate-type work ID, see page 116.

Analyzing the effects of a migration

After you create the change or migration requests in a Work ID or in a migrate profile, Analysis checks the requests for validity with the DB2 catalog, develops an optimal implementation strategy, and generates a worklist. The worklist contains the utility commands, AMS commands, DB2 commands, and SQL statements that are necessary for implementing the changes. In addition, Analysis propagates changes into dependent structures.

TIP
To analyze requests and generate a worklist, see page 153.

NOTE
For more information, view the Quick Course Performing Analysis. You must have a BMC Support ID to view the Quick Course.
Analyzing requests using work IDs

Analysis uses the migrate-type work ID’s migrate options when it analyzes the requests for a migration. These options tell Analysis which dependent objects to include in the migration (if they are not specified) and whether to migrate data and authorizations as well. For example, a migration option for a database named DBACM01 could specify to include dependent table spaces and tables, but not indexes, foreign keys, synonyms, aliases, triggers, or data.

Analysis also uses change rules from a migrate-type work ID. The change rules tell Analysis which global changes to make to objects that are being migrated. These changes include renaming objects, excluding columns from tables, or modifying the volume list for storage groups. For example, a rule might specify changing the name of all tables that match DEVTBL* to PRODTBL*.

Analysis reads the information in the CD tables for a work ID as its primary input. When you request a change or a migration by using Specification, the request is stored in the products’ CD tables. For each request, the CD tables contain the work ID the request was made under, the original object information, and the new object attribute information.

**NOTE**
The CD tables do not store information about object subelements unless a subelement is actually being modified. For example, if a work ID stores changes for a table, it will not store any information about the table’s columns unless the columns themselves are being modified.

When Analysis builds a worklist, the status of the work ID is updated from Created to Analyzed. If you specify the REBUILD keyword in the ALUIN input stream, the work ID status is set to Created at the beginning of the Analysis job. The REBUILD keyword is needed only if the work ID status is set to Exec Strted (Execution Started) or Exec Comp (Execution Complete).

In order to improve performance, Analysis can be installed to use a copy of the DB2 catalog rather than the catalog itself. Using a copy of the catalog is called catalog indirection. For more information about catalog indirection, see the Installation System User Guide and the BMC Products and Solutions for DB2 Configuration Guide.

**NOTE**
If you are experiencing performance problems, verify whether the recommended indexes were created when CHANGE MANAGER was installed. If you add the indexes after installation, rebind the plans. For more information about improving performance, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.
Analyzing requests using migrate profiles

In CHANGE MANAGER, Analysis can also use an outbound migrate profile to analyze the requests for a migration. If the profile contains scope rules, you can use them instead of the work ID change requests for selecting the objects to migrate. If the profile contains change rules, Analysis applies the rules before generating a worklist.

When Analysis uses an outbound migrate profile to generate a migrate-type worklist, it normally generates one worklist for each location specified in the profile. The JCL for each location uses the ddname WORKLnnn, where nnn is a three-digit, sequential number beginning with 001 for the first worklist. For example, if the profile contains three locations, the ddnames are WORKL001, WORKL002, and WORKL003.

If locations are grouped together using the group ID in the outbound migrate profile, Analysis generates a single worklist per group ID. If some locations have a group ID while other locations do not, Analysis generates one worklist per group ID and one worklist for each location whose group ID is blank.

NOTE

If you group several locations into a single worklist, the sequence number for worklist commands might exceed the maximum value allowed (999999). To avoid this problem, you can modify the sequence number increment (SEQI) value in the installation options module.

Use the MIGPROFILE keyword in the ALUIN input stream to indicate that Analysis should use an outbound migrate profile when you generate a migrate-type worklist. To instruct Analysis to use the scope rules of an outbound migrate profile rather than work ID migration requests, use the MIGSCOPE keyword in the ALUIN input stream. You can also include specific locations that are defined in an outbound migrate profile. To do so, specify the MIGLOCATIONS keyword to Analysis.

NOTE

Analysis does not apply the suppress rules that are specified in a migrate profile. Suppress rules are applied by the Compare and Import components only.

Selecting the run type

You can run Analysis either in the foreground or in batch. When you select to run Analysis in the foreground, the product builds the ALUIN input stream in a temporary data set. Additional foreground processing is not permitted until the input stream is built. After the input stream is built, you can edit it. During this time, the Front End sends the values for the POF keywords to Analysis. To generate a worklist, you run the input stream in Analysis.
When you select to run Analysis in batch, the product creates a batch JCL job that builds the ALUIN and AJXPOFIN input streams. The AJXPOFIN input stream includes most of the POF keywords and their values. After the product creates the JCL, you can edit it. To generate a worklist, you submit the batch JCL job.

**Overriding the default processing options**

Analysis uses the user options that you set in the product options panels to generate analysis JCL and to generate keywords in the ALUIN parameter input data stream.

If you run Analysis in the foreground or in batch, the Front End uses the values in the installation options module to initially build the keywords in the ALUIN input stream. The product also uses the values in the initial POF to build the AJXPOFIN input stream. The user options in the ISPF profile can override the keywords from the installation options module and ISPF profile in the ALUIN input stream. The Analysis options can temporarily override the keywords or options from the installation options module and ISPF profile in the ALUIN input stream. (If you are dynamically allocating copy or unload data sets, the JCL Generation options can also override the keywords in the AJXPOFIN input stream.)

In the foreground, you can manually edit the ALUIN keywords before generating a worklist. In batch, you can edit the batch JCL to manually override the ALUIN and POF keywords before generating the worklist.

You can use the Analysis options to override the default settings for

- allocation parameters for data sets
- global authorization IDs
- frequency of commands appearing between sync points
- elements and parameters that are included in a worklist
- utilities that are called from a worklist
- binds and rebinds

The values that you set through the Analysis options panels are not stored in your ISPF profile. They remain in effect only for that invocation of Analysis, until you return to the WORKID Action Menu or to the WORKID Selection List.
Overriding general options

The Analysis Worklist Options panel (see Figure 12) provides a way to override general analysis options.

Figure 12  Analysis Worklist Options panel

ALUFOAN1 ----------------- Analysis Worklist Options --------------------------
Command ==>  
Type information. Then press Enter to continue.

Specify allocation parameters for new datasets.
Dataset  Unitname  Volume  Priqty  Secqty  Alloc Unit
Worklist  SYSDA       15      5     Tracks

Specify optional Global AUTHID.
Global AUTHID  ..

Specify sync point frequency.
Frequency  .....

Commands:  HELP END

The options are defined as follows.

Allocation parameters

Specifies the allocation parameters for a new worklist data set.

Global AUTHID

Inserts the GLAUTHIDxxx keyword into the ALUIN input stream. The global authorization ID is the default authorization ID for a worklist. For information about using global authorization IDs, see “Using global authorization IDs” on page 96.

Frequency

Indicates the maximum number of -SQL commands a worklist can contain between -SYNC commands. For more information about sync points, see “Using sync tables” on page 101.
Overriding the commands to include in a worklist

The Analysis Worklist Command Options panel (see Figure 13) provides a way for you to specify the types of commands to include in a worklist.

Figure 13   Analysis Worklist Commands Options panel—migrate-type work ID

When you select the SQL, Data, AMS, or Bind options, Analysis inserts the INCLUDE keyword followed by one or more parameters that are enclosed in parentheses in the ALUIN input stream. The parameters are separated by spaces, as shown in the following example:

```
INCLUDE(DATA AMS SQL BIND)
```

The INCLUDE keyword only specifies which commands are included in the worklist. Analysis performs the same functions, but only the INCLUDE commands are written to the worklist.

The options are defined as follows.

**SQL**

Includes commands for SQL statements to implement change or migrate requests.
Data

Specifies to include commands to unload and load data, including Unicode LOB, EBCDIC LOB, ASCII LOB, XML, and clone data. By default, the Data and Clone Data options are selected. Data is included in a migration only if it is specified in the scope of a migration.

Consider the following items when you select this option:

- To include commands to unload, load, and copy data contained in clone tables, you must select the Data option and the Clone Data option.

- To ensure that the proper utilities are used when you have tables that are defined with referential constraints, mark all of the dependent foreign keys for migration before you generate the worklist.

- If you have multi-table table spaces in a database, select the table spaces for migration.

- To ensure that data is unloaded with the ORDER BY clause of a SELECT statement for those tables that have a clustering index, select the indexes for migration.

- If you are migrating only data structures, do not select the Data option. In addition, ensure that data is not specified in the scope of the migration.

When the Data option is not selected, Analysis generates a worklist that contains only SQL statements and does not contain commands to unload and load data.

- This option must be selected for worklist parallelism to be enabled. For more information about worklist parallelism, see “Enable worklist parallelism (Database Administration solution only)” on page 71.

- This option is selected by default if you are creating a full-recovery baseline. For more information about full-recovery baselines, see “Creating full-recovery baselines” on page 183.

AMS

Includes commands for IDCAMS ALTER, DELETE, and DEFINE statements. The AMS parameter also includes commands for specifying the IBM Storage Management Subsystem (SMS) STORCLAS, DATACLAS, and MGMTCLAS parameters to support VCAT-defined partitions. In addition, when the AMS installation option is set to C, the option includes the CATALOG parameter in the IDCAMS DEFINE CLUSTER statement.
Bind

Includes commands for binding the most recently bound packages and plans.

If you specify changes that invalidate one or more plans or packages, Analysis includes -BIND or -REBD commands in the worklist for these objects. For example, suppose you have a parent table that refers to a child table. If a new foreign key is defined for the child table and the delete rule is set to either CASCADE or SET NULL, the two tables are now in a delete-connected relation. In other words, a deletion of a row in the parent table affects the child table.

Defining such a constraint invalidates all plans on the parent table and the parent table would never be a part of the change request list in the CD tables. Analysis detects invalidated plans such as this and includes -REBD or -BIND commands in the worklist for them. If packages refer to the related structures, Analysis generates the appropriate -BIND or -REBD commands for package management.

Rebind (alter-type work ID only)

Includes commands for rebinding plans and packages when you use an alter-type work ID. Rebind commands are necessary when a change has occurred which will cause a plan or package to be invalidated. For more information about the use of rebind commands, see the discussion about “Bind” on page 65.

DBRMLIB

If Bind is selected, you can further specify the DBRMLIB option. Selecting the DBRMLIB option includes the LIBRARY parameter on the BIND statement for plans and packages.

NOTE

A disadvantage to adding the LIBRARY parameter to the BIND PLAN command is that the order of the libraries on the BIND might be incorrect. If some DBRMs are present in multiple libraries ALTER and CHANGE MANAGER cannot guarantee that the concatenation will result in every DBRM coming from the correct library.

Bindauth (migrate-type work ID only)

Includes commands for granting plans and package authorizations when you use a migrate-type work ID. The Bind option must be specified for the Bindauth option to be valid.

Analysis provides the ability to generate multiple worklists for the same work ID. You can, by using different options with the INCLUDE keyword, generate different worklists to carry out different portions of the work for a work ID. For example, you can create one worklist with INCLUDE(AMS), which would include file definition...
commands. The same work ID could be used to generate another worklist with INCLUDE(DATA, SQL), which would have unload, SQL, load, and other specified data utilities and commands. A third worklist, using INCLUDE(BIND), would contain the BIND statements. To execute the separate worklists, you must override the work ID status and select **Build Startover JCL from previous execution JCL**.

Execute the worklists in the following order:

1. Worklist containing INCLUDE(AMS) on the receiving subsystem
2. Worklist containing INCLUDE(DATA, SQL) in phase 1 on the sending subsystem
3. Worklist containing INCLUDE(DATA, SQL) in phase 2 on the receiving subsystem
4. Worklist containing INCLUDE(BIND) on the receiving subsystem

**NOTE**

A user could remove the second -STOP command in the worklist if the -AMS commands have already been executed.

**WARNING**

You can execute worklists that contain INCLUDE(DATA) or INCLUDE(SQL). When you execute these worklists, you might experience a problem with the foreign keys that are created before the data is loaded.

**Overriding the Analysis parameters**

The Analysis Input Stream Parameters panel provides a way for you to select optional parameters to be included in the ALUIN input stream for the Analysis component. **Figure 14** shows the Analysis Input Stream Parameters panel for a migrate-type work ID.
The options are defined as follows.

**Display installation environment**

Inserts the ENV keyword into the ALUIN input stream. This option provides additional diagnostics about the processing for Analysis and lists the keywords and the values in the installation options module. This option is also used to provide information to BMC Customer Support.

**Propagate parent key changes to foreign keys**

Inserts the PKEYPROP keyword into the ALUIN input stream.

Analysis provides a feature called *parent key propagation*. If you specify changes to the following elements of a column in a parent key and parent key propagation is enabled, Analysis propagates those changes throughout the DB2 system to any foreign key columns that reference the original parent key:

- data type
- length
- scale
- field procedure
- field procedure parameters
Overriding the default processing options

**NOTE**
If any of the foreign key columns are also used in the parent key of the table, changes are recursively propagated.

**Display errors as warnings for conditions in text**

Inserts the VALWARN keyword into the ALUIN input stream. Analysis flags errors in the trigger, view, materialized query table (MQT), or index text, but still creates a worklist. The errors are treated as warnings.

**NOTE**
You must correct the errors, verify the text and qualifiers in the worklist, or set the owner to the correct value before you execute the worklist. Otherwise, Execution fails.

**Propagate table column changes to base views**

Inserts the VVALPROP keyword into the ALUIN input stream. Select this option to propagate table column changes to base views. A base view is a view that contains all of the columns of one table in the same order and with the same names. A base view does not contain a WHERE clause.

For base views only, you can propagate the following table column changes when you select this option:

- column order
- addition of columns
- deletion of columns

A base view must meet the following criteria for Analysis to propagate the changes:

- If an explicit view column list or an explicit SELECT column list is specified for the view, the column lists must exactly match the column names from the table in the SELECT clause. Figure 15 shows an example.

**Figure 15  Explicit view column list**

```sql
CREATE TABLE A
  (COL1 INTEGER,
   COL2 SMALLINT,
   COL3 CHAR(15))
IN DB1.TS1;
CREATE VIEW VIEW.A (COL1, COL2, COL3)
AS SELECT COL1, COL2, COL3
FROM TABLE.A;
```
The name of each column in the view column list must be unique and unqualified.

- If an implicit view column list or an implicit SELECT column list is specified for the view, the columns of the view inherit the names of the columns in the table in the SELECT clause. Figure 16 shows an example.

![Figure 16 Implicit view column list](CREATE TABLE A
    (COL1 INTEGER,
     COL2 SMALLINT,
     COL3 CHAR(15))
    IN DB1.TS1;
CREATE VIEW VIEW.B
    AS SELECT *
    FROM TABLE.A;

If you do not select the option to propagate table column changes to base views, the NOVVALPROP keyword is inserted into the ALUIN input stream. This keyword specifies not to propagate extended view text.

**Omit the ORDER BY clause for unloads in the worklist**

Inserts the NOORDERBY keyword into the ALUIN input stream. When the work ID is analyzed, Analysis omits the ORDER BY clause from the unloads in the worklist. If this option is not selected, the product includes an ORDER BY clause when a clustering index is present on the table that is being unloaded.

**NOTE**

When the NOORDERBY keyword is analyzed, the subsequent load is performed from an unsorted file. The keyword improves the performance of unloads. However, a reorganization and a rebuild might be necessary to improve the performance of the application.

**Do not unload tables that IBM RUNSTATS indicates are empty**

Inserts the NOUNLOADEMPTY keyword into the ALUIN input stream. When you select this option, the product does not build SYSRnnnn data sets in the JCL for empty tables. A table is empty when DB2 catalog statistics show that the cardinality of the table is 0 (zero).

If Analysis determines that the table is empty, it generates the -ISMT command in the worklist. The command is used to ensure that the table is still empty at Execution. Analysis also generates the -ISMT command in the worklist if the DEFINE parameter is set to NO for an index or a table space. UNLOAD and LOAD statements will not be generated for the table.
Analysis does not generate the -ISMT command if one of the following conditions exists:

- a base table is incomplete

A base table can be incomplete under the following conditions:

- if the unique index for a ROWID column is missing
- if the auxiliary table space, table, or index is missing
- the DB2 catalog statistics show that the cardinality of the table is -1, which indicates that statistics do not exist

**Generate an AT (COMMIT) statement for every STOP command**

Inserts the STOPCOMMIT keyword into the ALUIN input stream.

**Create unique indexes with DEFER YES**

Allows the product to create new unique indexes with the DEFER YES parameter. The product inserts the DEFERUNIQUEIX keyword into the ALUIN input stream.

---

**NOTE**

To create an index with the DEFER YES parameter, you must also select the BMC RECOVER PLUS utility or the IBM REBUILD INDEX utility from the Analysis Utility Options panel. For information about using the Recover and Rebuild Utilities, see Chapter B, “Using utilities with ALTER and CHANGE MANAGER.”

---

**WARNING**

IBM recommends that you do not use DEFER YES when creating a unique index on a populated table. With DEFER YES, the table itself is not accessed when the CREATE INDEX statement executes; thus, any duplicate values do not cause the statement to fail. However, when you attempt to build the index with the REBUILD INDEX or REORG TABLESPACE, the job fails with an error message. If this happens, you must either drop and re-create the index as nonunique, or you must run the IBM REPAIR utility to remove the duplicate rows.

---

**Create all VSAM data sets with a CISIZE of 4 KB**

For VCAT-defined table spaces (partitioned, simple, segmented, and large object [LOB]), instructs Analysis to create VSAM data sets with a control interval size (CISIZE) of 4 KB, regardless of the page size (buffer pool size) of the table space. When you select this option, the product inserts the CISIZE4K keyword into the ALUIN input stream.
Enable worklist parallelism *(Database Administration solution only)*

For the Database Administration solution, instructs Analysis to include commands to run a worklist in parallel. When you select this option, the product inserts the PARALLEL keyword into the ALUIN input stream.

**NOTE**

The presence of the parallelism commands in the worklist does not signify that the worklist will be executed in parallel. To run the worklist in parallel, you must set the ACM_PARALLEL_WORKLST keyword to Y in the AJXPOFIN input stream or specify the Run Worklist in Parallel mode Execution override option. For information, see “Overriding the default processing options” on page 61.

When you select to enable worklist parallelism, CHANGE MANAGER automatically selects the following additional Analysis options:

- On the Analysis Utility Options panel, BMC UNLOAD PLUS is selected for *Unload* and BMC LOADPLUS is selected for *Load*.

- On the Analysis Image Copy Options panel, BMC COPY PLUS is selected for *Copy*.

- On the Analysis Utility Dataset Options panel, the *Use Dynamic SYSRE datasets* and *Use Dynamic COPY datasets* options are selected.

These options are required to enable Worklist parallelism. In addition, you must ensure that on the Analysis Worklist Command Options panel, the *Data* option is selected.

**Omit the -STOP command between phases of a migrate worklist (migrate-type work ID only)**

When analyzing a migrate-type work ID, inserts the SINGLEPHASE keyword into the ALUIN input stream. If you want to migrate your data structures, data, or both within the same subsystem, select this option to create a worklist that unloads data, creates structures, and loads data all in one step.
When you migrate your data structures, data, or both to a different subsystem, deselect this option to generate the -STOP command between the first and second phases of the migration.

Figure 17 shows the Analysis Input Stream Parameters panel for an alter-type work ID.

Figure 17  Analysis Input Stream Parameters panel—alter-type work ID

Command ===> Analysis Input Stream Parameters --------------------------

Select any of the following analysis parameters.

- ENV  Display installation environment
- PKEYPROP Propagate parent key changes to foreign keys
- VALWARN Display errors as warnings for conditions in text
- VVALPROP Propagate table column changes to base views
- NOORDERBY Omit the ORDER BY clause for unloads in the worklist
- NONLOADEMPTY Do not unload tables that IBM RUNSTATS indicates are empty
- STOPCOMMIT Generate an AT(COMMIT) statement for every STOP command
- DEFERUNIQUEIX Create unique indexes with DEFER YES
- CISIZE4K Create all VSAM data sets with a CISIZE of 4 KB
- PARALLEL Enable worklist parallelism
- NOTABLEACCESS Make tables inaccessible during execution

Commands: HELP END PREVIOUS

Reorg a table space when a column is added to a table (alter-type work ID on DB2 Version 8 only)

When analyzing an alter-type work ID, inserts the REORGALT keyword into the ALUIN input stream. If you alter a column at the end of a table, select this option to insert the REORG TABLESPACE command into the worklist.

NOTE

When you select this option, the product cannot reorganize the table space if the table space is in REORG-pending (REORP) restrictive status. To reorganize the table space, specify NOREORGALT and REORGPENDONLY, or REORGALT and REORGALL.

Make tables inaccessible during execution (alter-type work ID only)

When analyzing an alter-type work ID, inserts the NOTABLEACCESS keyword into the ALUIN input stream. Analysis inserts START DATABASE commands into the worklist to reduce access to the tables in the worklist during execution by putting objects into read-only status.
Validity and performance checks

When the Analysis component analyzes the requests for migration, it performs the following validity and performance checks. Errors are reported in the diagnostic output file, ALUPRINT.

- verifies that the definitions of changing objects have not been changed in the DB2 catalog since the change was requested

For migrate-type work IDs, if data is involved, the product verifies that no column changes have occurred. For example, if you request changing a column’s data type from CHAR(50) to CHAR(75), Analysis verifies that the column’s current data type really is CHAR(50). If not, Analysis issues an error.

- checks the change requests under the work ID that is being processed against the changes that are requested by other pending work IDs and warns of any conflicts

- checks the change requests for compliance with DB2 rules

Analysis reports errors if you attempted to create duplicate objects, you did not define a clustering index for a partitioned table space, and so on.
Validity and performance checks

- locates all dependent structures, which might exist in the DB2 catalog or might be the result of requested changes

Analysis verifies consistency across all objects and sets a strategy for each dependent structure. It then places actions in the correct sequence for rebuilding. Errors that are found are reported in the diagnostic output file.

**WARNING**

If you change the name of a storage group, Analysis issues a warning message if objects exist that use the old storage group name. Analysis will not automatically rebuild objects that are dependent on that storage group. You must modify the objects that use the storage group. You can use the TS or IX commands on the Mixed List panel to identify and edit the objects that use the storage group.

- checks for objects indirectly effected by the requested changes

This check provides warnings for objects that will be lost as a result of a DROP or an ALTER, as well as warnings for views and plans that might require modifications.

If no errors are found, Analysis builds a worklist that contains commands to implement the changes. The data set name of the worklist with the work ID is stored and used as the default worklist when you execute the work ID.

- checks the objects that are specified in the scope of the migration

If an object is not allowed, Analysis generates an error and does not create a worklist.

- develops an efficient strategy for implementing the requested changes

For example, for an alter-type work ID, Analysis uses the SQL ALTER statement for any requested changes that can be accomplished in that manner. Analysis only uses a drop and rebuild strategy if it is required.

- uses the DB2 catalog statistics that are gathered by the BMCSTATS or the IBM RUNSTATS utility to verify whether a DB2 object contains data

**NOTE**

In a migrate-type worklist in which data is not included, CHANGE MANAGER does not verify whether the data set is defined.
Validity and performance checks

You can specify to create a worklist even though it contains errors related to views or triggers. The worklist will not execute until you correct the problem. For example, the following errors can be detected by Analysis, yet it will still create the worklist:

- A table references a trigger that was not found.
- A table is referenced by a view or trigger that has been dropped.
- A column name used in the view or trigger was not found.
- A column referenced by a view or trigger exists in multiple referenced tables.
- The owner name for an unqualified view or trigger references an unqualified table with a different owner name.
- A syntax error exists in a view or in a trigger.

Analysis creates the worklist, provided that the options to display errors as warnings are selected. For more information about these options, see “Display errors as warnings for conditions in text” on page 68.

**Validation of text dependencies**

Analysis implements text validation for the following objects to enhance making changes to these objects or dependent objects of the objects:

- views
- triggers
- check constraints
- indexes on an expression
- materialized query tables (MQTs)

When a work ID contains requests to modify any of the objects or columns that are used in a view, trigger, check constraint, index on an expression, or MQT, the text validation feature within Analysis performs the following functions:

- checks for the existence of all columns and objects that are used in the text

Analysis determines whether dependent objects and their columns exist. Analysis always performs this check for an alter-type work ID. For a migrate-type work ID, however, Analysis performs the check only if the dependent objects of the object are included in the migration.
checks the basic syntax

Analysis validates the basic structure and syntax of the text, identifies objects and columns, and determines the scope of correlation names. It does not guarantee that a view, trigger, check constraint, index on an expression, or MQT is accepted by DB2. However, Analysis does catch the majority of syntax errors. Analysis generates warning messages if it finds any errors.

propagates object and column name changes into the text

If you change the names of the objects and columns that are used in the view, trigger, check constraint, index on an expression, or MQT in the work ID, Analysis propagates those name changes into the text. After the names have been changed, Analysis checks for ambiguous column names that are not qualified by object name in the text. For a migrate-type work ID, Analysis applies change rules to the object and column names. For more information, see “Resolving change rules” on page 53.

Analysis resolves object names through aliases, synonyms, and other objects to determine the base object. This search applies to the following names:

— names that are used in base clauses of the view, trigger, check constraint, index on an expression, or MQT

— names that are used in subselect statements and correlated subqueries

By resolving all object names to the base object, Analysis accounts for all proposed changes in the worklist.

cascades column name changes through the column names

Analysis also verifies that columns in the SELECT column list are consistent with the column list, if a column list is specified. The explicit column list of the object must have the same number of names as expressions in an explicit SELECT column list in the text.

If you specified an implicit column list for the object and an implicit SELECT column list for the text, Analysis cascades column name changes into other objects that use the modified object as a dependent. If you specified an implicit SELECT column list for the text and additional columns exist in the referenced object, the column list expands to match the new object column list. Analysis rebuilds implicit column lists with the same names and the same numbers of names as in the SELECT column list in the text. Analysis cascades the changes to the column names if the column names were changed by the user, by propagation, or by a change rule.
Normally, Analysis does not modify text except to change the names of objects and columns that are used in a view, trigger, check constraint, index on an expression, or MQT. Therefore, if you drop a column from a table that is used by one of these objects, Analysis does not drop the column from the text. Analysis always preserves the exact form of the text; Analysis marks any inconsistencies involving dependent object definitions as errors that you must resolve.

**NOTE**

Analysis drops the column from the view SELECT column list and the explicit view column list if the following conditions exist:

- the view is a base view
- you have selected the Analysis override option to propagate table column changes to base views (VVALPROP ALUIN keyword)

(views only) reorders views for worklist output

After propagating name changes into the view text, Analysis captures the modified view dependency list for all views that need to be changed or created. Analysis uses this list to build CREATE VIEW statements in the correct order.

**Rebuilding indexes and tables**

Analysis includes worklist commands to rebuild indexes whose keys contain table columns that have been dropped. The dropped table column is removed from the key and the index is rebuilt. If all table columns in the key have been dropped, the index is dropped.

Tables with DROP RESTRIC cannot be dropped until the table is altered and the drop restriction is removed. The parent table space and database of a drop-restricted table cannot be dropped either. Analysis generates the ALTER statement to remove the DROP restriction on the table and issues a warning message that the DROP restriction will be removed. The table is rebuilt with DROP RESTRIC.
Reviewing the results of Analysis

When you migrate data structures, Analysis produces two types of output: worklists and diagnostic output.

Reviewing the migrate-type worklist

When you migrate data structures, Analysis generates a special type of worklist, called the migrate-type worklist. The migrate-type worklist is a sequential, 80-column data set (or PDS member) that contains worklist commands. The worklist also includes a list of the keywords in the ALUIN input stream. When you run Analysis in batch mode, the worklist data set is identified by the ddname WORKL001.

NOTE

Analysis generates the worklist commands in a specific order. You can edit the worklist, but changing the order of commands might yield incorrect results.

Each worklist command (such as -TIME, -SSID, or -SQL) consists of a dash followed by a three- or four-character command identifier, and a blank in column 6. The command identifier is followed by a six-digit sequence number. The sequence number is used when worklists are restarted. The contents of the command line following the sequence number vary according to the command. For example, the -AUTH command requires a user ID to follow the sequence number. The last line of each command, in columns 73–80, contains a hash verification number. CHANGE MANAGER uses the hash number for diagnostics on the worklist file to detect changed commands and inserted commands. Do not insert or modify this number.

NOTE

If hash fail is enabled, any modification to worklist commands or hash verification numbers ends worklist execution.

The worklist that is generated for a migrate-type work ID is very similar to one that is generated for an alter-type work ID, with the following differences:

- A migrate-type worklist does not contain SQL DROP statements.
- Most migrate-type worklists are separated into two parts, called phases, indicated with the -MIGR PHASE-1 and -MIGR PHASE-2 worklist commands. If you are migrating data with the data structures, phase 1 contains the unload commands for extracting the data from the sending subsystem. Phase 2 contains the commands for creating the data structures at the receiving subsystem and commands for loading the migrated data, if necessary. A migrate-type worklist is processed in one phase if you do not include data in the migration or if you select the Omit the -STOP command between phases of a migrate worklist Analysis option.
NOTE

If you are using the BMC RECOVER PLUS utility, you will not have an unload or load phase.

For more information about commands that the products generate in a worklist, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*. For more information about worklists that are created from an alter-type work ID, see Chapter 4, “Altering data structures in a database environment.”

Editing the migrate-type worklist

Certain types of tasks involve modifying worklists and data input streams. The ability to modify a worklist, or to edit the input stream for Analysis or Execution, enables you to customize tasks. You can edit the worklist in the following ways:

- in the Analysis JCL Processing Interface panel, select Edit Worklist

  If you generate a worklist using an outbound migrate profile with locations, you cannot edit the worklist in foreground Analysis.

- in the WORKID Selection List panel, type W in the Act column adjacent to the work ID that you want to use

  The work ID must have a status of Analyzed, Exec Strt, or Exec Comp. In addition, the work ID must not be used with an outbound migrate profile that specifies multiple locations, because the profile can be used to generate multiple worklists. For more information about the status of a work ID, see “Viewing the status of Execution” on page 99.

Migrating within the same subsystem

If you are migrating data structures, data, or both within the same subsystem, you can edit the migrate-type worklist and remove the -STOP and -MIGR commands between the two phases of the worklist before you submit it for processing. If a global authorization ID is in effect, you should also remove the second -GLID command.

Specifying the work ID

The work ID that is specified for Execution to process phase 2 of the worklist must match the work ID specified with the -WKID command in the worklist. If you specify a work ID with a different name from the migrate-type work ID that is used to create the worklist, edit the worklist and insert the proper work ID name in the -WKID command for phase 2.
Migrating authorization commands

If you neglected to migrate your authorization commands when you created the worklist to migrate your data structures, you can use that worklist to create a worklist that contains only the authorization commands by using the ALUXGRNT REXX executable. This executable creates an additional worklist that contains -SETS and -SQL authorization commands only. You can run the executable when you edit the worklist by typing the following text on the COMMAND line:

```
ALUXGRNT 'new_worklist_data_set'
```

Alternatively, you can run the executable from the ISPF COMMAND line by typing the following text:

```
ALUXGRNT 'oldWorklistDataSet' 'newWorklistDataSet'
```

Including macros

You can include a REXX or CLIST macro in a worklist by editing the worklist to include a -REXC command. A -REXC command allows a subroutine to execute outside of the worklist and return back to the worklist after the subroutine is completed.

To use this feature, you must

- allocate any files required for processing either dynamically or through JCL

You can locate the REXX executable either implicitly or explicitly:

- For implicit data set location of the REXX executable, the worklist must contain:

```
-REXC 000450 MYREXX
```

`MYREXX` is the member containing the REXX executable. You must manually insert, for foreground or batch processing,

```
\%SYSPROC DD DSN='location.of.REXX.exec'
```

in the Execution JCL.

- For explicit data set location of the REXX executable, the worklist must contain:

```
-REXC 000450 EX 'location.of.REXX.exec(MYREXX)'
```
Reviewing the results of Analysis

- Ensure that the return code for the REXX executable is sent back through IKJEFT1B by including a final REXX statement of `EXIT RC` (where `RC` is the return code to pass back to AEXEMAIN).

**NOTE**
If the required return code is not passed back to AEXEMAIN, unless a TSO command abends, AEXEMAIN will receive a return code zero (`RC=0`) and continue to process the worklist.

**WARNING**
BMC strongly recommends that you evaluate any REXX executable inserted in worklists in a test environment before implementing production versions of the macro. Take particular care to determine whether any data sets are freed by the REXX executable that are allocated by ALTER or CHANGE MANAGER. Execution of REXX macros that free data sets allocated by ALTER or CHANGE MANAGER could have drastic and unpredictable results.

Checking the sequence numbers in a worklist

You can run the ISPF edit macro CHKSQNUM against a worklist to determine whether any out-of-sequence numbers exist. You should run this macro if you have manually edited the worklist. To run the macro from within an edit session, type `CHKSQNUM` on the `COMMAND` line. For more information, refer to the CHKSQNUM macro in the `HLQ.UDBCLIB` data set.

If you found any out-of-sequence numbers, you can run the ISPF edit macro FIXSQNUM against the worklist to correct the problem. To run the macro from within an edit session, type `FIXSQNUM` on the `COMMAND` line. For more information, refer to the FIXSQNUM macro in the `HLQ.UDBCLIB` data set.

If a unit of work includes numerous DB2 objects, commands, and utilities, the sequence number for worklist commands might exceed the maximum value allowed (999999). To avoid this problem, you can modify the sequence number increment (SEQI) value in the installation options module.

Reviewing the diagnostic output

Analysis also produces diagnostic output, ALUPRINT, which consists of status messages, error messages, and warnings that are generated during the analysis process. The diagnostic output might be assigned to TERM (in foreground), assigned to SYSOUT, or sent to a sequential file. You cannot assign the output to a member of a partitioned data set.

Several conditions can cause Analysis to issue warnings or to halt processing. For a description of some of the checks that can generate warnings or errors, see “Validity and performance checks” on page 73. For information about error and warning messages, see the Administrative Products for DB2 Messages Manual.
**Executing changes**

The Execution component performs the commands in the worklists that ALTER and CHANGE MANAGER generate. Execution is the only component of ALTER and CHANGE MANAGER that makes physical changes to DB2 data structures or data. Specification and Import are used to build a set of change requests, while Analysis generates a worklist from those requests. Execution performs the following tasks:

- executes SQL
- unloads DB2 tables
- switches DB2 authorization IDs
- runs DB2 utilities
- establishes synchronization (sync) points
- performs restarts

**NOTE**
For more information, view the Quick Course Executing a Worklist. You must have a BMC Support ID to view the Quick Course.

**Specifying options for Execution**

The Execution Interface panels provide you with a way to specify processing options and direct the flow of the Execution job. In addition to specifying options to process a worklist, you can specify JCL generation options.

When you run Execution, you

- select JCL build options
- indicate whether you want to generate JCL online (in the foreground) or in batch
- specify whether to create a multistep job, which includes creating a baseline and performing a comparison
- specify the Execution JCL, worklist, and diagnostics data set names

You can also override defaults and edit the worklist before you submit the execution job.
Specifying options for Execution

Chapter 1 Migrating database environments 83

Generating JCL

When you generate JCL for Execution, the Front End component of ALTER or CHANGE MANAGER uses symbolic variables to resolve all data set names that appear on the interface panels of the components. Parameters (from the product options file [POF] and Execution panels) are passed to Batch Execution JCL Generation using the AJXIN or AJXPOFIN input streams. These parameters include the names of input files, JCL files, and the diagnostic output files. (For more information about the AJXIN and AJXPOFIN input streams, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.)

After the data set names have been resolved, the JCL Generation component performs the following functions:

- scans the input worklist for all utilities and commands that will require JCL
- analyzes each utility command for its DD requirements

  Depending on the override options that you select, the following tasks might be performed for each identified DD:

  — sizing the data sets
  — using the values specified for using DASD or tape units, as well as the values for tape-related options
  — using the values specified for the SMS and alternate SMS data sets, data set thresholds, and alternate units
  — processing the options for image copy GDGs

For more information about override options, see “Overriding the default JCL Generation options” on page 90.

- resolves the names of work data sets that JCL Generation passes from the JCL Generation option panels, such as SORTWORK, and the unload data sets that Execution uses
- merges DD statements (for example, SYSUT1 or SORTWORK) that more than one command uses so that the component does not generate duplicates and uses the highest estimated space

_TIP_

To execute a migrate-type worklist, see page 158.
JCL Generation either performs standard ISPF file tailoring or simulates file tailoring by using compiled skeleton libraries (SLIBs) to generate JCL. A large number of symbolic variables are available to automatically vary the JCL generated according to, for example, the subsystem name and the database name. These symbolic variables are listed in the ALTER and CHANGE MANAGER for DB2 Reference Manual.

The generated JCL includes DD statements for all data sets that the job or the Execution component needs, as well as the EXEC statement for the program and any necessary control parameters. For many of the work data sets that Execution uses, you can create the JCL for a cleanup job step that automatically deletes the work data sets at the end of the run. Other data sets are commented out in the JCL, providing you with the option to delete these as well.

**Specifying the type of JCL**

You can build the following types of JCL:

- Initial JCL
  
  Initial JCL initially starts the job. The JCL is built to run the job for the first time from the worklist. The disposition of the data sets to be allocated is set to NEW.

- Restart JCL
  
  Restart JCL restarts the job after it fails or stops with the -STOP command. You can build restart JCL from existing JCL. The product scans the previous JCL and worklist and modifies the file dispositions. In addition, the product preserves any data set name changes that were made to the old JCL and eliminates data set sizing and symbolic variable substitution. To restart the worklist, the product reads the sync table.

  In the course of executing a worklist, output is created in the form of records being inserted into the sync table. Sync table records are used as progress indicators during execution of a worklist and as a means of committing work to DB2. These records are used by Execution to restart or resume a job that has failed or has been stopped. (For more information about sync tables, see “Using sync tables” on page 101.)

  For more information about restarting a worklist, see “Restart processing” on page 107.
Startover JCL

Startover JCL executes the job from the beginning of the worklist or after the -STOP worklist command, depending on the type of worklist that you are processing.

— If you are processing an alter-type worklist or a single-phase migrate-type worklist, startover JCL starts the job over from the beginning of the worklist after it has failed.

— If you are processing a two-phase migrate-type worklist on the sending subsystem, the JCL starts over from the beginning of the worklist.

— If you are processing a two-phase migrate-type worklist for a receive-type work ID on the receiving subsystem, the JCL starts after the -STOP command, which is the beginning of the second phase of the worklist. For more information about receive-type work IDs, see “Creating a receive-type work ID” on page 155.

NOTE

You can insert the -GOTO command after the first -SYNC command in the worklist, or in phase 2 of a migrate-type worklist on the receiving subsystem, to allow processing to start at a specified sequence number. For more information about the -GOTO command, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

You can build startover JCL from existing Execution JCL. The product scans the previous JCL and modifies the file dispositions. In addition, the product preserves any data set name changes that were made to the old JCL and eliminates data set sizing and symbolic variable substitution.

For more information about starting a worklist over, see “Startover processing” on page 111.

Indicating the type of run

You can create Execution JCL in the foreground or in batch mode.

Build JCL in foreground

Specifies that the JCL Generation component should generate JCL in the foreground. Additional foreground processing is not permitted until the generation of the JCL is complete. This option is not recommended for processing a large worklist. Creating Execution JCL for large jobs, such as a long worklist with data set sizing using VSAM object sampling, can take several minutes and make your workstation unavailable until the JCL is created.
Specifying options for Execution

**Build JCL in batch**

Enables you to create a Batch JCL job that generates the Execution JCL that is required to process the specified worklist. Using this option enables you to perform other tasks at your workstation while the worklist is generated. It also enables you to create debugging information to send to BMC, if necessary.

**Creating a multi-step job**

In CHANGE MANAGER, when you generate Execution JCL from the Front End, you can optionally create a multi-step job. The job can include such steps as creating a baseline and performing a comparison before or after the actual execution run. While you can create a multi-step job for a migrate-type work ID, this feature is commonly used for an alter-type work ID. For information about creating a multi-step job, see “Creating a multi-step job” on page 349.

**NOTE**

Because Compare and Baseline are not components in the ALTER product, you cannot create a multi-step job in ALTER.

**Specifying the names of data sets**

You can specify the names of your data sets and other processing options in the Execution JCL Processing Interface panel (see Figure 18).

**Figure 18  Execution JCL Processing Interface panel—foreground**

ACMFEXC3 --------------- Execution JCL Processing Interface -------------------
Command ===>  
WORKID . . . . : RDACRJ.MIGRATE2  
Specify Dataset Names.  
  Execution JCL . . : 'RDACRJ.V10.EXECJCL(MIGRATE2)'  
  Worklist . . . . : 'RDACRJ.V10.WLBASE01(MIGRATE2)'  
  Diagnostics . . . : SYSOUT  
JCL Build Options.  
  Dataset Sizing : No Sizing (use Override Defaults to set or change)  
Select Processing Options. Then press Enter to continue.  
  _ Override Defaults  
  _ Edit Worklist  
  S Build Execution JCL  
  S Edit Execution JCL  
  _ Submit Execution JCL  

Commands:  BROWSE VIEWX PREVIOUS HELP END
If you select Build Initial JCL as the JCL type and Build JCL in Batch as the run type in the Execution JCL Build Interface panel, you must specify a data set name for the batch JCL job.

If you select Build Restart JCL from previous execution JCL or Build Startover JCL from previous execution JCL in the Execution JCL Build Interface panel, you must additionally specify a data set name for the old Execution JCL, as shown in Figure 19.

**Figure 19  Execution JCL Processing Interface panel—batch**

![ACMFEXC3 Execution JCL Processing Interface](image)

If you select Build Restart JCL from previous execution JCL or Build Startover JCL from previous execution JCL, Execution performs the following functions:

- changes the last character of an eight-character member name for the old Execution JCL to a dollar sign ($) and uses that member name as the name of the new Execution JCL

  If the member name is less than eight characters, the dollar sign ($) is appended to the member name.

**NOTE**

The old Execution JCL must be a member of a PDS and the member name must not already end with a dollar sign ($).

- produces an error if you use the same data set member for the old Execution JCL and the new JCL
Specifying options for Execution

- automatically saves the name of the new Execution JCL data set in the work ID
- uses the worklist that is specified in the old Execution JCL

Execution does not perform data set sizing and symbolic variable substitution.

Specifying data set sizing

To specify the method of data set sizing, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

Overriding the default processing options

You can select Override Defaults in the Execution JCL Processing Interface panel to override the default options that are set for Execution. When you select this option, the Execution Override Options panel (see Figure 20) is displayed. On this panel, you can specify whether to invoke the CATALOG MANAGER Drop Recovery function and whether to generate JCL for the send phase (phase 1) of a migration.

NOTE
You cannot select the Override Defaults option when you select to build restart or startover JCL from previous Execution JCL.

Figure 20 Execution Override Options panel—migrate-type work ID

ALUFOEX1 -------------- CHANGE MANAGER Execution Override Options --------------

Command ===> 

Type S to select the function. Then press Enter to continue.

- Invoke BMC CATALOG MANAGER Drop Recovery
- Generate JCL for the send phase only

Commands: HELP END
The options are defined as follows.

**Invoke BMC CATALOG MANAGER Drop Recovery**

Invokes the CATALOG MANAGER Drop Recovery option when DB2 objects are dropped in an alter-type work ID. JCL Generation inserts the CATRECOVER keyword into the AEXIN input stream. This option causes Execution to invoke CATALOG MANAGER to log the dropped objects. CATALOG MANAGER can rebuild the objects. To use this option, you must have CATALOG MANAGER installed.

---

**NOTE**

To recover data, as well as data structures, you should use a full-recovery baseline instead of the CATALOG MANAGER Drop Recovery option. This option only recovers data structures.

**Generate JCL for the send phase only (migrate-type work ID only)**

Allows you to generate JCL for only phase 1 of a migration. For Batch Execution JCL Generation, selecting this option inserts the SENDONLY YES keyword into the AJXIN. The data sets that are required by the utilities in phase 2 are not included in this JCL.

**Determining How to Execute a Worklist**

A migrate-type worklist for migrating within the same subsystem might contain only one phase, in which all of the worklist commands are executed as one job. A migrate-type worklist for migrating to another subsystem, however, would contain two phases, each identified by the -MIGR worklist command. A typical migrate-type worklist that includes both DDL and data has unloads in phase 1, followed by DDL, loads, and other utilities in phase 2. A -STOP worklist command separates the two sections. Phase 1 is executed on the sending subsystem, and phase 2 is executed on the receiving subsystem.

By default, JCL is generated initially for a migrate-type work ID to execute both phase 1 and phase 2. If you want to generate the JCL for only phase 1, you can select the Generate JCL for the send phase only option on the Execution Override Options panel. Selecting this option prevents the deletion of data sets when the work data sets are put on temporary work packs, or prevents the unnecessary usage of space if the receive-type worklist and JCL are not executed immediately.
When you migrate data, data structures and data, or data structures only to a different subsystem, you must use a receive-type work ID on the receiving subsystem to execute phase 2 of the worklist. Phase 2 of the worklist creates the data structures and loads any migrated data. Receive-type work IDs cause Execution to begin processing with phase 2 of the worklist. Execution reads the worklist and searches for the -MIGR PHASE-2 command. If Execution does not find the command, it processes the worklist from the beginning.

When the worklist is executed in two phases, you must also create a receive-type work ID on the receiving subsystem to build the new data structures and load any migrated data. The receive-type work ID cannot contain any migrate or change requests. It is only used to track the execution process and to restart the worklist if any errors occur. In addition, if the receiving subsystem does not have access to the data sets on the sending subsystem, or if the two subsystems do not share the same DASD, you must transfer the worklist and all of the SYSRnnnn data sets (unloaded flat files) to the receiving subsystem.

**TIP**

To create a receive-type work ID, see page 155.

**NOTE**

For more information, view the Quick Course Creating Work IDs. You must have a BMC Support ID to view the Quick Course.

---

**Overriding the default JCL Generation options**

You can also select Override Defaults in the Execution JCL Processing Interface panel to override the JCL Generation options. On the Execution Override Options panel, press Enter to display the JCL Generation Override panel. The JCL Generation override panels enable you to override the user options for ALTER or CHANGE MANAGER. The override values update the AJXPOFIN input stream.

You can override the following options with these panels:

- jobcard information
- names of load libraries
- data set sizing
- data set cleanup
- tape units and stacking preferences
Specifying options for Execution

- individual data set characteristics
  - primary and secondary quantity
  - threshold values
  - SMS names
- GDG definitions
- debugging
- installation option module names
- worklist parallelism

For information about using the JCL Generation override panels, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

---

**NOTE**
Choosing to dynamically allocate your copy or unload data sets negates any changes that you make to the SYSCOPY, SYSREC, baseline recovery data set, or tape options in the JCL Generation override panels.

---

**Batch Execution JCL Generation Report**

JCL Generation produces a Batch Execution JCL Generation Report file. The report, AJXPRINT, consists of messages and job options. The report output might be assigned to TERM (in foreground), assigned to SYSOUT, or sent to a sequential MVS file. You cannot assign the report to a member of a PDS.

**Execution JCL**

You can edit and review Execution JCL before you run a job.

---

**NOTE**
JCL Generation generates DISCARD, SYSMAP, SYSER001, SORTWORK, SYSUT, and SORTOUT data sets that are needed for a -BMCL or -LOAD command. References to SYSMAP, SYSER001 and SORTOUT are optional in a worklist but are required to run the load utility.
When worklist parallelism is enabled in the Database Administration solution, CHANGE MANAGER includes several additional features in the JCL:

- a preallocation step (\PREALLOC) for allocating data sets
- the AEXPIN input stream
- the DYNWORKUNIT keyword in the AEXIN input stream
- a data set cleanup step (\CLEANUP) for deleting the work data sets

**Preallocation step**

The Execution JCL includes a preallocation step in which permanent work data sets are created for parallel processing. For each initiator, several of each of the types of work data sets (such as SYSUT, SUT, SORT, SORTP, and SYSER) are allocated. You must specify a DASD unit, not tape, for these work data sets. In addition, you must not specify a threshold value for tape for these data sets.

The data sets are indicated by the .UOWnn suffix, where nn is the number of the initiator. The number of data sets that is allocated is influenced by the maximum number of initiators. These data sets will be allocated, regardless of whether the maximum number of initiators is used.

**AEXPIN input stream**

The XIMSTART, XIMPROC, MAXINITS, MININITS, TRACE, and PARALLEL keywords are included in the AEXPIN DD step. You can also specify the following additional AEXPIN keywords:

- XIMRETRYMINS \(n\), which specifies the number of minutes that Execution waits until it attempts to schedule initiators that could not be scheduled because of resource shortages

**NOTE**
You cannot specify both the XIMRETRYMINS keyword and the XIMRETRYSECS keyword.

- XIMRETRYSECS \(n\), which specifies the number of seconds that Execution waits until it attempts to schedule initiators that could not be scheduled because of resource shortages

**NOTE**
You cannot specify both the XIMRETRYSECS keyword and the XIMRETRYMINS keyword.
Specifying options for Execution

- **XIMRETRYCOUNT** $n$, which specifies the maximum number of attempts that Execution makes to schedule the initiators that could not be scheduled because of resource shortages

  If the initiators cannot be allocated within the specified number of retry attempts, Execution terminates.

- **XIMPCTSURMN** $n$, which specifies the percentage of the total computing resources that should be reserved

  For example, if you set XIMPCTSURMN to 10 percent and the total service unit (SU) capacity of an image is 45,000 SU per second, an XIM initiator would not start if the estimated SU availability is less than 4,500 SUs.

**DYNWORKUNIT keyword**

The DYNWORKUNIT keyword in the AEXIN input stream is automatically generated in the Execution JCL. This keyword defines the unit (such as SYSDA) that Execution uses to dynamically allocate temporary work data sets.

**Cleanup step**

When you enable parallelism, if you did not specify to include a cleanup step, one is automatically generated in the JCL to delete many of the permanent work data sets. To execute the step, you must manually change the step condition code to COND=(4,LT). For example,

```
//CLEANUP EXEC PGM=IEFBR14,COND=(4,LT)
```

**Running the JCL multiple times**

To run your JCL several times on different subsystems without having to use a different data set for each subsystem, use cataloged procedures (which begin with a PROC statement) in your JCL. In the PROC statement of the procedure, manually specify the SSID as the second value of the PARM parameter in the EXEC statement, as shown in the following example:

```
//PROC ...
//STEP EXEC PGM=<ALUAMAIN or AEXEMAIN>,REGION=0M,
//       PARM='QA101BDC, DBDC'
```

The value of the SSID overrides the value of the SSID keyword in the ALUIN and AEXIN input streams.
Controlling authorizations

The Execution component performs an authorization switching function to ensure that DB2 structures are created with the proper owner and creator. To secure this function, restrict the EXECUTE authority on the Execution Main plan, use the Execution security exit, or do both.

You might want to switch authorizations during the execution of a worklist for the following reasons:

- to create DB2 objects with the proper owner
- to create DB2 objects with the proper creator (the DB2 catalog CREATEDBY field)
- to obtain the necessary authorizations for performing a particular task

Execution provides commands and operational modes for controlling authorizations. The following parameters or keywords determine which commands and modes are used:

- AUTHSW parameter in the installation options module
- AUTHSW, NOAUTHSW, AUTHSWGLID, and AUTHSWOFF keywords in the ALUIN input stream

The user who executes the worklist might not have the necessary authorizations for all of the actions that occur in the worklist. To solve this problem, you can specify a global authorization ID (GLID) to execute the worklist.

Setting the authorization switching modes

For some objects, such as tables and indexes, you can specify the owner as part of the fully-qualified name. Therefore, authorization switching is not required for those objects. For other objects, such as table spaces and synonyms, the owner cannot be specified as part of the name, so it is implied by the current SQLID. The owner for these objects can be changed with either of two techniques, -AUTH or -SETS command processing, which are determined by the setting of the AUTHSW keyword in the installation options module. Analysis inserts the -AUTH or -SETS commands in the worklist, which is processed by Execution.

The AUTHSW keyword can be set to the following values in the installation options module:

- If AUTHSW=N, the products generate the -SETS command in the worklist to change the authorization IDs with SET CURRENT SQLID statements. For additional security, you cannot edit -AUTH commands into the worklist. This option uses DB2 secondary authorization IDs.
Controlling authorizations

If AUTHSW=Y, the products generate the -AUTH command in the worklist to change the authorization IDs for subsequent SQL statements and reBIND commands. You can also manually add -SETS commands to the worklist for setting the authorization IDs.

The -AUTH command that is generated in the worklist cannot switch to 8-byte authorization IDs. Because the -AUTH command simulates the user ID environment inherent to MVS, and MVS itself is limited to seven characters for user IDs, the authorization ID must be seven characters or less.

If your DB2 environment has committed to the use of SET CURRENT SQLID, the best way to ensure that 8-byte IDs are correctly used is to set the installation option for AUTHSW to N. When AUTHSW=N, the products generate -SETS commands in the worklist instead of -AUTH commands. Using -SETS commands exploits DB2’s built-in switching mechanism and the commands are not limited to 7-byte IDs.

If using -SETS commands is not suitable for your environment, you can switch -AUTH commands to 8-byte IDs through the Execution security exit. This exit has many possible uses, including the ability to examine and optionally change the authorization ID that the -AUTH command uses. Through this exit you can cause an 8-byte ID to switch to a corresponding 7-byte (or shorter) ID, which causes a switch in the DB2 DSN3@TH exit that is already installed. In the HLQ.DBSAMP data set, member ALUEUSX1 provides an example of the Execution security exit.

For more information about the Execution security exit, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

If AUTHSW=B, the products generate both -AUTH and -SETS commands in the worklist to change the authorization IDs. The products generate -AUTH commands to set the original CREATEDBY values and to DROP objects. The CREATEDBY field in the DB2 catalog remains unchanged after updates. The product generates -SETS commands to set new OWNER values for all objects. In addition, when AUTHSW=B, the products change the authorization IDs before CREATE TABLE and CREATE INDEX statements.

NOTE
BMC recommends that you specify AUTHSW=N.

NOTE
Use AUTHSW=Y only if your site does not use DB2 secondary authorization IDs.
Controlling authorizations

WARNING
Setting AUTHSW=B is not recommended because of a potential security exposure and the additional complexity if you try to restart the worklist. The security exposure exists because the DB2 catalog does not accurately reflect the primary authorization ID of the creator of the objects. If you must set AUTHSW=B, use the sample security exit (ALUEUSX1 in the HLQ.DBSAMP data set) to avoid the security exposure.

- If AUTHSW=X, the products disable authorization switching for a worklist. The products do not generate -AUTH, -SETS, and -GLID commands in the worklist.

NOTE
Use AUTHSW=X only under the following conditions:

- you are not allowed to execute any -AUTH or -SETS commands in your environment
- you previously edited your worklists and removed all of the -AUTH, -SETS, and -GLID commands

- If AUTHSW=G, the products disable authorization switching for a worklist. The products do not generate -AUTH and -SETS commands; they do, however, generate -GLID commands in the worklist.

For a list of the authorizations that are used for authorization switching, as well as information about -AUTH and -SETS commands, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Using global authorization IDs

CHANGE MANAGER uses the concept of a GLID. The GLID is the default authorization ID for a worklist, rather than the authorization ID of the user who runs the Execution job.

You invoke GLID processing by using the -GLID command. Only one -GLID command is permitted per migrate phase of a worklist. Execution processes the -GLID command as follows:

- performs hash checking
- checks syntax
- checks to see if a -GLID command is already in effect (if so, Execution terminates with an error condition)
- calls the Execution security exit, if it is active (function code X'06')
- determines whether the authorization ID is seven characters or less (if AUTHSW=Y in the installation options module)

---

**NOTE**

The quotation mark characters used as delimiters for delimited identifiers are not considered part of the actual authorization ID.

---

- performs an initial switch by using either the -AUTH or the -SETS command, depending on the setting of the AUTHSW option in the installation options module

After the GLID is set, all -AUTH or -SETS worklist commands with a value of USER set the authorization ID to the GLID. The following rules apply:

- If AUTHSW=Y in the installation options module and an -AUTH command set to USER is used to switch authorizations, the authorization ID is set to the GLID.

- If AUTHSW=B or AUTHSW=N in the installation options module and a -SETS command set to USER is used to switch authorizations, the authorization ID is set to the GLID.

- If AUTHSW=X in the installation options module, the product does not use authorization switching and does not generate -AUTH, -SETS, and -GLID worklist commands.

- If AUTHSW=G in the installation options module, the products do not generate -AUTH and -SETS commands; they do, however, generate -GLID commands in the worklist.

---

**NOTE**

When using a GLID, specify the AUTHSWGLID keyword in the ALUIN input stream.

---

For information about the authorizations that are used when switching authorization for processing DB2 objects, utilities, and commands, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*. 
Processing a worklist

Execution evaluates the parameters that are generated from the Execution interface and, if these parameters are valid, uses them to control the processing of the worklist.

Execution ensures that the work ID exists under the SSID that is specified, and makes connections to DB2 as required to validate or execute a worklist command. The work ID that follows the -WKID command in the worklist must match the Work ID that is specified by the WORKID keyword in the Execution AEXIN input stream, or Execution fails.

Execution also checks the status of the job so that a job is not inadvertently executed twice, or to ensure that the job requests RESTART or STARTOVER if it was previously stopped.

To ensure command integrity, Execution checks the hash values in the worklist and issues messages to flag user-modified commands or inserted commands. You can enable a parameter option or use the Execution security exit to force the end of a run if Execution encounters such changes. Execution ensures that the sequence number of each command is numeric, ascending, and unique.

To accomplish the actions that are specified in the worklist, DB2 and BMC Software utility functions are invoked and monitored for successful completion.

**WARNING**

When you use REORG PLUS in a worklist environment, your work files might not automatically be dynamically allocated, even if you have specified ACTIVE=YES in your installation options module.

The worklist processes SQL statements to build various DB2 objects. Execution runs this code as required and traps SQL errors. If an error results from the execution of the worklist, the Execution component terminates the processing of the worklist and displays the full SQL error text. You can correct the error and restart the job. Table 6 describes Execution return codes.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The worklist successfully executed with no warnings or errors.</td>
</tr>
<tr>
<td>4</td>
<td>The worklist successfully executed, but the product wrote warnings that appear in the diagnostic output file.</td>
</tr>
<tr>
<td>8</td>
<td>The worklist did not execute, and the product wrote errors to the diagnostic output file.</td>
</tr>
<tr>
<td>12</td>
<td>Processing terminated because of a system error, such as a missing data set. No other output is generated.</td>
</tr>
</tbody>
</table>
When Execution must unload and reload data (when, for example, a DB2 object must be dropped and rebuilt), the data is stored in a number of data sets. These data sets are created by several of the BMC Software and IBM utility programs that are invoked by Execution. Whenever data sets are included in a worklist (using the INCLUDE(DATA) keyword to Analysis, for example), unload data sets are created and used by Execution to save and restore data. Because the ddnames for these data sets usually have the form SYSRnnnn (where nnnn is a number), they are sometimes referred to as SYSREC data sets. The names of the data sets are recorded in the diagnostic output file.

If you use ALTER or CHANGE MANAGER with the BMC CATALOG MANAGER product, you can automate log entries in CATALOG MANAGER as follows:

- To automate entries in the CATALOG MANAGER DDL Audit Log for any DDL action, include the CATAUDIT keyword in the AEXIN input stream.
- To automate entries in the CATALOG MANAGER Drop Recovery Log, include the CATRECOVER keyword in the AEXIN input stream.

You can set these keyword defaults in the installation options for ALTER or CHANGE MANAGER.

**Viewing the status of Execution**

To view the progress of Execution, you can select View Execution Status of a WORKID from the WORKID Action Menu panel. Figure 21 shows the Execution Status panel.
Figure 21  Execution Status panel

Table 7 describes the fields in the Execution Status panel.

Table 7  Execution status fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq</td>
<td>sequence number of a sync point or -STOP command (statement that is being executed) Execution uses the sequence number to perform a restart.</td>
</tr>
<tr>
<td>IN</td>
<td>for the Database Administration solution, the number of the XIM initiator in which the work is performed, if the worklist is executed in parallel If the worklist is not executed in parallel, the IN is 0 (zero).</td>
</tr>
<tr>
<td>AUTHID</td>
<td>authorization ID in the sync point or -STOP command</td>
</tr>
<tr>
<td>Status Text</td>
<td>information about the sync point, -STOP command, or XIM initiator</td>
</tr>
</tbody>
</table>

When processing a worklist, Execution updates the status of the work ID to Exec Strt (for Execution started). If errors occur during worklist execution, the status of the work ID remains at Exec Strt until the problems are corrected and the worklist executes successfully. After successfully processing the entire worklist, Execution updates the status of the work ID to Exec Comp (for Execution complete).
You should always determine the extent of processing done by ALTER or CHANGE MANAGER when a work ID has a status of Exec Strt. Actual changes to DB2 data structures could have already been performed.

You can also view the status of the work ID, and the completion date and time of execution by selecting **Browse a WORKID** from the WORKID Action Menu panel.

### Canceling a worklist

If you cancel a worklist that is running in parallel on multiple OS/390 or z/OS images in a data sharing environment, you might also need to inactivate each XIM initiator on each OS/390 or z/OS image. To determine the OS/390 or z/OS images on which the worklist is running, review the diagnostic output file for worklist parallelism (AEXPRINT). For more information about inactivating an XIM initiator, see “Inactivating XIM initiators” on page 736.

### Using sync tables

Execution inserts records into the sync table at various points throughout the processing of a worklist. The records in the sync table for a particular work ID are used as progress indicators during execution of a worklist. In addition, these records are used during restarts to resume execution at the point of a failure. The sync records also contain information about the settings for authorization IDs at the time of failure.

Table 8 describes several of the columns in the sync table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKOWNER</td>
<td>owner of the work ID</td>
</tr>
<tr>
<td>WKNAME</td>
<td>name of the work ID</td>
</tr>
<tr>
<td>SYNCCMD</td>
<td>worklist command that is being executed at the time of the sync</td>
</tr>
<tr>
<td>SYNCSEQ</td>
<td>worklist sequence number</td>
</tr>
<tr>
<td>SYNAUTH</td>
<td>current switched authorization ID (using the -AUTH command), if any</td>
</tr>
<tr>
<td>SYNCSETS</td>
<td>current complete SET CURRENT SQLID command, if any</td>
</tr>
<tr>
<td>SYNCGLID</td>
<td>GLID in effect, if any</td>
</tr>
<tr>
<td>SYNCDATA</td>
<td>variable text describing why the sync was recorded</td>
</tr>
<tr>
<td>SYNCSETP</td>
<td>path for triggers and views</td>
</tr>
</tbody>
</table>
Table 9 describes the conditions under which sync records are written.

Table 9  Conditions for writing sync records

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-AMS command</td>
<td>Execution generates a sync record before any AMS (IDCAMS ALTER, DEFINE, or DELETE) commands.</td>
</tr>
<tr>
<td>-AUTH command</td>
<td>An -AUTH command causes a sync record.</td>
</tr>
<tr>
<td>-BEGU command</td>
<td>For the Database Administration solution, Execution generates a sync record before a UOW is sent to an XIM initiator.</td>
</tr>
<tr>
<td>-SETS command</td>
<td>A -SETS worklist command causes a sync record. SYNCDATA contains the actual SET CURRENT SQLID command text.</td>
</tr>
<tr>
<td>-SETP command</td>
<td>A -SETP worklist command sets a path for triggers and views and executes before the creation of a trigger or a view.</td>
</tr>
<tr>
<td>-SQL DROP statements</td>
<td>Execution generates a sync record before any DB2 object is dropped.</td>
</tr>
<tr>
<td>-SYNC command</td>
<td>An explicit sync request causes the product to write the text on the -SYNC worklist command to column SYNCDATA along with the sequence number. -SYNC also causes a commit. The name of an object in the -SYNC command might be truncated if the name contains up to 128 characters. To determine the full name of the object, review the worklist command that precedes the -SYNC command in the worklist.</td>
</tr>
<tr>
<td>-STOP command</td>
<td>A -STOP command is identical to a -SYNC command except that execution of the worklist is halted with return code 4.</td>
</tr>
<tr>
<td>-UOWC command</td>
<td>For the Database Administration solution, Execution changes the BEGU command to a UOWC command when a UOW completes in an XIM initiator.</td>
</tr>
<tr>
<td>end of job</td>
<td>The product writes an end of job (EOJ) sync record at the end of the worklist. The EOJ sync record has a SYNCCMD of DONE and SYNCSEQ 999999.</td>
</tr>
<tr>
<td>error</td>
<td>Any worklist error, such as an SQL error or utility failure, will cause an error sync record.</td>
</tr>
<tr>
<td></td>
<td>You can recognize error sync records by the characters EMSG in column SYNCCMD.</td>
</tr>
</tbody>
</table>

You can delete all sync records in the sync table for a work ID at the end of a successful (no errors) Execution run by specifying the SYNCDELETE keyword in the AEXIN input stream. The sync tables require ongoing maintenance. For more information about maintaining sync tables, see the **ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1**.
Reviewing the results of a migration

The primary output from Execution is the modified data structures that are stored in the DB2 catalog and any cataloged output from the utilities such as image copy data sets. However, the exact nature of these modifications depends on the content of the worklist that is being executed.

In addition to the DB2 modifications, Execution produces a diagnostic output file. The diagnostic output file, also called a worklog or SYSOUT, is a log or audit trail of all input statements shown in their entirety before execution. The worklog also contains the responses that are caused by those actions. It is the primary tool for tracking events during execution. The worklog ddname is AEXPRINT.

When Execution runs component programs, it captures the SYSPRINT outputs from those programs in the log. Lines beginning with a colon (:) echo the input worklist commands to the log. All other lines are generated by Execution.

Output for parallel worklists

When a worklist is run in parallel in the Database Administration solution, Execution produces the output files listed in Table 10. The table lists the files in the order in which you should review them.

Table 10  Output files for worklist parallelism (part 1 of 4)

<table>
<thead>
<tr>
<th>#</th>
<th>Output file</th>
<th>ddname</th>
<th>Description and use</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>diagnostic output file</td>
<td>AEXPRINT</td>
<td>provides a summary of the results of execution for all UOWs that are running in an XIM initiator and the status of the work ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For each UOW that is sent to an XIM initiator, the file provides the completion code and a time stamp. Because XIM initiators start and complete at various times, the sequence of the output for UOWs might not match the sequence of the commands in the worklist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: This file also provides detailed output for UOWs that have been restarted and are being processed serially.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use this file to perform the following tasks:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ Verify that execution starts the initiators, sends each UOW to an initiator, completes the UOWs, and ends the initiators.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ Review Execution return codes and determine why a job has failed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Messages that are prefixed with three asterisks (***)) indicate the UOWs that failed. The messages refer to the appropriate output files to review, such as AEXPRnnn or AEXSYnnn.</td>
</tr>
</tbody>
</table>
Reviewing the results of a migration

AEXSTLOG contains summary information about the status of each command in every UOW. The log contains the following columns:

- **TIME**—time at which Execution began
- **SEQ**—sequence number for the worklist command
- **CMND**—name of the worklist command
- **INIT**—number of the XIM initiator that is processing a UOW
- **SYID**—subsystem ID

You can use this file to perform the following tasks:

- Monitor the status of a worklist.
- Monitor the status of each XIM initiator.
- **INITS**—the numbers of the active XIM initiators

If one of the following conditions exists, a hyphen (-) replaces the number of the initiator:

- an initiator is not active at the specified time
- a UOW is pending and a response has not yet been received from an initiator

You can monitor the status of each XIM initiator by reviewing the INITS column and the messages at the end of the log.

- **COMMAND TEXT**—portion of the text for the worklist statement

You can manually insert the STATUSINFO NONE keyword in the AEXPIN input stream to eliminate this log and all of the AEXSTnnn logs.

### Table 10  Output files for worklist parallelism (part 2 of 4)

<table>
<thead>
<tr>
<th>#</th>
<th>Output file</th>
<th>ddname</th>
<th>Description and use</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>worklist command status log</td>
<td>AEXSTLOG</td>
<td>contains summary information about the status of each command in every UOW</td>
</tr>
</tbody>
</table>

The log contains the following columns:

- **TIME**—time at which Execution began

Because each XIM initiator starts and completes at various times, the sequence of the output for UOWs might not match the sequence of the commands in the worklist.

- **SEQ**—sequence number for the worklist command
- **CMND**—name of the worklist command
- **INIT**—number of the XIM initiator that is processing a UOW
- **SYID**—subsystem ID

You can use this file to perform the following tasks:

- Monitor the status of a worklist.
- Monitor the status of each XIM initiator.
- **INITS**—the numbers of the active XIM initiators

If one of the following conditions exists, a hyphen (-) replaces the number of the initiator:

- an initiator is not active at the specified time
- a UOW is pending and a response has not yet been received from an initiator

You can monitor the status of each XIM initiator by reviewing the INITS column and the messages at the end of the log.

- **COMMAND TEXT**—portion of the text for the worklist statement

You can manually insert the STATUSINFO NONE keyword in the AEXPIN input stream to eliminate this log and all of the AEXSTnnn logs.
### Table 10  Output files for worklist parallelism (part 3 of 4)

<table>
<thead>
<tr>
<th>#</th>
<th>Output file</th>
<th>ddname</th>
<th>Description and use</th>
</tr>
</thead>
</table>
| 3 | worklist execution log for XIM initiators        | AEXPRnnn | contains worklist commands, worklist output, and diagnostic output for an instance of Execution that is running in an XIM initiator.  
One AEXPRnnn file is created for each XIM initiator. The first file, AEXPR001, also contains output for sections of the worklist that are not processed in parallel, such as DROP, CREATE, and ALTER statements.  
**Note:** If UOWs are restarted, Execution processes the UOWs serially and writes the output to AEXPRINT.  
You can use this file to view the execution of the commands, SQL, and utilities for each UOW. |
| 4 | worklist command status log for XIM initiators   | AEXSTnnn | contains detailed information about each command in an XIM initiator.  
For each XIM initiator, the log lists the entire command and the time that the execution of the command began. The first file, AEXST001, also contains output for sections of the worklist that are not processed in parallel, such as DROP, CREATE, and ALTER statements.  
You can manually insert the STATUSINFO NONE keyword in the AEXPIN input stream to eliminate this log and the AEXSTLOG.  
You can use this file to monitor the execution of commands, SQL, and utilities for a specific XIM initiator. |
| 5 | system messages and job log for XIM initiators   | AEXSYnnn | contains system messages and job information for each XIM initiator.  
You can use this file to view the allocation of data sets. |
| 6 | tracing log                                      | AEXPTRAC | contains tracing records for the worklist  
The tracing log is optional, depending on whether you selected to display tracing messages during worklist execution.  
Provide this log to BMC Customer Support to diagnose a problem. |
| 7 | tracing log for XIM initiators                   | UOWTRnnn | contains tracing records for each XIM initiator  
The tracing log for XIM initiators is optional; it is dependent upon whether you selected to display tracing messages during the execution of the worklist.  
Provide this log to BMC Customer Support to diagnose a problem. |
Restart methods

If an error is returned during the initial processing of a worklist, inspect the diagnostic output to determine which job step caused the error. Depending on the error, you might need to restart the worklist.

The Execution component uses one of the following methods for restarting a worklist:

- **Restart processing** resumes execution of the worklist after a failure occurs.

  The RESTART keyword in the AEXIN input stream indicates that Execution uses restart processing. Typically, you use restart processing after a utility worklist command fails. For more information, see “Restart processing” on page 107.

  **TIP**
  
  To restart a worklist, see “Restarting a worklist” on page 164.

- **Startover processing** completely reruns a worklist from the beginning of the worklist, or from the beginning of the second phase of a two-phase worklist after a failure occurs.

  The STARTOVER keyword in the AEXIN input stream indicates that Execution uses startover processing. Use startover processing after generating a subsequent worklist or multiple worklists for the same work ID, or when you have extensively edited or altered a worklist and cannot restart it. For more information, see “Startover processing” on page 111.

  **WARNING**
  
  BMC recommends that you use restart processing to restart a worklist.

---

### Table 10  Output files for worklist parallelism (part 4 of 4)

<table>
<thead>
<tr>
<th>#</th>
<th>Output file</th>
<th>ddname</th>
<th>Description and use</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>sort processing log</td>
<td>SYSnnmn</td>
<td>for restart processing, provides information about sort processing that is performed by the BMC BMCSORT technology</td>
</tr>
</tbody>
</table>

---

*a* The variable *nnn* represents the number of the initiator. The DD or output file for each initiator is dynamically allocated, based on the number of initiators that are used.

*b* The variable *nnnnn* represents the number of the message DDN.
Restart methods

Chapter 1 Migrating database environments

Restart processing

When restarting a worklist, Execution performs the following steps:

1. checks the status of the work ID, which must be Exec Strt (execution started)

2. searches the sync table backwards to find the last valid sync entry for the work ID

3. reads the worklist, searching for the restart sequence number, and performs the following tasks:
   - validates the subsystem ID (-SSID command) upon encountering it
   - validates the work ID (-WKID command) upon encountering it
   - if the failing command was a utility worklist command, follows the procedures described in “How the RESTARTPARM keyword affects restarting utilities” on page 108

4. reestablishes values for the runtime environment from the sync table, if the commands listed in Table 11 were included in the worklist

Table 11 Reestablished sync values

<table>
<thead>
<tr>
<th>Worklist command</th>
<th>Reestablished value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-GLID</td>
<td>global authorization ID</td>
</tr>
<tr>
<td>-AUTH</td>
<td>authorization ID</td>
</tr>
<tr>
<td>-SETS</td>
<td>current SQL ID</td>
</tr>
<tr>
<td>-SETP</td>
<td>PATH</td>
</tr>
</tbody>
</table>

NOTE
Because authorization data is stored in the products’ sync tables, you cannot change the authorization data for a restart.

5. continues to process the worklist from the last sync point or from the -STOP worklist command

TIP
To start a worklist over, see “Starting a worklist over” on page 166.
JCL Generation determines the sequence number of the command that failed and the disposition of the data set:

- Any tape data set that was opened before the sequence number of the command that failed will have a disposition of SHR or (OLD,KEEP,KEEP).

- Any tape data set that would have been opened at the sequence number of the command that failed or after that sequence number will have a disposition of (NEW,CATLG,CATLG).

- Any DASD data sets will have a disposition of SHR.

\section*{NOTE}
Verify that the data set has not been allocated. If the data set has been allocated, modify the disposition of the data set.

\section*{WARNING}
Do not use a disposition of MOD (DISP=MOD), which could cause unpredictable results. Do not change the disposition value for the SYSIN, SYSTSIN, SYSPRINT, SYSTSPRT, and ALUIN ddnames.

To restart a worklist, select the Build Restart JCL from previous execution JCL option on the Execution JCL Build Interface panel. When you select this option, the products insert the RESTART keyword into the AEXIN input stream.

For additional considerations that apply to restart processing, see “Considerations for restarting or starting over a worklist” on page 112.

\section*{How the RESTARTPARM keyword affects restarting utilities}

To restart a utility in a worklist, you can either omit or include the RESTARTPARM keyword in the AEXIN input stream.
Omitting the RESTARTPARM keyword

If you do not specify the RESTARTPARM keyword in the AEXIN input stream, the utility reruns or restarts with the parameters listed in Table 12.

Table 12 Restarting utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC CHECK PLUS</td>
<td>Rerun</td>
<td>SSID, UTILID, NEW</td>
</tr>
<tr>
<td>BMC BASIC UNLOAD</td>
<td>Rerun</td>
<td>SSID, UTILID, NEW/RESTART</td>
</tr>
<tr>
<td>BMC COPY PLUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC LOAD PLUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC RECOVER PLUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC UNLOAD PLUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMC DASD MANAGER PLUS components: BMCCPRS, BMCSTATS, BMCTRIG, or BMCUPRS</td>
<td>Rerun</td>
<td>SSID, VCATname, ASUDOPTSname, WORKID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For information about parameters, see the DASD MANAGER PLUS for DB2 documentation.</td>
</tr>
<tr>
<td>BMC REORG PLUS</td>
<td>Restart</td>
<td>SSID, UTILID, NEW/RESTART</td>
</tr>
<tr>
<td>IBM LOAD or other IBM utilities, except IBM REORG</td>
<td>Rerun</td>
<td>SSID, UTILID, lineOneParms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The variable lineOneParms represents positional parameters that are allowed on the utility EXEC statement for restarting the utility. The parameters follow the first line of the utility command in the worklist. For information about necessary parameters, see the manual for the IBM utility that you want to restart.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For IBM LOAD, Execution issues commands to terminate the execution of the utility. Also, if necessary, Execution issues commands to start table spaces, start index spaces, and delete table rows before rerunning the utility. The worklist contains a -LOAD command and, on the same line, a DELETE FROM clause. The next line of the command contains the LOAD RESUME YES option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For other IBM utilities, except IBM REORG, Execution issues only the command to terminate execution of the utility before rerunning the utility.</td>
</tr>
<tr>
<td>IBM REORG</td>
<td>Restart</td>
<td>SSID, UTILID, RESTART</td>
</tr>
</tbody>
</table>
Including the RESTARTPARM keyword

If you specify the RESTARTPARM keyword in the AEXIN input stream, the utility restarts with the following parameters:

- SSID, which is the subsystem ID as specified in the worklist
- UTILID, which is the utility ID for the job
- restartParmString, which is the parameter string specified with the RESTARTPARM keyword

The variable restartParmString contains positional parameters that are allowed on the utility EXEC statement for restarting. For information about necessary parameters, see the manual for the utility that you want to restart.

In addition, Execution does not perform a cleanup for the utilities (with the exception of the IBM LOAD utility). For example, no commands are issued to terminate the execution of the utility, delete table rows, start table spaces, or start index spaces.

When you specify the RESTARTPARM keyword, use the following guidelines:

- Ensure that the RESTART keyword is also specified in the AEXIN input stream.
- Do not enter the SSID and UTILID parameters as part of the restartParmString parameter. Execution passes the SSID and UTILID parameters for you.

**NOTE**
Do not use positional commas for placeholders of the SSID and UTILID parameters.

- Do not enclose a restartParmString parameter with single quotation marks. This action causes an error.
- Do not specify the RESTARTPARM keyword without the restartParmString parameter. If you do so, Execution ignores the keyword.

The following example shows the AEXIN keywords to restart the REORG PLUS utility for a particular job:

```plaintext
RESTART
RESTARTPARM NEW/RESTART.,MSGLEVEL(1)
```
Startover processing

**WARNING**
During startover processing, Execution makes no provisions for any reversal or recovery of processes that might have been performed already.

Execution performs the following steps when it starts over a worklist:

1. checks the status of the work ID
2. executes the worklist from the beginning and establishes the runtime environment from the values in the worklist
3. starts the first BMC utility in the worklist with the \texttt{SSID,UTILID,NEW/RESTART} parameters

**NOTE**
Execution starts the COPY PLUS utility with the \texttt{SSID,UTILID,NEW/RESET} parameters, and the CHECK PLUS utility with the \texttt{SSID,UTILID,NEW} parameters.

**NOTE**
If the BMC or IBM utility encounters an existing UTILID or a UTILID that is associated with another utility on this object, the utility might terminate with an error. For example, if the first utility is REORG PLUS but the existing UTILID is associated with LOADPLUS, REORG PLUS might terminate with an error. You might need to terminate the utility or allow the utility to complete.

To start a worklist over, select the **Build Startover JCL from previous execution JCL** option on the Execution JCL Build Interface panel. When you select this option, the products insert the \texttt{STARTOVER} keyword into the AEXIN input stream.

If the worklist has been executed once and the DASD data sets exist, JCL Generation creates DD statements with a disposition of SHR. If the worklist has not been executed and the data sets do not exist, you must ensure that the DD statements are created with a disposition of (NEW,CATLG,CATLG). To do so, select **Build Initial JCL** from the Execution JCL Build Interface panel. Then, manually add the \texttt{STARTOVER} keyword to the AEXIN input stream.

If you want to start a worklist over at a new beginning, you can edit the worklist to remove unwanted commands and then select the **Build Start Over JCL from previous execution JCL** option to execute the modified worklist.
For additional considerations that apply to startover processing, see “Considerations for restarting or starting over a worklist” on page 112.

## Considerations for restarting or starting over a worklist

When you restart or start over a worklist, consider the following items:

- If the copy data sets were dynamically allocated and GDGs were specified for the data sets, you do not need to delete the data sets before starting the worklist over. However, if you did not specify GDGs for the copy data sets, you must change the name of the dynamically allocated data set in each of the -BMCC worklist commands. Then, you can start the worklist over.

- When you enable worklist parallelism for the Database Administration solution and run your job in parallel, use the following guidelines:
  - To build restart JCL, select **Build Restart JCL from previous execution JCL** on the Execution JCL Build Interface panel.
  - To build startover JCL, select **Build Startover JCL from previous execution JCL** on the Execution JCL Build Interface panel.
  - Do not increase the maximum number of initiators in the MAXINITS keyword in the AEXPIN input stream.
  - If you initially execute a worklist with parallelism enabled, you can restart the worklist with parallelism disabled. However, if you initially execute a worklist with parallelism **disabled**, you cannot restart the worklist with parallelism **enabled**.

An exception exists if you are using a receive-type work ID on a receiving subsystem to execute phase 2 of a migrate-type worklist. In this case, you can execute phase 1 of the worklist with parallelism disabled and then restart phase 2 of the worklist with parallelism enabled.

- Execution stores the names of the data sets that are dynamically allocated (with a suffix of .UOWnn, where nn is the number of the initiator) in the data set name allocation (DSNA) records in the sync table. Execution uses these records when it restarts the commands in a UOW in serial mode.
— When you restart a worklist, Execution completes the UOWs that ended abnormally in serial mode, not in parallel mode. The diagnostic output file (AEXPRINT) lists all of the commands in the worklist that Execution restarted.

— The worklist commands between the -BEGG and -ENDG commands are always executed when Execution restarts a worklist.

— If you build restart or startover JCL, the preallocation step and the data set DD statements are commented out. If you need to create the data sets, remove the comments.

— If you specify TAPE instead of SYSDA for Error (SYSERnnn) or Map (SYSMAP) data sets and you need to restart the IBM LOAD utility, you must uncatalog the existing data set. Then, you must change the DD statements to a disposition of (NEW,CATLG,CATLG).

Accomplishing your goals

You can use ALTER and CHANGE MANAGER to create data structures on a subsystem from structures that already exist. You can use scripts in the CM/PILOT component of CHANGE MANAGER to migrate data structures. (For more information about scripts, see Appendix A, “Using scripting tools to automate change management.”) Table 13 lists the tasks that you can perform to migrate data structures and data.

Table 13 Migration tasks (part 1 of 2)

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifying objects for migration</td>
<td></td>
</tr>
<tr>
<td>“Creating a migrate-type work ID”</td>
<td>page 114</td>
</tr>
<tr>
<td>“Creating a migrate-type work ID in batch mode”</td>
<td>page 115</td>
</tr>
<tr>
<td>“Defining migrate options for a migrate-type work ID”</td>
<td>page 116</td>
</tr>
<tr>
<td>“Creating change rules for a migrate-type work ID”</td>
<td>page 117</td>
</tr>
<tr>
<td>“Converting an external SQL procedure to a native stored procedure”</td>
<td>page 118</td>
</tr>
<tr>
<td>“Specifying the data structures in a migrate-type work ID”</td>
<td>page 119</td>
</tr>
<tr>
<td>“Migrating the auxiliary objects for a LOB column”</td>
<td>page 121</td>
</tr>
<tr>
<td>“Migrating data structures by using DML”</td>
<td>page 123</td>
</tr>
<tr>
<td>“Assigning a unique name by importing DML”</td>
<td>page 128</td>
</tr>
<tr>
<td>“Defining primary and secondary quantities by importing DML”</td>
<td>page 129</td>
</tr>
<tr>
<td>“Setting a secondary quantity to a factor of a primary quantity by importing DML”</td>
<td>page 131</td>
</tr>
</tbody>
</table>
Creating a migrate-type work ID

You can create a migrate-type work ID to migrate data structures. A migrate-type work ID can contain migrate options and change rules.

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the **WORKID** and select **Create a new WORKID**. Then, press Enter.

   The Create WORKID panel is displayed.

3. Specify the information for the work ID.
Creating a migrate-type work ID in batch mode

A Select Migrate for the Type.

B (optional) Specify a Comment to describe the work ID.

C To specify migrate options for the work ID, select WORKID Migrate Options. For more information about specifying migrate options, see “Defining migrate options for a migrate-type work ID” on page 116.

D To specify change rules for the work ID, select WORKID Change Rules. For more information about specifying change rules, see “Creating change rules for a migrate-type work ID” on page 117.

E Press Enter.

4 Press END.

The WORKID Action Menu is displayed.

Where to go from here

After you create a migrate-type work ID, perform the tasks in “Specifying the data structures in a migrate-type work ID” on page 119.

Creating a migrate-type work ID in batch mode

ALTER and CHANGE MANAGER enable you to automate batch processes by creating new work IDs in batch. You create these work IDs by editing the JCL that ALTER or CHANGE MANAGER created outside of the products.

1 (optional) Create an outbound migrate profile with scope rules (page 146 and page 148).

2 In ISPF, edit the Analysis JCL as follows:

A In the ALUIN input stream, change the WORKID keyword to NEWWORKID and specify the name of the new work ID that you want to create.

B Ensure that the MIGPROFILE and the MIGSCOPE keywords are included in the ALUIN input stream.

3 To create a worklist, submit the JCL (page 153).
Defining migrate options for a migrate-type work ID

You can specify migrate options for an existing migrate-type work ID. The migrate options define the default settings for the dependent objects that will be migrated.

**TIP**
To create a migrate-type work ID, see page 114.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.
   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select List and press Enter.
   The WORKID Selection List is displayed.

3. Type E in the Act column adjacent to the work ID that you want to use. Then, press Enter.
   The Edit WORKID panel is displayed.

4. Type S to select WORKID Migrate Options and press Enter.
   The Edit WORKID Migration Options panel is displayed.

5. Specify the options for the migration.

   A Type the name of the subsystem to which you want to migrate your data structures, data, or both. Optionally, you can type an asterisk (*), which allows phase 2 of the worklist to execute on any subsystem.

   B Type Y adjacent to the dependent objects that you want to migrate.

   C Press Enter.

6. Press END until the WORKID Action Menu is displayed.
Creating change rules for a migrate-type work ID

You can create change rules for an existing migrate-type work ID. Change rules are used to modify the attributes of data structures.

**TIP**
To create a migrate-type work ID, see page 114.

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.

   The WORKID Action Menu is displayed.

2. Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **List** and press **Enter**.

   The WORKID Selection List is displayed.

3. Type **E** in the **Act** column adjacent to the work ID that you want to use. Then, press **Enter**.

   The Edit WORKID panel is displayed.

4. Type **S** to select **WORKID Change Rules** and press **Enter**.

   The Change Rules panel is displayed.

5. Type **C** in the **Act** column and press **Enter**.

   The Edit Change Rules panel is displayed.

6. Specify the information for the change rule.

   **A** Select **Change** for the type of rule.

   **B** Select the type of data structure to which you want the change rule to apply.

   **C** Type the name of the attribute of the data structure.

   **NOTE**
   To display a list of the valid attributes, type **S** to select **Display Attribute List**. The Edit Rules Attribute Selection panel is displayed. Type the number for the attribute that you want to use and press **Enter**. Then press **END**. The Edit Change Rules panel is displayed again, with the name of the **Attribute** displayed.
Converting an external SQL procedure to a native stored procedure

You can use change rules to convert an external SQL procedure to a native stored procedure.

**TIP**
To create a migrate-type work ID, see page 114.

1. On the CHANGE MANAGER Main Menu, select WORKID and press Enter.
   
The WORKID Action Menu is displayed.

2. Type the name of the WORKID, or type a wildcard pattern to display a list of work IDs. Then, select List and press Enter.

3. In the WORKID Selection List, type E in the Act column adjacent to the work ID that you want to use. Then, press Enter.

4. In the Edit WORKID panel, type S to select WORKID Change Rules and press Enter.
5 On the Change Rules panel, type C in the Act column and press Enter.

6 On the Edit Change Rules panel, specify the information for the change rule:

A In the Option field, select Change.

B In the Object field, select Stored Procs.

C In the Attribute field, type SPType.

D In the Current Attribute Value field, type * or leave the field blank.

E In the New Attribute Value field, type NATIVE.

7 Press END until the WORKID Action Menu is displayed.

---

Specifying the data structures in a migrate-type work ID

You can specify the data structures for a migration in an existing migrate-type work ID.

---

To create a migrate-type work ID, see page 114.
5 Type the letter of an action in the Act column to create, edit, drop, or migrate an object or to override the default dependencies. For a description of the action codes, see Table 1 on page 44.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

An asterisk (*) and a change-level indicator are displayed adjacent to the data structures that you selected for migration.

6 Press END.

The Object Specification panel is displayed.

7 To view a list of the data structures that have changed, type S to select Changed Objects List.

The Changed Objects List is displayed.

8 On the Changed Objects List panel, you can perform the same actions as in the Mixed List panel. Then, press END.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

The Object Specification panel is displayed.

9 Press END.

The WORKID Action Menu is displayed.

**Where to go from here**

After you specify the data structures to migrate, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Migrating the auxiliary objects for a LOB column

You can specify to migrate a LOB base table (a table that contains a LOB column) with its auxiliary objects in an existing migrate-type work ID. The auxiliary objects include a LOB table space, auxiliary table, and an index on the auxiliary table.

--- TIP ---
To create a migrate-type work ID, see page 114.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name (or use wildcard characters) of a base table. Alternatively, you could specify a database or a table space that contains a base table. Press Enter.

   The Mixed List panel is displayed.

4. Type MO in the Act column adjacent to the object that you want to migrate.

   The Migrate Options Overrides panel is displayed.

5. Type Y adjacent to Migrate Auxiliary Objects. If you specified the name of a database or a table space in step 3, type Y adjacent to Migrate Tables also. Press Enter.

6. Press END.

   The Mixed List panel is displayed. An asterisk (*) and a change-level indicator are displayed adjacent to the data structures that you selected for migration.

7. To view the auxiliary objects, type AX in the Act column adjacent to the base table object (TBB) that you selected to migrate.
The Auxiliary Objects List panel is displayed. This panel shows the auxiliary objects for the base table (see Figure 22).

Figure 22  Auxiliary Objects List panel

<table>
<thead>
<tr>
<th>Act</th>
<th>Object-Type</th>
<th>Objects</th>
<th>Column</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>CJACML01</td>
<td>L01PSL1A</td>
<td>XCLOB</td>
<td>1</td>
</tr>
<tr>
<td>TB</td>
<td>RDACRJ</td>
<td>T_L01PSL1A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>RDACRJ</td>
<td>L01PSL1A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>CJACML01</td>
<td>L01PSL1B</td>
<td>XCLOB</td>
<td>2</td>
</tr>
<tr>
<td>TB</td>
<td>RDACRJ</td>
<td>T_L01PSL1B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>RDACRJ</td>
<td>L01PSL1B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>CJACML01</td>
<td>L01PSL1C</td>
<td>XCLOB</td>
<td>3</td>
</tr>
<tr>
<td>TB</td>
<td>RDACRJ</td>
<td>T_L01PSL1C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>RDACRJ</td>
<td>L01PSL1C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8  Press END until the WORKID Action Menu is displayed.

Where to go from here

After you specify the data structures to migrate, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Migrating data structures by using DML

The CM/PILOT DML_MIGRATE script contains the steps to create an Analysis worklist to migrate data structures from a local subsystem. You can use DML to define the data structures to migrate. The script uses a DML trigger to determine whether data structures exist. The DML trigger is the SEARCH condition of a DML WHERE clause. If no data structures are found, the CM/PILOT worklist processing stops, and an Analysis worklist is not created. If data structures are found, an Analysis worklist is created to migrate the data structures.

In this task, you will

1. create a task ID
2. create a DML statement
3. execute the task ID

Before you begin

If you want to use change rules or locations, use CHANGE MANAGER to create an outbound migrate profile.

--- TIP ---

To create an outbound migrate profile, see page 146.

--- To create a task ID ---

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3. Type the name of a new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

4. Select Migrate data structures using a DML trigger and press Enter.

   The Create TASKID panel is displayed.
Migrating data structures by using DML

5 Specify the information for a task ID.

A CM/PILOT uses the name of the task ID as the name for a new migrate-type work ID. You can type a different name for WORKID or type the name of a work ID name template.

B (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

C (optional) Specify a Comment to describe the task ID.

D Type S to select Create or Edit DML.

E Press Enter.

The Select Object and Action for Migrate DML panel is displayed.

To create the DML statement

1 Specify the information to create a DML statement.

A Select an Object type to be included in a DML statement.

B Select an Action.

NOTE
If you select Select Editor, you cannot specify an Object. The Select Editor action enables you to enter the DML in an ISPF editor without first typing values for the SET and WHERE clause in a Migrate Object DML panel.

C Press Enter.

The panel that is displayed is based on the object that you select.

D Type a value for an attribute for the SET clause. Then, type an operator and a value for the WHERE clause. Press END.

The DML statement is displayed in the ISPF Editor, as shown in the example in Figure 23.

Figure 23  DML for a MIGRATE statement (part 1 of 2)

<table>
<thead>
<tr>
<th>MIGRATE DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCLUDE DATA, TABLES, FOREIGNKEYS, INDEXES, VIEWS,</td>
</tr>
<tr>
<td>TRIGGERS, TABLESPACES, AUTHORIZATIONS, SYNONYMS, ALIASES,</td>
</tr>
<tr>
<td>CHECKS, CONSTRAINTS, AUXILIARY</td>
</tr>
</tbody>
</table>
2 Edit your DML statement in the ISPF Editor, if necessary. For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

3 When you have finished editing the statement, press END.

The List DML Statements panel is displayed.

4 You can perform the following actions by typing the action code adjacent to the DML statement:

- **A** (add a new statement) displays the Select Object and Action for Migrate DML panel, so that you can create another DML statement. The new statement is placed at the end of the list.

- **B** (browse) displays the ISPF Editor and places you in browse mode. After you browse the statement, press END.

- **C** (copy) copies the DML statement and displays the ISPF Editor. After you modify the statement, press END. The copied and modified statement is located at the end of the list.

- **D** (delete) deletes a DML statement. The statement is immediately deleted. No delete confirmation panel is displayed.

- **E** (edit) displays the ISPF Editor.

---

**NOTE**

Syntax checking is performed on the values that you enter in the input fields of a DML panel. No syntax checking is performed when you edit DML statements.
5 When you have finished modifying the DML statements, press END until the TASKID Action Menu is displayed.

The task ID containing the DML statements for the script is created.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.
   
   A Type the data set name for the CM/PILOT Worklist.
   
   B Type the data set name for the CM/PILOT JCL.
   
   C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

   A Type S to select Override CHANGE MANAGER options and then press Enter.

   The Override CHANGE MANAGER Options panel is displayed.

   B Select Override CHANGE MANAGER Options.

   C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

   For more information about overriding the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

---

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.
5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select the steps in the script that you want to modify, and press Enter.

---

**NOTE**

You can use a CHANGE MANAGER outbound migrate profile to define change rules or specify multiple locations for Analysis worklists that are created using this script. To specify an outbound migrate profile, change the Analysis step. You can specify the profile in an application or on the TASKID Migrate Profiles panel that is displayed. Changes that you specify using DML will take precedence over change rules in an outbound migrate profile.

C To display subsequent Step Settings panels, press Enter.

After the Step Settings panels are displayed, the CHANGE MANAGER Datasets panel is displayed.

D Specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press END until the TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

7 To migrate your data structures, execute the Analysis worklist.

---

**NOTE**

The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.
Assigning a unique name by importing DML

You can import DML to migrate a database and assign a unique name to the database and its related table spaces.

**NOTE**
You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1. In an ISPF editor, type the DML statements to match those shown in Figure 24, and provide the information that is unique to your site.

   For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**Figure 24** DML for assigning a unique name

```
MIGRATE DATABASE
  INCLUDE ALL
  SET
    NAME = 'DBM' CONCAT
      SUBSTR(CHAR(CURRENT DATE, IS0), 6, 2)
    CONCAT 'D' CONCAT
      SUBSTR(CHAR(CURRENT DATE, IS0), 9, 2)
  WHERE
    NAME = 'ALUQATBS'
;
MIGRATE TABLESPACE 'ALUQATBS'
  INCLUDE ALL
  SET
    NAME = 'TSM' CONCAT
      SUBSTR(CHAR(CURRENT DATE, IS0), 6, 2)
    CONCAT
      SUBSTR(NAME, 6, 3)
  WHERE
    DBNAME = 'ALUQATBS'
;```

**TIP**
To execute a worklist, see page 158.
The sample DDL in Figure 24 includes the month and day of the month in the database and table space names. It has been created using concatenation, substrings, and CURRENT DATE. The name of the database will be DB{mm}{dd}, where {mm} is the month, and {dd} is the day of the month. Table spaces resolve to TSM{mm}{nnn}, where {mm} is the month and {nnn} are the last three characters of the table space name.

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>

Defining primary and secondary quantities by importing DML

You can import DML to migrate a table space and indexes that are specified in a user-defined table. The primary and secondary quantities for the migrated table space are set using the values in the user-defined table. This task is particularly useful for migrating objects from a development environment to a test environment to a production environment on a periodic basis when change rules need to be specified in a user-defined table.

NOTE

You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1 In an ISPF editor, type the DML statements to match those shown in Figure 25, and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.
The DML in Figure 25 migrates the table space partition using information from the user-defined table that identifies the primary and secondary quantities, and the database and table spaces that are affected. The index partition is migrated by using the same method.

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Setting a secondary quantity to a factor of a primary quantity by importing DML

You can import DML to set a secondary quantity to a factor of a primary quantity for all STOGROUP-defined table spaces and indexes in a database.

NOTE
You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1 In an ISPF editor, type one of the DML statements shown in Figure 26, and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

NOTE
PRIQTY and SECQTY are expressed in 4K pages.

Figure 26 Example DML for setting SQTY (part 1 of 2)

```
MIGRATE TABLEPART
SET SQTY = PQTY / 2 -- set secondary quantity to half of primary quantity
WHERE
  DBNAME LIKE 'CRJ%'
  AND (PQTY BETWEEN 4 AND 60) -- pages, not K
  AND STORTYPE = 'I' -- STOGROUP-defined;

MIGRATE TABLEPART
SET SQTY = 2 * PQTY -- set secondary quantity to twice primary quantity
WHERE
  DBNAME LIKE 'CRJ%'
  AND PQTY < 4 -- pages, not K
  AND STORTYPE = 'I' -- STOGROUP-defined;

MIGRATE INDEXPART
JOIN SYSIBM.SYSINDEXES IX
SET SQTY = PQTY * 1.5 -- set secondary quantity to 1 1/2 of primary quantity
WHERE
  STORTYPE = 'I'
  AND IXNAME = IX.NAME
  AND IXCREATOR = IX.CREATOR
```
Changing table spaces and indexes to DEFINE NO by importing DML

Figure 26  Example DML for setting SQTY (part 2 of 2)

AND IX.DBNAME LIKE 'CRJ%'

2  Import the DML file.

For information, see "Importing a CDL, DDL, or DML file in CHANGE MANAGER" on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>&quot;Analyzing a migrate-type work ID and generating a worklist&quot; on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>&quot;Executing a migrate-type worklist&quot; on page 158</td>
</tr>
</tbody>
</table>

Changing table spaces and indexes to DEFINE NO by importing DML

You can import DML to migrate one or more table spaces and their dependent indexes and to change the table spaces and indexes to DEFINE NO. This task is especially useful to DBAs who manage an ERP application and want to limit creation of empty data sets.

NOTE

You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1  In an ISPF editor, type the DML statements to match those shown in Figure 27, and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.
Changing table spaces and indexes to DEFINE NO by importing DML

**TIP**
You can copy the DML statements to perform this task from member ACMDMLM1 in the HLQ.DBCNTL data set.

**Figure 27  DML to migrate table spaces and indexes**

<table>
<thead>
<tr>
<th>MIGRATE TABLESPACE</th>
<th>INCLUDE DATA, TABLES, FOREIGNKEYS, INDEXES, VIEWS, AUTHORIZATIONS, SYNONYMS, ALIASES, CHECKS, TRIGGERS</th>
<th>SET</th>
<th>DEFINE = 'N'</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE</td>
<td>NAME = 'TSNAME' --specify existing TS name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND DBNAME = 'DBNAME' --specify existing DB name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIGRATE INDEXES</td>
<td>JOIN SYSIBM.SYSTABLES TB</td>
<td>SET</td>
<td>DEFINE = 'N'</td>
</tr>
<tr>
<td></td>
<td>WHERE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSINDEXES.TBNAME = TB.NAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND SYSIBM.SYSINDEXES.TBCREATOR = TB.CREATOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND TB.DBNAME = 'DBNAME' --specify existing DB name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND TB.TSNAME = 'TSNAME' --specify existing TS name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Migrating a partitioned table space to a range-partitioned table space by importing DML

You can import DML to migrate table-controlled or index-controlled to range-partitioned table spaces.

**NOTE**
You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1. In an ISPF editor, type the appropriate DML statements, as follows, and provide the information that is unique to your site:

   For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

   - To migrate existing table-controlled partitioned table spaces to range-partitioned table spaces, type the DML shown in Figure 28.

   **Figure 28**  DML to migrate table-controlled partitioned table spaces to range-partitioned table spaces

   ```sql
   MIGRATE TABLESPACE
   INCLUDE ALL
   SET
   SEGSIZE = 32 --a value > 0 indicates range partitioning
   WHERE
   DBNAME LIKE 'DEMTCP%'
   --specify existing DB name; table spaces in this database use table-controlled partitioning
   ;
   ```

   - To migrate table-controlled table spaces by using a JOIN statement, type the DML shown in Figure 29. The WHERE clause in the DML statement selects the table spaces that contain a table with a value for PARTKEYCOLNUM that is greater than 0. This value indicates that the table space uses table-controlled partitioning.

   **Figure 29**  DML to migrate table-controlled partitioned table spaces to range-partitioned table spaces by using a JOIN statement (part 1 of 2)

   ```sql
   MIGRATE TABLESPACE
   JOIN SYSIBM.SYSTABLES TBLS
   INCLUDE ALL
   SET
   SEGSIZE = 32 --a value > 0 indicates range partitioning
   WHERE
   ;
   ```
To migrate index-controlled partitioned table spaces to range-partitioned table spaces and to convert the index-controlled partitioned table spaces to table-controlled table spaces, type the DML shown in Figure 30.

**Figure 30 DML to migrate index-controlled partitioned table spaces to range-partitioned table spaces**

```sql
MIGRATE TABLESPACE
  INCLUDE ALL
  SET
    SEGSIZE = 32  --a value > 0 indicates range partitioning
  WHERE
    DBNAME LIKE 'DEMICPA%'  --specify existing DB name
;  
MIGRATE TABLES
  SET
    TCPART = 'Y'  --converts index-controlled partitioning to table-controlled partitioning
  WHERE
    DBNAME LIKE 'DEMICPA%'  --specify existing DB name
;  
```

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Migrating a single-table table space to a partition-by-growth table space by importing DML

You can import DML to migrate table-controlled or index-controlled partitioned table spaces or other single-table table spaces to partition-by-growth table spaces. CM/PILOT automatically deletes all but the first partition of a partitioned table space and any dependent indexes.

NOTE
You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1 In an ISPF editor, type the appropriate DML statement (Figure 31), and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 31  DML to migrate a table space to a partition-by-growth table spaces

MIGRATE TABLESPACE INCLUDE ALL
SET MAXPARTITIONS = 10
WHERE
  DBNAME = 'MG025'
  AND NTABLES = 1
;

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Migrating tables from an explicit database to an implicit database by importing DML

You can import DML to migrate tables from an explicit database to an implicit database.

**NOTE**
You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1 In an ISPF editor, type the appropriate DML statement, as follows, and provide the information that is unique to your site:

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

- To migrate tables from an explicit database to an implicit database, type the DML shown in Figure 32.

**Figure 32  DML to migrate tables from an explicit database to an implicit database**

```
MIGRATE TABLES
INCLUDE ALL
SET
   DBNAME = ''
WHERE
   DBNAME LIKE 'MG025%%'
;
```

- To migrate tables from an explicit database that includes a table space with more than one table to an implicit database, type the DML shown in Figure 33.

**Figure 33  DML to migrate tables from an explicit database to an implicit database using a JOIN**

```
MIGRATE TABLES
INCLUDE ALL
JOIN SYSIBM.SYSTABLESPACE TS
SET
   DBNAME = ''
WHERE
   TS.DBNAME LIKE 'MG%%'
   AND TS.DBNAME = SYSIBM.SYSTABLES.DBNAME
   AND TS.NAME = SYSIBM.SYSTABLES.TSNAME
   AND TS.NTABLES > 1
;
```
Migrating tables that contain inline LOB columns by importing DML

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>

Migrating tables that contain inline LOB columns by importing DML

You can import DML to migrate an inline LOB column from a table that is in a partition-by-growth or range-partitioned table space.

NOTE

You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1 Create a DML statement in an ISPF editor. Provide the information that is required for your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

TIP

You can copy the DML statements to perform this task from member ACMDMLM2 in the HLQ.DBCNTL data set.

The following figures show sample DML statements:

- Figure 34 illustrates how to migrate all tables that have a CLOB column. The DML also sets the length of the CLOB column to 100 usable bytes, which indicates that the column is an inline CLOB column.
Migrating tables that contain inline LOB columns by importing DML

Chapter 1  Migrating database environments  139

Figure 34  DML to migrate tables with CLOB columns

<table>
<thead>
<tr>
<th>MIGRATE COLUMNS</th>
<th>-- migrates all tables with LOB columns</th>
<th>-- that match the results of the WHERE clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTH = 104</td>
<td>-- sets length for an inline LOB to 4 plus</td>
<td>-- the inline length (in bytes)</td>
</tr>
<tr>
<td>WHERE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLTYPE IN ('CLOB') AND TBCREATOR = ‘COAIN1’ ;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 35  DML to migrate all dependent objects of an inline CLOB column

<table>
<thead>
<tr>
<th>MIGRATE COLUMNS</th>
<th>-- includes the database, table space, table, and all dependent objects for the table</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCLUDE PARENTS</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td></td>
</tr>
<tr>
<td>LENGTH = 104</td>
<td>-- sets length for an inline LOB to 4 plus</td>
</tr>
<tr>
<td>WHERE</td>
<td></td>
</tr>
<tr>
<td>LENGTH &gt; 4 AND COLTYPE IN ('CLOB') AND TBCREATOR = ‘C2BIN1’ ;</td>
<td></td>
</tr>
</tbody>
</table>

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Migrating stored procedures by importing DML

You can import DML to migrate native SQL stored procedures, external stored procedures, and external SQL stored procedures.

**NOTE**

You can also perform this task by using the CM/PILOT DML_MIGRATE script. For more information, see “Migrating data structures by using DML” on page 123.

1 In an ISPF editor, type the DML statements to match those shown in Figure 36, and provide the information that is unique to your site.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**TIP**

You can copy the DML statements to perform this task from member ACMDMLPI in the HLQ.DBCNTL data set.

---

**Figure 36** DML to migrate stored procedures

```
-- DML TO MIGRATE SPECIFIC STORED PROCEDURES
--
MIGRATE ROUTINES
   SET SCHEMA = 'MC017'
WHERE (NAME = 'SPSQN001' OR
   NAME = 'SPSQN002')
   AND SCHEMA = 'IDML017';
--
MIGRATE ROUTINES
   SET SCHEMA = 'MC017'
WHERE (NAME = 'SPOP0106' OR
   NAME = 'SPSQNG1')
   AND SCHEMA = 'IDML0106';
--
MIGRATE ROUTINES
   SET SCHEMA = 'MC017'
WHERE (NAME = 'SPOP0035' OR
   NAME = 'SPOP0101')
   AND SCHEMA = 'IDML1ALL';
--
-- DML TO MIGRATE ALL EXTERNAL SQL STORED PROCEDURES
--
MIGRATE ROUTINES
WHERE LANGUAGE = 'SQL' AND
   ORIGIN = 'E';
```
2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>

Migrating data structures by using a migrate-type work ID

The CM/PILOT WKID_MIGRATE script contains the steps to create an Analysis worklist to migrate data structures from a local subsystem. This process uses a migrate-type work ID and predefined changes to data structures.

In this task, you will create and execute a task ID.

Before you begin

1. Using CHANGE MANAGER, create a migrate-type work ID and define the data structures to migrate.

   **TIP**
   
   To create a migrate-type work ID, see page 114.

2. To define the data structures to migrate, you can use either the Specification component of CHANGE MANAGER or a CHANGE MANAGER outbound migrate profile with scope rules.

   **TIP**
   
   To create an outbound migrate profile, see page 146.
To create a task ID

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3 Type the name of a new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

4 Select Migrate data structures using an existing WORKID and press Enter.

   The Create TASKID panel is displayed.

5 Specify the information for a task ID.

   A Type the name of an existing migrate-type WORKID or type the name of a work ID name template.

   B (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C (optional) Specify a Comment to describe the task ID.

   D Press END.

   The task ID is created and the TASKID Action Menu is displayed.

To execute the task ID

1 Select Execute a TASKID and press Enter.

   The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

   A Type the data set name for the CM/PILOT Worklist.

   B Type the data set name for the CM/PILOT JCL.

   C Type the data set name for Diagnostics or type SYSOUT.
3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 *(optional)* Override the options that are set in CHANGE MANAGER.

A Type S to select Override CHANGE MANAGER options and then press Enter.

The Override CHANGE MANAGER Options panel is displayed.

B Select Override CHANGE MANAGER Options.

C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

For more information about overriding the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select the steps in the script that you want to modify, and press Enter.

**NOTE**

You can use a CHANGE MANAGER outbound migrate profile to define change rules or specify multiple locations for Analysis worklists that are created using this script. To specify an outbound migrate profile, change the Analysis step. You can specify the profile in an application or on the TASKID Migrate Profiles panel that is displayed.

C To display subsequent Step Settings panels, press Enter.

After the Step Settings panels are displayed, the CHANGE MANAGER Datasets panel is displayed.
Creating a stop list

D Specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press END until the TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

7 To migrate your data structures, execute the Analysis worklist.

---

NOTE
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.

TIP
To execute a worklist, see page 158.

---

Creating a stop list

This task describes how to create a list of tables that should not be analyzed (stop list). An example of a table that should not be analyzed might be those that are defined with the DATA CAPTURE CHANGES attribute. When you analyze a work ID, Analysis determines whether the tables to be modified or migrated in the work ID are listed in a work ID for the work ID creator specified in the STOPLIST ALUIN keyword or STOPLIST installation option. If the tables are included in the stop list for the creator work ID, and the NOSTOPLIST ALUIN keyword is not specified, Analysis will not produce a worklist. If the NOSTOPLIST keyword is specified, Analysis produces the worklist.
1 Create a migrate-type work ID.

**TIP**
To create a migrate-type work ID, see page 114.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.

The Object Specification panel is displayed.

3 Specify the name (or use wildcard characters) for tables that you want to include in the stop list. Press **Enter**.

The Mixed List panel is displayed.

4 Type **M** in the **Act** column to migrate the table.

**NOTE**
You must explicitly specify the tables to be included in the list. That is, you cannot type **M** next to a database or table space.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.

An asterisk (*) and a change-level indicator are displayed adjacent to the data structures that you selected for migration.

5 Press **END**.

The Object Specification panel is displayed.

6 Press **END**.

The WORKID Action Menu is displayed.

**NOTE**
You can also perform this task by using the CM/PILOT DML_MIGRATE script to select the tables based on the value of an attribute. For more information, see “Migrating data structures by using DML” on page 123.
Creating an outbound migrate profile

An outbound migrate profile is used to perform a change migration. An outbound migrate profile can contain scope rules, locations, and change rules.

1. On the CHANGE MANAGER Main Menu, select Migrate Profiles, and press Enter.

   The Migrate Profile Action Menu is displayed.

2. Type the name of the Migrate Profile and select Create a new Migrate Profile. Then, press Enter.

   The Create Migrate Profile panel is displayed.

3. Specify the information for the migrate profile. The profile must contain specifications for scope rules, change rules, or both to be usable.

   A. Select Outbound for the Type.

   B. (optional) To use the scope rules that are defined in a baseline profile, type the name of the Baseline Profile Ref.

   

   **NOTE**

   If you specify scope rules in a migrate profile, you do not need to specify a baseline profile reference.

   

   C. (optional) To specify scope rules for the migrate profile, select Scope Rules.

   

   **NOTE**

   If you use the scope rules from a baseline profile, you do not need to specify scope rules for the migrate profile.

   

   **TIP**

   To create scope rules, see page 148.
D (optional) To specify the locations for the migrate profile, select Outbound Locations.

**TIP**
To specify locations, see page 149.

E (optional) To specify change rules for the migrate profile, select Change Rules.

**TIP**
To create change rules, see page 151.

F (optional) Specify a Comment to describe the migrate profile.

G Press Enter.

4 Press END.

The Migrate Profile Action Menu is displayed.

5 Press END.

The CHANGE MANAGER Main Menu is displayed.

6 Select WORKID and press Enter.

The WORKID Action Menu is displayed.

**Where to go from here**

After you create an outbound migrate profile, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 153</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing a migrate-type worklist” on page 158</td>
</tr>
</tbody>
</table>
Creating scope rules for an outbound migrate profile

You can create scope rules for an existing outbound migrate profile. The scope rules define the set of data structures to include in a migration. If you migrate objects by using scope rules of an outbound migrate profile, the migration requests that are stored in a migrate-type work ID are ignored.

---

**TIP**

To create an outbound migrate profile, see page 146.

---

1. On the CHANGE MANAGER Main Menu, select Migrate Profiles, and press Enter.

   The Migrate Profile Action Menu is displayed.

2. Type the name of the Migrate Profile (or type a wildcard pattern) and select Edit a Migrate Profile. Then, press Enter.

   The Edit Migrate Profile panel is displayed.

3. Type S to select Scope Rules and press Enter.

   The Scope Rules panel is displayed.

4. Type C in the Act column and press Enter.

   The Edit Scope Rule panel is displayed.

5. Specify the information for the scope rule.

   **A** Select the type of data structure that you want to include in the migration.

   **B** Select whether to include or exclude all of the data structures of the type that you selected for Object.

   **C** Type Y to select the dependent object types that you want to migrate.

   **D** Type the relevant first, second, and third parts of the name of the data structure. Wildcard characters are allowed. Press F1 for a list of the object types to which the name applies.

---

**NOTE**

You cannot select the Exclude option in the Edit Scope Rule panel for an auxiliary table space, table, or index.
Specifying locations in an outbound migrate profile

You can specify the location for an existing outbound migrate profile. Each location represents a copy of the subsystem to which you want to migrate data structures. Do not specify locations if you intend to use the profile as CHANGERULESIN2 in a comparison.

**TIP**

To create an outbound migrate profile, see page 146.

1. On the CHANGE MANAGER Main Menu, select **Migrate Profiles**, and press **Enter**.

   The Migrate Profile Action Menu is displayed.

2. Type the name of the **Migrate Profile** (or type a wildcard pattern) and select **Edit a Migrate Profile**. Then, press **Enter**.

   The Edit Migrate Profile panel is displayed.

3. Type **S** to select **Outbound Locations** and press **Enter**.

   The Migrate Profile Locations panel is displayed.

E. Press **Enter**.

6. Press **END**.

   The Scope Rules panel is displayed.

7. Type the letter of an action in the **Act** column to create an additional scope rule, create a scope rule like an existing scope rule, edit or delete an existing scope rule, or insert a scope rule.

8. When you have finished specifying the scope rules, press **END**.

   The Edit Migrate Profile panel is displayed.

9. Press **END**.

   The Migrate Profile Action Menu is displayed. A message on the panel indicates that the migrate profile has been updated.
4 Type E or C in the Act column and press Enter.

The Edit Migrate Profile Location panel is displayed.

5 Specify the information for the location.

A (optional) Type the name of the Group ID four-character identifier. The group ID links locations together.

---

**NOTE**

If you define one or more group IDs for the migrate profile, a single worklist is generated for each group. If you do not define a group ID, one worklist is generated for each location.

---

B Type the Location Name. The name must be unique for the migrate profile.

C Type the value for the SSID. The value for the SSID must match that of the SSID that receives the worklist.

D (optional) Type the name of the Reference Location. The reference location refers to another location that is defined for the migrate profile. When you specify a reference location, the change rules that are defined for that location are used for this location. If you do not specify a reference location, the change rules that are defined explicitly for this location are used.

E Type the name of the default destination data set for a CDL file or a worklist.

F Press END.

The Migrate Profile Locations panel is displayed.

6 Type the letter of an action in the Act column to insert an additional location, create a location like an existing location, or specify the change rules for a location.

---

**TIP**

To specify change rules for a location, see page 151.

---

7 When you have finished specifying the locations, press END.

The Edit Migrate Profile panel is displayed.
Creating change rules for a migrate profile

You can create change rules for an existing outbound or inbound migrate profile. Change rules are used to modify the attributes of data structure definitions.

--- TIP ---
To create an outbound migrate profile, see page 146. To create an inbound migrate profile, see page 361.

--- NOTE ---
If the migrate profile contains locations, the Migrate Profile Locations panel is displayed. Type M in the Act column adjacent to the location for which you want to create change rules.

1 On the CHANGE MANAGER Main Menu, select Migrate Profiles, and press Enter.

The Migrate Profile Action Menu is displayed.

2 Type the name of the Migrate Profile (or type a wildcard pattern) and select Edit a Migrate Profile. Then, press Enter.

The Edit Migrate Profile panel is displayed.

3 Type S to select Change Rules, and press Enter.

The Change Rules panel is displayed.

4 Type C in the Act column and press Enter.

The Edit Change Rules panel is displayed.

5 Specify the information for the change rule.

   A Select Change for the type of rule.
   
   B Select the type of data structure to which you want the change rule to apply.
   
   C Type the name of the attribute of the data structure.

8 Press END.

The Migrate Profile Action Menu is displayed. A message on the panel indicates that the migrate profile has been updated.
Creating change rules for a migrate profile

NOTE
To display a list of the valid attributes, type S to select Display Attribute List. The Edit Rules Attribute Selection panel is displayed. Type the number for the attribute that you want to use and press Enter. Then press END. The Edit Change Rules panel is displayed again, with the name of the Attribute displayed.

D Type the Current Attribute Value of the attribute that you want to change. For more information, see “Specifying values for change rule attributes” on page 49.

E Type the New Attribute Value of the attribute that you want to change. For more information, see “Specifying values for change rule attributes” on page 49.

F (optional) Type the first, second, and third parts of the name of the data structure that you want to change. Wildcard characters are allowed. Press F1 for a list of the object types to which the name applies.

6 Press END.

The Change Rules panel is displayed.

7 From the Change Rules panel, type the letter of an action in the Act column to create an additional change rule, create a change rule like an existing change rule, edit or delete an existing change rule, or insert a change rule. When you have finished specifying the change rules, press END.

The Edit Migrate Profile panel is displayed.

8 Press END.

The Migrate Profile Action Menu is displayed.

Using change rules to set objects to DEFINE NO

You can use a change rule in an existing outbound migrate profile or an inbound migrate profile to set the DEFINE attribute to NO for a table space or index. Specifying the change rule in a migrate profile enables you to reuse the profile again with different work IDs. For an ERP application, you can perform this task to reduce the creation of empty data sets. When you specify the information for the change rule, select Tablespace or Index for Object and then type DEFINE for Attribute. Type the Current Attribute Value and the New Attribute Value of the attribute that you want to change.

TIP
You can also use a CM/PILOT script to set the DEFINE attribute to NO for several table spaces or indexes. For information, see “Changing table spaces and indexes to DEFINE NO by importing DML” on page 132.
Analyzing a migrate-type work ID and generating a worklist

You can analyze your change or migration requests and generate a worklist by using the following methods:

- Analyze a migrate-type work ID in which migrate options and change rules exist in the work ID.
- Analyze a migrate-type work ID in which migrate options exist in the work ID, and the change rules and locations exist in an outbound migrate profile.
- Analyze a migrate-type work ID in which scope rules, change rules, and locations exist in an outbound migrate profile.

The worklist that is generated contains the requests for migration.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of a migrate-type WORKID (or type a wildcard pattern) and select Analyze WORKID and create a worklist. Then, press Enter.

   The Analysis Migrate Worklist Interface panel is displayed.

3. Specify the information to analyze the work ID.

   A. Select the type of analysis that you want to perform.

   B. If you chose a type of analysis in step 3A that uses an outbound migrate profile, type the name of an outbound Migrate Profile or type a wildcard pattern to display a list of migrate profiles and then select the migrate profile to use.

   C. Specify the Run Type.

   D. Press Enter.

   If you specified an outbound migrate profile that uses locations in step 3B, the Migrate Profile Locations panel is displayed. Otherwise, the Analysis JCL Processing Interface panel is displayed.

4. On the Migrate Profile Locations panel, specify the information for the locations.

   A. To include a location, type the name of the worklist data set for the Worklist-DSN.

   B. To exclude a location, type X in the Act column adjacent to the location name.
C Press Enter.

5 On the Analysis JCL Processing Interface panel, specify the data set names to be used by Analysis.

A (batch run type) Type the data set name for the JCL that is generated by this process.

B Type the data set name for the Worklist that is generated by this process.

NOTE
If you specified an outbound migrate profile that uses locations, the Worklist data set name is not displayed.

C Type one of the following options for Diagnostics:

- (foreground run type) To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D (foreground run type) If you specified SYSOUT in step 5C, specify the Sysout Class.

6 Select your options for processing the work ID.

A To override the options for analyzing the work ID, type S to select Override. The Analysis Options panels are displayed. For more information about overriding options for analysis, see “Overriding the default processing options” on page 61.

B To create the input to Analysis, select one of the following options:

- (foreground run type) Type S to select Create analysis input.
- (batch run type) Type S to select Create JCL.

C To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit analysis input.
- (batch run type) Type S to select Edit JCL.

D To create a worklist, select one of the following options:

- (foreground run type) Type S to select Run analysis.
- (batch run type) Type S to select Submit JCL.
Creating a receive-type work ID

When you migrate data only or data structures and data to a different subsystem, you must use a receive-type work ID on the receiving subsystem to execute phase 2 of the worklist. Phase 2 of the worklist creates the data structures and loads any migrated data. You can create a receive-type work ID by using any of the following methods:

- creating the work ID in the Front End before executing the worklist
- building the JCL manually in batch mode by using the NEWWORKID or REPLACEWORKID keyword in the AEXIN input stream

**NOTE**
The receive-type work ID must have the same name as the migrate-type work ID that you used to generate the migrate-type worklist on the sending subsystem.

This task requires an existing migrate-type worklist.
Creating a receive-type work ID

To create a receive-type work ID in the Front End

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of the migrate-type work ID that you used to generate the migrate-type worklist and select Create a new WORKID. Then, press Enter.

   The Create WORKID panel is displayed.

3 Specify the information for the work ID.

   A Select Receive for the Type.

   B (optional) Specify a Comment to describe the work ID.

   C Press Enter.

   The WORKID Action Menu is displayed.

To create a receive-type work ID in batch

You can create a receive-type work ID in batch by editing the JCL outside of ALTER or CHANGE MANAGER.

1 In ISPF, edit the Execution JCL.

2 Insert the NEWWORKID a.b (a.b is the new work ID that you want to create) keyword into the AEXIN data input stream, as shown in Figure 37. The work ID must not currently exist.

   Figure 37 Use of the NEWWORKID keyword

   ```
   //AEXIN DD *
   ACM
   SSID DEBA NEWWORKID RDMAF.REC
   ```

   NOTE

   In the CM/PILOT component of CHANGE MANAGER, you can use a template to specify the NEWWORKID keyword. The #### characters are resolved by an ascending sequence of numbers and the @@@@@@@@ characters are resolved by using the current date. For more information, see “Creating a task ID” on page 628.
To re-create an existing receive-type work ID in batch

You can receive a migrate-type worklist from a remote location to re-create the existing receive-type work ID by editing the JCL outside of ALTER or CHANGE MANAGER.

1. In ISPF, edit the Execution JCL.

2. Insert the REPLACEWORKID a.b (where a.b is the name of the existing work ID that you want to replace) keyword into the AEXIN data input stream, as shown in Figure 38.

Figure 38 Use of the REPLACEWORKID keyword

```
//AEXIN DD *
ACM
SSID DEBA REPLACEWORKID RDAMAF.REC
```

--- NOTE ---
If the specified work ID does not exist, Execution creates it.

When the job is run, the receive-type work ID is replaced.

--- NOTE ---
In the CM/PILOT component of CHANGE MANAGER, you can use a template to specify the REPLACEWORKID keyword. The #### characters are resolved by an ascending sequence of numbers and the @@@@@@@@ characters are resolved by using the current date. For more information, see “Creating a task ID” on page 628.

REPLACEWORKID is not allowed if the existing work ID status indicates that execution was started but not successfully completed.

Where to go from here

After you create the receive-type work ID, select Execute WORKID worklist created by Analysis to execute the worklist.

--- TIP ---
To execute a worklist, see page 158.
Executing a migrate-type worklist

You can execute a migrate-type or a receive-type work ID to execute the worklist that is generated by Analysis. The steps that you perform to migrate data structures, data, or both vary, depending on whether you are migrating within the same subsystem or to a different subsystem. This task describes the basic steps that you perform to execute a work ID and worklist.

TIP
To migrate only data, only data structures, or data structures and data, see “Migrating only data structures” on page 160 or “Migrating data only or data structures and data” on page 161.

Before you begin

Before you can execute a worklist, you must analyze the work ID.

TIP
To analyze a work ID, see page 153.

To execute a migrate-type worklist

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter. The WORKID Action Menu is displayed.

2. Type the name of the WORKID (or type a wildcard pattern) and select Execute WORKID worklist created by Analysis. Then, press Enter. The Execution JCL Build Interface panel is displayed.

3. Specify the information to build and run the JCL.

   A. Select Build Initial JCL to build the JCL for the first time.

   B. Specify a Run Type.

   C. Press Enter.

   The Execution Pre- and Post-Processing Interface panel is displayed.
4 Specify the options for a multi-step job.

--- TIP ---
To create a multi-step job, see page 349.

5 Press Enter.

The Execution JCL Processing Interface panel is displayed.

6 Specify the data set names to be used by Execution.

A Type the data set name for the Execution JCL that is generated by this process.

B Type the data set name for the Worklist that is used by this process.

C Type one of the following options for Diagnostics:

- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D (batch run type) Type the name of the data set to contain the Batch JCL Job.

The batch job is used to generate the Execution JCL.

7 Select your options for processing the worklist that was generated by Analysis.

A To override the options for executing the worklist, type S to select Override Defaults.

The Execution Options panels are displayed. For more information about overriding options for execution, see “Specifying options for Execution” on page 82.

B To edit the worklist that was generated by Analysis, type S to select Edit Worklist.

C To create the input to Execution, select one of the following options:

- (foreground run type) Type S to select Build Execution JCL.
- (batch run type) Type S to select Build Batch JCL.

D To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Execution JCL.
- (batch run type) Type S to select Edit Batch JCL.
Executing a migrate-type worklist

**E** Select one of the following options:

- *(foreground run type)* To execute the worklist, type S to select **Submit Execution JCL**.

- *(batch run type)* To submit the batch JCL, type S to select **Submit Batch JCL**.

**F** Press **Enter**.

**8** *(batch run type)* To execute the worklist, type S to select **Edit New Execution JCL**. From the ISPF editor, submit the JCL.

## Migrating only data structures

To migrate only your data structures within the same subsystem or to a different subsystem, perform the following steps. When you migrate only data structures, the migrate-type worklist contains only the commands for phase 2.

### To migrate only data structures within the same subsystem

Execute the migrate-type work ID and worklist.

### To migrate only data structures to a different subsystem

1. On the receiving subsystem, create a receive-type work ID that has the same name as the migrate-type work ID that you analyzed.

   **TIP**
   
   To create a receive-type work ID, see page 155.

2. Execute the receive-type work ID and worklist.

   **NOTE**
   
   If the sending and receiving subsystems reside on different LPARs, and these LPARs do not share DASD, use FTP to send the data set for the worklist to the LPAR on the receiving subsystem.
Migrating data only or data structures and data

To migrate data only or structures and data, perform the following steps.

To migrate within the same subsystem

1. Execute the migrate-type work ID and worklist.

   **NOTE**
   When you migrate data, the migrate-type worklist contains two phases. By default, Analysis inserts a -STOP command in the worklist to separate the two phases. When you execute the migrate-type work ID, only phase 1 of the worklist is executed. To execute phase 2 of the worklist, you must build restart JCL.

2. Build restart JCL and execute the second phase of the worklist.

   **TIP**
   To execute the second phase of the worklist, see page 164.

To migrate to a different subsystem

The method that you use to migrate to a different subsystem varies, depending on whether you require data set sizing:

- If the values that you specify for the primary and secondary space will satisfy the space requirements for the amount of data that you are migrating, see “If no data set sizing is required” on page 162.

- Otherwise, see “If data set sizing is required” on page 162.

**WARNING**
If you migrate data structures, data, or both to another SSID (subsystem) that is on the same OS/390 system, and you do not change the VCAT name that the STOGROUP is using, you must change either the database name or the table space name.
Executing a migrate-type worklist

If no data set sizing is required

1 On the sending subsystem, execute the migrate-type work ID.

NOTE
On the Execution Override Options panel, select the Generate JCL for the send phase only option.

2 On the receiving subsystem, create a receive-type work ID.

TIP
To create a receive-type work ID, see page 155.

3 Execute the receive-type work ID and worklist.

If data set sizing is required

1 On the sending subsystem, execute the migrate-type work ID.

NOTE
If the Dataset Sizing option is set to no sizing, you must override the default processing options. On the JCL Static Data Set Options panel, specify a data set sizing option. For more information about sizing options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

The PDS member will contain JCL for both phases of the worklist. You can use the JCL to determine the correct sizes for the data sets that you use when executing phase 2 of the worklist. You could name the PDS member as Phase2, for example.
2 On the receiving subsystem, create a receive-type work ID.

--- TIP ---
To create a receive-type work ID, see page 155.

3 Use the execution JCL that you created on the sending subsystem to build restart JCL.

--- NOTE ---
The name that you specify for the PDS member for the old execution JCL is the name that you used for the execution JCL on the sending subsystem. In the example in step 1, you named the PDS member Phase2. You must choose a different name for the PDS member for the new execution JCL. For example, you could name the PDS member for the new execution JCL Phase2$.

--- TIP ---
To execute the second phase of the worklist, see page 164.

4 Edit the new execution JCL as follows:

A In the EXEC statement, change the value in the PARM parameter to the name of the product's installation options module on the receiving subsystem.

B In the STEPLIB DD statement, change the names of the data sets to data sets that are valid on the receiving subsystem.

   For example, change the names of the DSNEXIT or DSNLOAD data sets.

C In the AEXIN input stream, modify the keywords as follows:

   1. Remove the RESTART keyword.

   2. Change the value for the SSID keyword to the name of the receiving subsystem.

   3. (optional) Change the value for the DASDDOPT keyword to the name of the installation options module for DASD MANAGER PLUS on the receiving subsystem.

   4. (optional) Change the value for the CATDOPT keyword to the name of the installation options module for CATALOG MANAGER on the receiving subsystem.
Restarting a worklist

To resume execution after a failure, you can build restart JCL from your existing JCL. The product scans the previous JCL and worklist and modifies the file dispositions. Execution continues from the worklist statement that failed or from the -STOP command. The product executes the worklist from the last valid sync table entry for the work ID. This method preserves any data set name changes that were made to the previous execution override options or to the previous Execution JCL. For more information about restart processing, see “Restart methods” on page 106.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of the WORKID (or type a wildcard pattern) and select Execute WORKID worklist created by Analysis. Then, press Enter.

   The Execution JCL Build Interface panel is displayed.

3 Specify the information to build and run the JCL.

   A Select Build Restart JCL from previous execution JCL.

   B Select a Run Type.

   C Press Enter.

   The Execution JCL Processing Interface panel is displayed.

4 Specify the names of the data sets to be used by Execution.

   A Type the name of the data set for the New Execution JCL that is generated by this process.

   B Type the name of the data set for the Worklist that is used by this process.

   C Type one of the following options for Diagnostics:

       - To write the diagnostics to a sequential file, type the name of the data set.
       - To write the diagnostics to a print data set, type SYSOUT.

   NOTE

   When you submit the new execution JCL, you will be submitting the JCL for phase 2 of the worklist.
D  (*batch run type*) Type the name of the data set to contain the **Batch JCL Job**.

The batch job is used to generate the Execution JCL.

E  Type the name of the data set to contain the **Old Execution JCL**.

---

**NOTE**

The name of the data set for the **Old Execution JCL** cannot be the same as the name of the data set for the **New Execution JCL**.

---

5  Select your options for restarting the worklist.

A  To edit the worklist that was generated by Analysis, type S to select **Edit Worklist**.

B  To create the input to Execution, select one of the following options:

- (*foreground run type*) Type S to select **Build New Execution JCL**.
- (*batch run type*) Type S to select **Build Batch JCL**.

C  To review or modify the JCL, select one of the following options:

- (*foreground run type*) Type S to select **Edit New Execution JCL**.
- (*batch run type*) Type S to select **Edit Batch JCL**.

D  To review or modify the old Execution JCL, type S to select **Edit Old Execution JCL**.

E  Select one of the following options:

- (*foreground run type*) To execute the worklist, type S to select **Submit New Execution JCL**.

- (*batch run type*) To submit the batch JCL, type S to select **Submit Batch JCL**.

F  Press **Enter**.

6  (*batch run type*) To execute the worklist, type S to select **Edit New Execution JCL**.

From the ISPF editor, submit the JCL.
Starting a worklist over

To completely rerun a worklist from the beginning of the worklist, you can build startover JCL from the existing JCL. The product scans the previous JCL and modifies the file dispositions. This method preserves any data set name changes that were made to the previous execution override options or to the previous Execution JCL. For more information about restart processing, see “Restart methods” on page 106.

NOTE
When you build the startover JCL from the previous Execution JCL, all auto-restart capability is lost for the work ID.

1 On the CHANGE MANAGER Main Menu, select WORKID and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID (or type a wildcard pattern) and select Execute WORKID worklist created by Analysis. Then, press Enter.

The Execution JCL Build Interface panel is displayed.

3 Specify the information to build and run the JCL.

A Select Build Startover JCL from previous execution JCL.

B Select a Run Type.

C Press Enter.

The Execution Pre- and Post-Processing Interface panel is displayed.

4 Specify the names of the data sets to be used by Execution.

A Type the name of the data set for the New Execution JCL that is generated by this process.

B Type the name of the data set for the Worklist that is used by this process.

C Type one of the following options for Diagnostics:

■ To write the diagnostics to a sequential file, type the name of the data set.
■ To write the diagnostics to a print data set, type SYSOUT.

D (batch run type) Type the name of the data set to contain the Batch JCL Job. The batch job is used to generate the Execution JCL.
Type the name of the data set to contain the **Old Execution JCL**.

**NOTE**

The name of the data set for the **Old Execution JCL** cannot be the same as the name of the data set for the **New Execution JCL**.

5 Select your options for restarting the worklist.

A To edit the worklist that was generated by Analysis, type **S** to select **Edit Worklist**.

B To create the input to Execution, select one of the following options:

- **(foreground run type)** Type **S** to select **Build New Execution JCL**.
- **(batch run type)** Type **S** to select **Build Batch JCL**.

C To review or modify the JCL, select one of the following options:

- **(foreground run type)** Type **S** to select **Edit New Execution JCL**.
- **(batch run type)** Type **S** to select **Edit Batch JCL**.

D To review or modify the old Execution JCL, type **S** to select **Edit Old Execution JCL**.

E Select one of the following options:

- **(foreground run type)** To execute the worklist, type **S** to select **Submit New Execution JCL**.

- **(batch run type)** To submit the batch JCL, type **S** to select **Submit Batch JCL**.

F Press **Enter**.

6 **(batch run type)** To execute the worklist, type **S** to select **Edit New Execution JCL**.

From the ISPF editor, submit the JCL.
Where to go from here

Now that you have migrated your data structures, you can accomplish the various goals that are listed in the following table.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take a snapshot of a database environment</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>Compare database environments</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Create or alter objects in a database environment</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>Maintain database environments</td>
<td>Chapter 5</td>
</tr>
</tbody>
</table>
Taking a snapshot of a database environment

This chapter presents the following topics:

Before you begin ........................................ 170
Overview ........................................... 170
Using baselines ........................................ 173
  Establishing baselines ............................. 173
  Determining what type of baseline to establish 174
Creating baseline profiles ............................. 175
Creating structure-only baselines ..................... 180
  Using a DB2 catalog to create a catalog baseline 180
  Using a DDL file to create a DDL baseline .... 181
  Using a migrate-type worklist to create a DDL baseline 182
Creating full-recovery baselines ...................... 183
Generating a baseline ............................... 186
  Using the baseline table information .......... 187
  Baseline diagnostic output ......................... 190
Generating a baseline report ........................ 190
  Baseline report diagnostic output ............... 192
Managing baselines .................................. 192
  Protecting baselines from deletion ............. 192
  Deleting baselines ................................ 193
Improving the performance of a baseline .......... 197
Accomplishing your goals ............................ 198
  Creating a catalog baseline profile .......... 198
  Creating scope rules for a catalog baseline profile 199
  Creating a structure-only catalog baseline 201
  Creating a full-recovery baseline ............. 203
  Using a script to create a full-recovery baseline 207
Browsing a baseline ................................ 211
Generating a baseline report ....................... 212
Deleting baselines automatically ................... 213
Before you begin

Before you take an initial snapshot of your database environment, you need to determine which copy of the application database to use and at what point in the development cycle to take the snapshot. If you need to create your database environment, see Chapter 1, “Migrating database environments.”

Overview

A baseline is a snapshot of an application’s data structures at a given point in time. The process of establishing a baseline involves capturing and storing sets of data structure definitions in tables in the Baseline component of the CHANGE MANAGER product. The structure definitions that are stored in the baseline tables are captured from the DB2 catalog, or from a DDL file or migrate-type worklist (as shown in Figure 39).

Figure 39 Establishing a baseline
You can use baselines to accomplish the following goals:

- **Migrate changes**
  
  You can compare the baselines that you take for each version of an application to obtain a file that contains Change Definition Language (CDL). You can then apply the CDL file to instances of the old version of the application to update the data structures.

- **Recover structures and data**
  
  Before you install a new version of an application, take a baseline that includes the data structures and the data of the current version. If you install the new version of the application and have problems, you can recover to the previous version by using the baseline as the recovery point.

- **Control and record changes**
  
  A baseline preserves a version of an application’s data structures for change control. By comparing baselines that represent the same data structures at different points in time, you can see how those structures have changed over time. For example, you can create a baseline when you first install an application. At a later time, you can perform a comparison between the baseline and the DB2 catalog. You can examine the resulting CDL to see what modifications have been made to the application’s data structures since the initial version.

For more information about accomplishing these goals, see Chapter 5, “Maintaining database environments.”

---

**NOTE**

For more information, view the Quick Course Using Baselines. You must have a BMC Support ID to view the Quick Course.

---

Figure 40 illustrates the workflow for the baseline process.

This chapter describes the baseline process, and the tasks that you can perform to create and manage your baselines.
Figure 40  Workflow for the baseline process

Start

Create a baseline profile

Do you want to create a full-recovery baseline for fallback?

yes → Go to alter process

no →

Create a baseline

Do you want a report of the baseline?

yes → Generate baseline report

no →

Do you want to perform a comparison?

yes → Go to comparison process

no →

Do you want to alter the data structures?

yes → Go to alter process

no → Stop
Using baselines

One of the central issues in working with a database management system is recovery. Although DB2 provides a number of facilities for data recovery, it has none for structure recovery. If you make a change to a table, then discover that the change was in error, you must manually apply the changes to restore the structure to its previous state. CHANGE MANAGER, however, provides the tools for both structure-only and structure-with-data recovery.

At the core of the CHANGE MANAGER recovery strategy is the baseline. A baseline is a stored set of data structure definitions, captured from the DB2 catalog at a specific point in time. If the baseline contains only data structure definitions, it is called a structure-only baseline. If the baseline contains data from those structures as well, it is called a full-recovery baseline. The task of creating a baseline is also called establishing a baseline.

**WARNING**

You cannot recover data structures or data without having first established a baseline on those structures.

To have an effective recovery strategy, you must determine when to establish baselines (recovery points) and the type of baseline to establish.

Establishing baselines

Your recovery strategy will vary according to the type of application that is involved and its state of development. In general, three strategies for establishing baselines exist: application milestone, periodic, and exception.

**Establishing application milestones**

Application milestone baselines are established to capture and preserve an application before it is moved to another subsystem or environment. Before you migrate an application from a development environment to a test environment, for example, you can create a baseline of the application. After you create the baseline, you can use it to synchronize versions of the application or use it as a point of recovery.
Determining what type of baseline to establish

**Establishing periodic baselines**

Periodic baselines are established at regular intervals. In the case of the application development system in which structures change daily, you might establish a baseline daily. In this case, you can recover to any of the previously-established baselines if you determine that a particular change is not suitable.

In other situations, such as a query-only database that is updated monthly, a regular monthly baseline might be appropriate. If your system is equipped with a job-scheduling processor, you can define a regularly scheduled job that automatically establishes periodic baselines.

**Establishing exception baselines**

Exception baselines are established to protect the application from the effects of a significant change to its data structures. A significant change is one that carries with it the potential for data loss, such as dropping columns or reducing a column length. You can generate a multi-step job that establishes a baseline before and after an Execution run for an alter-type or migrate-type worklist.

**Determining what type of baseline to establish**

In addition to determining when to establish recovery points for your application, you need to determine whether the recovery points are for data structures only or whether they will also include data. During application development, losing data might be of little importance and a structure-only baseline can be sufficient. If you make changes to a production system, however, a full-recovery baseline gives you the ability to restore data as well.

For example, you can create a baseline that captures the structure definitions that are used by a specific version of an application. During testing, you might make modifications (such as repartitioning an index or using different storage groups) to those structure definitions for performance tuning. If the tuning attempts are unsuccessful, you can recover to the baseline and restore the previous index partitions and storage groups without having to track the changes and undo them manually. If only the structure has been restored, any data added between the creation of the baseline and its recovery is retained.

**Using structure-only baselines**

In a recovery process, you compare the current DB2 catalog to a baseline that contains the data structures that you want to restore. Compare generates CDL for making the changes that are necessary to restore the structures. You import the CDL to an alter-type Work ID and submit the Work ID to Analysis, which generates a worklist. Executing the worklist restores the data structures.
Using full-recovery baselines

When you recover to a full-recovery baseline, the process is much the same as recovering to a structure-only baseline. Using a full-recovery baseline, however, also requires you to decide whether you want to recover with current data, in which case the process is identical to recovering to a structure-only baseline, or with the data that is saved in the baseline.

For example, suppose you establish a full-recovery baseline on April 1 and then make changes to the production data structures. If you uncover a problem with the change that day, you might want to restore to the data in the baseline. If you do not discover the problem until April 15, you might prefer to recover using the current data to avoid losing two weeks of data.

Creating baseline profiles

To create a baseline from a DB2 catalog, DDL file, or a migrate-type worklist, you must first create a catalog or DDL baseline profile. A baseline profile contains information about how the baselines are named and established, and in some instances, when baselines are deleted.

**TIP**

To create a baseline profile, see page 198.

A baseline profile contains the following elements:

**Type**

The baseline profile type specifies the type of baseline (catalog or DDL) that is created with this profile. A catalog baseline uses the data structures from the DB2 catalog. A DDL baseline uses the data structures from a DDL file or a migrate-type worklist.
Baseline Template

The baseline profile can contain a template that is used to generate the names of the baselines that are created with that profile. The template can automatically name, date, and number a baseline. The template name must begin with an alphabetic character, cannot exceed 18 characters, and must contain either @@@@@@@@, @@@@@@@@@, or ####:

- @@@@@@@@ are replaced by the current date in the form YYMMDD.
- @@@@@@@@@ are replaced by the current date in the form YYYYMMDD.
- #### are replaced by an ascending sequence of numbers.

When the Baseline component is invoked to create a baseline, a BLNAME keyword and value can be specified in the ALUIN parameters to explicitly name the new baseline. If you do not specify BLNAME and do not define a template for the baseline profile, an error is returned. If you do not specify BLNAME and you have defined a template for the baseline profile, a name is generated. The owner portion of the baseline name is generated by using the baseline profile owner. You can use baseline templates as input for baseline reports or the comparison process.

The use of name templates have the following restrictions and recommendations:

- Use only one group of the same template characters in a baseline. For example, do not create a template like DDL####TEST####.

- You can specify a template with both a date and a sequence number. For example, you can specify a template with DDL$@@@@@@@@$#### that can resolve to a baseline such as DDL$20000207$0002.

**NOTE**

BMC recommends that the date sequence be defined before the number sequence.

- Baseline templates should have naming conventions which create unique baseline names. If they do not, baselines can share names with baselines that belong to other profiles.

For example, if baseline profile MJF.PROF1 sets the template to PAY@@@@@@@@#### and baseline profile MJF.PROF2 sets the template to PAY#######@@@@@@@@, the resulting baselines would have names that could be interspersed on a baseline list. This situation can be confusing and could result in the wrong baseline being selected if you choose to use the same templates as input.
to the comparison process or to create a baseline report. You can avoid this problem by specifying unique prefixes for each template, such as PAY1@@@@@@@@#### and PAY2####@@@@@@@@, or by using different baseline profile owner names.

**Migrate Profile Reference**

A baseline profile can reference the scope rules that are specified in an outbound migrate profile instead of using the scope rules that are specified in the baseline profile. If you do not specify a migrate profile reference, however, you must specify the scope rules in a catalog baseline profile.

---

**NOTE**

Scope rules are used by catalog baselines only. You cannot specify scope rules for DDL baselines. The DDL file or migrate-type worklist is the scope of a DDL baseline.

---

**Profile Created**

This property of the profile specifies the date and time that the profile was created, and the authorization ID of the user who created it.

**Profile Last Edited**

This property of the profile specifies the date and time that the profile was last edited, and the authorization ID of the user who edited it.

**Last Baseline Built**

This property of the profile specifies the date and time that the last baseline was built using the profile, and the authorization ID of the user who built it.

**Deleteage**

Deleteage identifies the minimum number of days to retain a baseline. This option specifies to automatically delete baselines that are built with this profile that are older than the DELETEAGE value. To retain all of the completed baselines, specify zero for this option and the Retainmax option. For more information, see “Managing baselines” on page 192.
Creating baseline profiles

Retainmax

Retainmax identifies the maximum number of complete baselines to retain. This option specifies to retain a number of the most current baselines and delete all others that are created with this profile. To retain all of the completed baselines, specify zero for this option and the Deletetage option. For more information, see “Managing baselines” on page 192.

CHANGE MANAGER checks the values for the Deleteage and Retainmax options when a new baseline is established with the baseline profile. At that time, all complete and any incomplete baselines are deleted that are outside the range specified for either of these options. Protected baselines are not deleted.

Information about database recovery is stored in the baseline tables of CHANGE MANAGER. If your installation regularly establishes baselines, these tables will grow over time. To control the size of these tables, see “Managing baselines” on page 192. For more information about the baseline tables in CHANGE MANAGER, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

Comment

The baseline profile contains a comment area. The comment is displayed or printed whenever the profile is used and can contain descriptive text about the profile. This comment is not the same as a baseline comment that applies to a specific baseline.

Scope Rules

In a baseline profile, scope rules specify the DB2 objects to include in a baseline. You must identify the scope rules for a catalog baseline profile before the profile can be used to create a baseline (see Figure 41). The scope rules must be defined in the baseline profiles explicitly, or they can be identified in an outbound migrate profile that is referenced from the baseline profile. Scope rules are not applicable for DDL baselines.
CHANGE MANAGER warns you if the scope rules of a baseline profile or the scope rules of an outbound migrate profile that is referenced by a baseline profile might have changed since the baseline was built. If you use the Front End component to modify the scope rules of a profile and existing baselines have been built using that profile, a message is displayed. You can choose either to update the scope rules or exit the profile without updating it.

**WARNING**
If two baselines are built with different scope rules, they might contain the definitions of additional or fewer objects. A comparison of these baselines could build DROP statements (or CREATE statements) for the objects that were not in the scope for both baselines.

**NOTE**
You cannot define scope rules in a baseline profile for stored procedures.

**TIP**
To create scope rules, see page 199.
Creating structure-only baselines

A structure-only baseline is a baseline that captures only data structure definitions. You can use the following items to create a structure-only baseline:

- DB2 catalog
- DDL file
- migrate-type worklist

You specify a baseline profile each time that you create a baseline. The baseline profile determines how the baseline is named and how many baselines are maintained. When you create a baseline, you have the option to delete other baselines that are created with the same profile.

**WARNING**

Some types of modifications can cause loss of data if you use a structure-only baseline for recovery. For example, if a table column is dropped after a structure-only baseline is created, recovering to that structure-only baseline does not restore data for the dropped column. The only method that you can use to restore data is to create a full-recovery baseline (with data) instead of the structure-only baseline. Other types of modifications, like creating new indexes or storage groups, do not cause loss of data when a structure-only baseline is used for recovery.

**NOTE**

Authorizations are not restored when you use a structure-only baseline in the recovery process. To view the authorizations that are included in the baseline, generate a baseline report. You cannot import the baseline report as a DDL file to restore the authorizations; you must re-create the authorizations manually.

Using a DB2 catalog to create a catalog baseline

When you use a DB2 catalog to create a baseline, the scope rules from the baseline profile select the DB2 objects to include in the baseline. You can specify the scope rules explicitly in the baseline profile or reference the scope rules in an outbound migrate profile.

**TIP**

To create a catalog baseline, see page 201.
Using a DDL file to create a DDL baseline

The Baseline component can create a baseline on a set of SQL DDL statements. The DDL file should meet the following criteria:

- DDL files can be either a sequential, 80-column data set or a member of a partitioned data set (PDS).
- The DDL statements must appear in columns 1-72. Columns 73-80 are ignored.
- The DDL commands in columns 1-72 are free format.
- If multiple DDL commands appear in the file, they can be separated by semicolons.

**NOTE**
The DDL statements can be separated by a character other than the semicolon if the --#SET TERMINATOR value control statement is inserted into the DDL file. The --#SET TERMINATOR statement is the convention that is used by the IBM DSNTEP2 sample dynamic SQL program and SPUFI.

The Baseline component uses the pound sign (#) as the value. The value can be any character except a blank, comma (,), double quotation mark ("), single quotation mark (‘), left parenthesis ([), right parenthesis (]), or an underscore (_).

- Comments in a DDL file are ignored by Baseline. Comments begin with two dashes (--) and continue until the end of the line. No other comment characters are recognized.

**NOTE**
The --#SET TERMINATOR statement is not treated as a comment.

The baseline profile that is used for a DDL baseline cannot contain scope rules. In other words, you cannot use a baseline profile that is used to create a catalog baseline to create a DDL baseline. The reverse is also true. You cannot use a baseline profile that is used to create a DDL baseline to create a catalog baseline. The scope of the DDL baseline is all of the objects in the DDL file or the migrate-type worklist.

**NOTE**
You must specify nonprintable and nonviewable characters, such as null and control characters, in limit keys, view text, trigger text, or check constraint text in an external hexadecimal format. For more information, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

A sample DDL file is shown in Figure 42.
Using a migrate-type worklist to create a DDL baseline

The Baseline component can also create a DDL baseline on a migrate-type worklist that is generated by CHANGE MANAGER or another BMC product. When you create a DDL baseline on a migrate-type worklist, the Baseline component reads the list of worklist commands in Table 14.

Table 14  Worklist commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-TIME</td>
<td>specifies the date and time that the worklist was created</td>
</tr>
<tr>
<td>-SSID</td>
<td>specifies the subsystem ID for which the worklist was created</td>
</tr>
<tr>
<td>-AUTH</td>
<td>sets the owner and creator of a DB2 object by using the Execution authorization switching</td>
</tr>
<tr>
<td>-SETS</td>
<td>sets the owner and creator of a DB2 object by using the SET SQLID command</td>
</tr>
<tr>
<td>-GLID</td>
<td>sets a global authorization ID that is used for the owner and creator of DB2 objects, unless it is overridden by -AUTH or -SETS commands</td>
</tr>
<tr>
<td>-SQL</td>
<td>specifies an SQL statement that is used to create, alter, or drop DB2 objects</td>
</tr>
</tbody>
</table>
When you create a DDL baseline, the order of statements in the file or worklist is significant. Baseline reads the file from the top to the bottom and it ignores the worklist sequence number. If Baseline finds an ALTER statement without a preceding CREATE for the same object, it issues an error.

If multiple statements exist for the same object, Baseline applies the statements in the order that it finds them. For example, a CREATE statement that is followed by several associated ALTER statements results in the ALTER statements being applied to the object definition in the order that they are specified in the file. The object definition that is stored in the baseline contains all of the changes that are specified by the associated ALTER statements.

Creating full-recovery baselines

A full-recovery baseline is a baseline that includes data. You use a full-recovery baseline to completely restore data and data structures. To create the baseline, you use an alter-type work ID with a DB2 catalog.

The BLRECOVERPOINT keyword in the ALUIN input stream requests the full-recovery baseline. The keyword tells Analysis to generate worklist commands for creating the baseline, unloading data for the baseline structures (that is, all tables in the baseline profile), and recording the names of the unload data sets.

You must provide the names of the baseline profile and the baseline to build a full-recovery baseline in the worklist. In the ALUIN input stream, specify the name of a catalog baseline profile with the BLPROFILE keyword. You might optionally include the BLNAME keyword, which specifies a baseline name to the Baseline component. If
you do not specify either the baseline name or BLNAME, the Baseline component uses the baseline name template and owner that are provided in the specified baseline profile. If the baseline profile does not contain a name template, however, the name of the baseline or the BLNAME keyword is required.

Figure 43 shows the ALUIN input stream for Analysis to request a full-recovery baseline.

Figure 43  ALUIN input stream for a baseline

<table>
<thead>
<tr>
<th>BLRECOVERPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLNAME RDACRJ.ALT01</td>
</tr>
<tr>
<td>BLPROFILE RDACRJ.BLP1</td>
</tr>
</tbody>
</table>

The Analysis component performs the following functions for a full-recovery baseline:

- generates a -BASE (baseline) command near the beginning of the worklist

  The -BASE command calls the Baseline component to create the baseline.

- generates a -BMCD or -UNLI (unload) command for every DB2 table that is included in the baseline profile

  The -BMCD or -UNLI command unloads the table data to a data set. Analysis records the unload and load options that are specified and stores this information with the baseline. (During the full-recovery process, Analysis uses the load information that is stored with the baseline to ensure that the correct load option has been specified.)

  The SYSRnnnn and the BLRPnnnn data sets are used as unload data sets, and the BLRPnnnn data sets are also used for recovery. You must retain the unload data sets for as long as a recovery point might be needed. However, you will need to retain the BLRPnnnn data sets for a longer time, so ensure that you name them accordingly.

  If you are unloading columns that contain XML or LOB data, the unload utility unloads the data to a file reference output data set that contains the data. The -BMCD command specifies an OUTPUT descriptor for the file reference data set; the -UNLI command specifies a TEMPLATE descriptor.

- generates and executes an -UNRC (unload record) command for every DB2 object whose data was unloaded

  When it is executed, the -UNRC command records the name of the data set that contains the unloaded object in the baseline, so that data can be restored when the baseline is used for recovery.
If the unload contains XML or LOB columns, the command specifies the FILEREF parameter for data sets that contain XML or LOB data.

If statistics indicate that a table is empty and if the NOUNLOADEMPTY keyword is specified in the ALUIN input stream, Analysis generates an -UNRC command to record that the table is included in the baseline but that it should not be unloaded because it is empty. In this case, the -UNRC command specifies the EMPTY parameter. Analysis also generates an -ISMT (is table empty) command for each table that it determines to be empty. If the data set for the DB2 object does not exist, the -UNRC command specifies the DEFINENO parameter.

The worklist and the baseline report provide informational messages that are related to the use of the NOUNLOADEMPTY parameter:

— The worklist that creates a full-recovery baseline provides messages about empty tables that are not unloaded.

— The baseline report for a full-recovery baseline provides messages about data sets that do not exist because the NOUNLOADEMPTY parameter was specified.

— The worklist that reloads current data from a full-recovery baseline provides messages about tables that were empty when the baseline was taken.

generates and executes a -BASF (mark baseline recoverable) command after all unloads are completed on the full-recovery baselines

The -BASF command marks the full-recovery baseline as recoverable.

Table 15 lists the unload and load options that you can specify when you request a full-recovery baseline, and the load options that are required for performing a full recovery.

<table>
<thead>
<tr>
<th>Unload</th>
<th>Load</th>
<th>Required for full recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC UNLOAD PLUS</td>
<td>BMC LOADPLUS</td>
<td>BMC LOADPLUS(^a)</td>
</tr>
<tr>
<td>BMC UNLOAD PLUS(^b)</td>
<td>IBM LOAD(^b)</td>
<td>IBM LOAD(^c)</td>
</tr>
<tr>
<td>IBM UNLOAD</td>
<td>IBM LOAD</td>
<td>IBM LOAD(^c)</td>
</tr>
</tbody>
</table>

\(^a\) If BMC LOADPLUS is not specified, Analysis issues an error message in the diagnostic output. The Front End also attempts to warn you of this situation. You must correct the problem before proceeding.

\(^b\) If you are unloading and loading tables with columns that contain XML or LOB data, you cannot use BMC UNLOAD PLUS to unload the data and IBM LOAD to load the data.

\(^c\) If IBM LOAD is not specified, Analysis changes the worklist to specify IBM LOAD and issues a warning message in the diagnostic output. The Front End also attempts to warn you of this situation. Processing continues.
Generating a baseline

The baseline is established when the Execution component processes the worklist.

CHANGE MANAGER maintains the current authorizations to DB2 objects when the product performs a full recovery. CHANGE MANAGER creates the authorizations that were recorded in the baseline for tables, table columns, views, and view columns that exist in the previously created baseline but which have been dropped from the catalog. However, if a data structure has been dropped from the catalog and is being recovered by a full-recovery baseline, Analysis generates the appropriate GRANT statements in the worklist using the authorizations that are recorded in the baseline for that object.

To restore the data that is stored in a full-recovery baseline, you must indicate to Analysis with the BLRECOVER keyword that you are recovering a baseline with data. For information about restoring a previous version of your data structures and data (or falling back to a full-recovery baseline), see Chapter 5, “Maintaining database environments.”

Tip
To create a full-recovery baseline, see “Creating a full-recovery baseline” on page 203 or “Using a script to create a full-recovery baseline” on page 207.

Generating a baseline

When you generate a structure-only DB2 catalog or DDL baseline, the Baseline component creates the following items:

- structure definitions and authorizations in the Baseline tables
- a diagnostic report

When you generate a full-recovery DB2 catalog baseline, the Analysis component creates the preceding items and unloads the data sets.
Using the baseline table information

Baselines contain a set of structure definitions and authorizations to those structure definitions. These structure definitions are similar to a DB2 catalog. The information that is contained in a baseline is stored in Baseline tables in the CHANGE MANAGER database. While the structure definitions cannot be modified or manipulated directly, they can be used for the following purposes:

- to generate CDL in the comparison process
- to create a baseline report
- to recover data and data structures

Browsing objects and authorizations in a baseline

The Browse Baseline panel (see Figure 44) shows the number of each type of object and authorization in a complete baseline, along with a total count of the objects. The Browse Baseline panel also contains other information pertaining to a baseline, such as the name of the profile that was used to create it, type, status, whether it is protected from deletion, and a baseline comment. The panel also shows the version of the product that was used to update the object and the version of DB2 on which the product is running.

Additionally, the Baseline Selection List and the Baseline Deletion List panels show the total counts for objects and authorizations in baselines.

**NOTE**

Baselines that are created in CHANGE MANAGER contain information about the statistics for each object and the recoverability of the baseline. In addition, the baselines might contain comments.
Using the baseline table information

Figure 44  Browse Baseline panel

Baselines that are created with a previous version of CHANGE MANAGER are updated during the product installation and migration so that their objects and authorizations can be displayed with the new version of the product.

Incomplete baselines display zeros in the count fields. Incomplete baselines are those that were not completed due to system errors.

**TIP**

To browse a baseline, see page 213.

### Storing work IDs

A work ID name is stored in the baseline tables, along with other full-recovery baseline information. The stored work ID name enables the Execution and Compare components to look up a baseline name that is not known when the job starts.

When Execution establishes a full-recovery baseline, it uses the -BASE command. If the baseline name is generated using the name template in the baseline profile, Execution does not know the exact name of the baseline that was created. However, it passes the work ID that established the baseline to the Baseline component using the WORKID keyword. The work ID is stored as part of the baseline information.
When unloading data for the baseline, Execution follows every -BMCD or -UNLI (unload) command with a -UNRC (unload record) command. The -UNRC command records the names of the data sets used for the baseline. When Execution performs an -UNRC command, it uses the work ID to look up the baseline name in which to store the data set names.

The baseline work ID name also enables Compare to look up a baseline name when it is not known. This function is used when Baseline and Compare are steps of a multi-step job stream. The work ID is passed to Compare with the BLWORKID keyword.

For example, a standard alter job might include the following three job parts:

- Baseline—to capture the old structure definitions
- Execution—to implement the changes
- Compare—to generate a record of the changes

When the Baseline step is executed, CHANGE MANAGER passes the name of the work ID that is being used. The actual baseline name is generated using the name template in the baseline profile and is not known at the start of the job. When Compare is run, CHANGE MANAGER passes the work ID using the BLWORKID keyword so that Compare can look up the name of the baseline to use in the comparison.

### Using the baseline data sets

If a full-recovery baseline is requested during Analysis, the worklist that creates the baseline generates one or more data sets to contain the unloaded data. The -UNRC worklist command stores the data set names in the baseline tables, and the -JCLP worklist command recovers these names. When XML or LOB data is included in a full-recovery baseline, the CM_UNLOADDSN product table includes a SOURCETYPE column to indicate whether the unload utility generated a file reference data set.

The Execution diagnostic output file identifies the data sets that are specific to the baseline and those that are permanent work data sets. The data sets that are needed for full-recovery baselines have the ddname BLRPnnnn. If the full-recovery baseline also includes XML or LOB data, file reference data sets are needed. The baseline data sets that Execution generates should be retained as long as needed to allow for baseline recovery. In contrast, you can delete the permanent work data sets after Execution completes successfully. You can generate a JCL job step that automatically deletes the permanent work data sets.
Baseline diagnostic output

The Baseline component, like the other CHANGE MANAGER components, produces diagnostic output in the ALUPRINT file. The diagnostic output might be assigned to TERM (in foreground), assigned to SYSOUT, or sent to a sequential MVS file. If a sequential data set does not exist, one is created with the correct data set information. The output cannot be assigned to a member of a partitioned data set.

Generating a baseline report

You can generate a baseline report to track what has been captured for a specific baseline. The report includes general information about the baseline, DDL and commented-out GRANT statements for each object, and the unload data set names that are associated with a recoverable baseline.

Figure 45 shows an example of the ALUIN keywords that are used to create a baseline report.

Figure 45  ALUIN input stream for a baseline report

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTTYPE</td>
<td>BASELINE</td>
</tr>
<tr>
<td>OBJECTS</td>
<td>(RDACRJ.BL02)</td>
</tr>
</tbody>
</table>

The REPORTTYPE keyword specifies that you want to create a baseline report. The OBJECTS keyword specifies to use the baseline named RDAMAF.IBMSAMPDB.

You can manually insert the new TERMINATOR keyword into the ALUIN input stream to change the character that separates SQL DDL statements. The value for the TERMINATOR keyword can be any character except a blank, comma (,), double quotation mark ("), single quotation mark (‘), left parenthesis [(], right parenthesis [)], or an underscore (_).

If the TERMINATOR keyword is not specified, and if the DDL contains a CREATE TRIGGER or ALTER TRIGGER SQL statement and the trigger text includes a semicolon, the Baseline Report component inserts the --#SET TERMINATOR control statement into the report and uses the pound sign as the SQL statement terminator or separator. Like the value for the TERMINATOR keyword, the value for the --#SET TERMINATOR statement can be any character except a blank, comma, double quotation mark, single quotation mark, left parenthesis, right parenthesis, or an underscore. If the trigger text does not include a semicolon, the Baseline Report component uses a semicolon as the terminator or separator.
The report contains the following information:

- baseline header information
  - baseline name
  - name of the profile that was used to create the baseline
  - timestamp of when the baseline was created
  - baseline status
  - whether the baseline can be used for a full recovery
  - DDL baseline (the data set name that contained the DDL or worklist)
  - whether the baseline is designated as protected from deletion
  - baseline type
  - counts of the objects and authorizations in the baseline
  - the version of CHANGE MANAGER that was used at the time that you took the baseline and at the time that you generated the baseline report
  - the version of DB2 on which CHANGE MANAGER was running at the time that you generated the baseline report
  - a list of the maintenance that was applied to the version of the product at the time that you generated the baseline report

- executable SQL DDL statements (in SPUFI format) for objects that are stored in the baseline

Objects that are implicitly defined are commented out.

--- WARNING ---

The DDL statements are in order by object type and object name (for example, table, synonym, alias, and view). As a result, they might not be in the correct order to execute without generating an error. That is, if an alias or synonym references a view, the product might generate an error because the view statement occurs after the alias or synonym statement.

- data set information that is associated with a full-recovery catalog baseline (table names, data set names, record identifiers, and source type for an XML or LOB file reference)
Baseline report diagnostic output

- SQL GRANT statements that indicate the authorizations for the objects that are stored in the baseline

**NOTE**
The SQL GRANT statements are in order by object type and object name and are commented out.

**TIP**
To create a baseline report, see page 212.

Baseline report diagnostic output

A baseline report produces diagnostic output in the ALUPRINT file. The diagnostic output might be assigned to TERM (in foreground), assigned to SYSOUT, or sent to a sequential MVS file. It cannot be assigned to a member of a partitioned data set.

Managing baselines

To have an effective recovery strategy, you need to determine when to establish baselines and the type of baseline to establish. In addition, you need to determine the period of time that you want to keep your baselines. If you want to keep several baselines and prevent them from being deleted, you can protect them. However, if you want to control the number of baselines that you have stored, you can use several methods to delete the baselines. Older baselines might be deleted when a new baseline is created. The baseline profile contains parameters to control the age or the number of baselines to be retained.

Protecting baselines from deletion

You can keep baselines from being deleted by designating them as protected. When you build a baseline, you can designate it as protected either in the Baseline JCL Processing Interface panel.
Deleting baselines

When you create a baseline and designate it as protected, the PROTECT keyword is placed in the ALUIN input stream.

You should designate your baselines as protected for all major milestones, such as new version release points, and for all major points needed for fall-back processing. These baselines are critical in later change migrations.

The PROTECT keyword is also used in the Analysis ALUIN input stream when you create a full-recovery baseline and designate it as protected.

To remove the protection designation from existing baselines, use the Baseline Selection List panel.

Deleting baselines

You can delete baselines by

- specifying a DELETEAGE or a RETAINMAX value in a baseline profile and then creating a baseline

- specifying a DELETEAGE or a RETAINMAX value in a baseline profile and NoBaseline in the Baseline JCL Processing Interface panel

- explicitly deleting baselines from a list

- for a batch run type, including the OVERRIDE( BASELINE-DELETE ) keyword in the ALUIN DD in the JCL

  The keyword deletes the baseline specified for a baseline profile.

Deleting baselines is optional. The following information about deleting baselines applies only to baselines that you have not designated as protected. Baselines that are designated as protected are not deleted, regardless of the age of the baseline or the number of baselines that you want to keep.
Deleting baselines using DELETEAGE or RETAINMAX

Specifying a value for DELETEAGE or RETAINMAX in a baseline profile enables you to delete baselines automatically when the next baseline is created from the profile. This feature enables you to retain a limited number of complete baselines that were built from a baseline profile. Complete baselines are those that were successfully run to completion. Incomplete baselines are those that were not built successfully due to a system error. When you delete baselines automatically, you delete all incomplete baselines and any complete baselines that match the value that you specify.

You can delete baselines automatically by specifying a nonzero value for Deleteage or Retainmax in the baseline profile. The Baseline component performs the deletions when the job is run. You can use only one method to delete baselines automatically for a profile. That is, you cannot use a nonzero value for both DELETEAGE and RETAINMAX in the same job.

The DELETEAGE or the RETAINMAX keyword, along with the specified value, is placed in the ALUIN input stream when you create a baseline using the profile. If both the DELETEAGE and the RETAINMAX keywords are omitted from the ALUIN input stream, the values from the baseline profile are used. If only one of the keywords is in the ALUIN input stream, the other is set to the default value of zero, regardless of the value in the profile. If a value of zero is specified for both of the keywords, no complete baselines are deleted.

**WARNING**

If you have not taken any baselines recently with the profile, you can delete all of your baselines that are created with the profile except the one that you just created. For example, if you set the DELETEAGE value to 180 and your most recent baseline is 190 days old, you are left with only the baseline you just created.

If you have taken many baselines in a short time and RETAINMAX has a low value, you can delete all baselines except those taken most recently. For example, if the RETAINMAX value is set to five in the profile and you have run six baselines using the profile today, you retain only the five created today that have the latest timestamp.

To determine which baselines to delete when you use DELETEAGE, specify the number of days before the current date to define the age. All incomplete and any complete baselines that are created by the profile that are equal to or older than the specified age are deleted when a baseline is created using the profile. Table 16 shows an example of the baselines that are deleted if five was specified for DELETEAGE, assuming that the current date is 02/28/2013. The baselines that are deleted are shaded.
Deleting baselines

To determine which baselines to delete when you use RETAINMAX, specify the maximum number of complete baselines to retain at all times. Those with the most current timestamp are retained when a baseline is created using the profile. Table 17 shows an example of the baselines that are deleted if three was specified for the number of baselines to retain, assuming that the current date is 02/28/2013. After the first time you delete baselines using RETAINMAX in this example, only one baseline is deleted every time you create a new baseline. The baselines that are deleted are shaded.

You can override the DELETEAGE value or the RETAINMAX value in the baseline profile by using the following methods:

- specify a different value for the DELETEAGE or RETAINMAX keyword in the ALUIN input stream
- specify a different value for Deleteage or Retainmax on the Baseline JCL Processing Interface panel. This method enables you to override the value in the profile for one execution. The profile values are not permanently changed.
Deleting baselines

Deleting baselines using NoBaseline

To delete baselines automatically without creating a new baseline, select NoBaseline on the Baseline JCL Processing Interface panel. This option adds the NOBASELINE keyword to the ALUIN input stream. The baselines that are created by the profile that are equal to or older than the age specified are deleted when you run the job.

Deleting baselines explicitly

You can explicitly delete unprotected baselines by displaying a list of baselines from which you can choose. You can optionally browse the baselines from this list also.

You can also explicitly delete unprotected baselines by listing the baseline profile dependencies for the desired baseline profile and specifically deleting a listed baseline.

Deleting baselines in batch

You can delete the baselines for a baseline profile in a batch run type by including the new OVERRIDE( BASELINE-DELETE ) keyword, in addition to the existing BLNAME and NOBASELINE keywords, in the ALUIN DD in the JCL.

TIP
To delete baselines using a baseline profile, see page 214.

TIP
To delete baselines automatically, see page 213.

TIP
To delete baselines explicitly, see page 215.

TIP
To delete baselines in batch, see page 216.
Improving the performance of a baseline

If the scope of an application contains about 20,000 or more objects, creating a baseline can consume a considerable amount of CPU time, memory, and disk space. To minimize the amount of resources used for the baseline, perform the following steps:

1. Create or verify indexes on the DB2 catalog tables shown in Table 18.

<table>
<thead>
<tr>
<th>Table</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIBM.SYSCOLAUTH</td>
<td>CREATOR, TNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSFIELDS</td>
<td>TBCREATOR, TBNAME, NAME</td>
</tr>
<tr>
<td>SYSIBM.SYSFOREIGNKEYS</td>
<td>CREATOR, TBNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSRELS</td>
<td>CREATOR, TBNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSSYNONYMS</td>
<td>TBCREATOR, TBNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSTABAUTH</td>
<td>TCREATOR, TTNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSTABLES</td>
<td>DBNAME, TSNAME, TYPE</td>
</tr>
</tbody>
</table>

Creating these indexes enables the Baseline component to use indexes to retrieve specific objects. If the indexes do not exist, the Baseline component might need to perform table space scans, which can consume a great deal of CPU time. For more information about creating indexes, see the Installation System User Guide and the BMC Products and Solutions for DB2 Configuration Guide.

2. Modify the value of the COMMIT keyword in the ALUIN input stream to change the frequency with which the inserts are committed.

To avoid lock contentions, by default CHANGE MANAGER generates an SQL COMMIT statement after every 500 SQL INSERT statements when it creates a baseline. You might improve the performance of your baseline by modifying the frequency with which the inserts are committed.

3. Modify the value of the THRESHOLD keyword in the ALUIN input stream to increase the threshold for the number of objects that are pulled from the DB2 catalog.

The THRESHOLD keyword specifies the integer that limits the number of objects that are pulled from the DB2 catalog using the nonunique catalog indexes. If the number of objects that are requested exceeds the limit, the nonunique catalog indexes are ignored. The relevant table space in the catalog is then scanned for the requested objects. For example, if the THRESHOLD is set to 175, and you request 200 objects, the product fully scans the table space instead of opening 200 DB2 cursors.
In general, the THRESHOLD keyword can be set to a very high value. The value is set to 100,069 by default. If you have an earlier version of CHANGE MANAGER, you might improve your performance significantly by manually inserting THRESHOLD 31000 into your ALUIN input stream.

## Accomplishing your goals

CHANGE MANAGER enables you to take a snapshot of an application’s data structures at a given point in time by establishing baselines. Table 19 lists the tasks that you can perform to create baseline profiles, create baselines, and manage baselines.

### Table 19  Baseline tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating a catalog baseline profile”</td>
<td>page 198</td>
</tr>
<tr>
<td>“Creating scope rules for a catalog baseline profile”</td>
<td>page 199</td>
</tr>
<tr>
<td>“Creating a structure-only catalog baseline”</td>
<td>page 201</td>
</tr>
<tr>
<td>“Creating a full-recovery baseline”</td>
<td>page 203</td>
</tr>
<tr>
<td>“Using a script to create a full-recovery baseline”</td>
<td>page 207</td>
</tr>
<tr>
<td>“Browsing a baseline”</td>
<td>page 211</td>
</tr>
<tr>
<td>“Generating a baseline report”</td>
<td>page 212</td>
</tr>
<tr>
<td>“Deleting baselines automatically”</td>
<td>page 213</td>
</tr>
<tr>
<td>“Deleting baselines using a baseline profile”</td>
<td>page 214</td>
</tr>
<tr>
<td>“Deleting a baseline explicitly”</td>
<td>page 215</td>
</tr>
<tr>
<td>“Deleting baselines in batch”</td>
<td>page 216</td>
</tr>
</tbody>
</table>

### Creating a catalog baseline profile

A catalog baseline profile is an object that contains the information that is necessary for establishing a baseline.

1. On the CHANGE MANAGER Main Menu, select **Baseline Profiles**, and press **Enter**.

   The Baseline Profile Action Menu is displayed.

2. Type the name of the **Baseline Profile** and select **Create a new Baseline Profile**. Then, press **Enter**.

   The Create Baseline Profile panel is displayed.
3 Specify the information for the baseline profile.

A Select Catalog for the Type.

B Specify the name of the Baseline Template that you want to use to generate the names of the baselines that are created with the profile.

C (optional) To use the scope rules that are defined in an outbound migrate profile, specify the name of the outbound migrate profile.

---

**NOTE**

If you do not refer to an outbound migrate profile for the scope rules, you must create the scope rules for the baseline.

---

D (optional) Specify values for Deleteage and Retainmax.

---

**NOTE**

You cannot specify nonzero values for both Deleteage and Retainmax.

---

E (optional) Specify a Comment for the baseline profile.

F If you specified an outbound migrate profile in step 3C, press Enter and go to step 4. Otherwise, to specify scope rules for the baseline, select Scope Rules.

---

**TIP**

To create scope rules, see page 199.

---

G Press Enter.

4 Press END to return to the Baseline Profile Action Menu. The baseline profile is added.

5 Press END to return to the CHANGE MANAGER Main Menu.

### Creating scope rules for a catalog baseline profile

You can create scope rules for an existing catalog baseline profile. The scope rules are used to specify the DB2 objects to include in a baseline.
Creating scope rules for a catalog baseline profile

**TIP**
To create a catalog baseline profile, see page 198.

---

1. On the CHANGE MANAGER Main Menu, select **Baseline Profiles** and press **Enter**.

   The Baseline Profile Action Menu is displayed.

2. Type the name of the **Baseline Profile** (or type a wildcard pattern) and select **Edit a Baseline Profile**.

   The Edit Baseline Profile panel is displayed.

3. Type **S** to select **Scope Rules**, and press **Enter**.

   The Scope Rules panel is displayed.

4. Type **C** in the **Act** column and press **Enter**.

   The Edit Scope Rule panel is displayed.

5. From the Edit Scope Rule panel, specify the information for the scope rule.

   **A** Select the type of data structure that you want to include in the baseline.

   **B** Select whether to include or exclude all of the data structures of the type that you selected for **Object**.

   **C** Type **Y** to select the dependent object types that you want to include in the baseline.

   **D** Type the relevant first, second, and third parts of the name of the data structure. Wildcard characters are allowed. Press **F1** for a list of the object types to which the name applies.

   **E** Press **Enter**.

6. Press **END**.

   The Scope Rules panel is displayed.

7. Type the letter of an action in the **Act** column to create an additional scope rule, create a scope rule like an existing scope rule, edit or delete an existing scope rule, or insert a scope rule. When you have finished specifying the scope rules, press **END**.
The Edit Baseline Profile panel is displayed.

8 Press END.

The Baseline Profile Action Menu is displayed. A message on the panel indicates that the baseline profile has been updated.

Creating a structure-only catalog baseline

You create a structure-only catalog baseline to establish a point-in-time snapshot of an application’s data structure definitions. This type of baseline can be used as a recover point in cases in which structure changes are implemented. Baselines that are taken at two different points in time can be compared to determine what changes have been applied to the data structure definitions for that specific time frame.

This procedure describes how to create a structure-only catalog baseline from the Baseline Action Menu. You can also create this type of baseline from the following panels:

- Baseline Profile Action Menu
- Baseline Profile Selection List
- Execution Pre- and Post-Processing Interface panel

1 On the CHANGE MANAGER Main Menu, select Baselines, and press Enter.

The Baseline Action Menu is displayed.

2 Type the name of the Baseline and select Create a Baseline from a Baseline Profile. Then, press Enter.

The Baseline Interface panel is displayed.

3 Type the name of the Baseline Profile and specify the Run Type. Then, press Enter.

The Baseline JCL Processing Interface panel is displayed. The panel displays the Baseline Profile that is used to create the baseline. If you specified a Baseline Template when you created your baseline profile, the Profile’s Baseline Name Template is displayed for the baseline profile. The baseline name uses a template with sequential numbering, dates, or both.
Creating a structure-only catalog baseline

4 On the Baseline JCL Processing Interface panel, if the Baseline Name is not displayed, type the name of the baseline.

5 (optional) Specify a BL Comment that is specific to this baseline. Baseline comments can be up to 62 characters in length.

6 Specify values for Deleteage and Retainmax. These values will override the values that are specified in the baseline profile.

7 To make the baseline ineligible for deletion, type Y for Protect. This option enables you to capture major points in the development cycle (such as versions and releases) that are exempted from the Deleteage and Retainmax deletion rules.

8 Specify the data set names to be used in the baseline.

A (batch run type) Type the data set name for the JCL that is generated by this process.

B Type one of the following options for Diagnostics:

- (foreground run type) To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

C (foreground run type) If you specified SYSOUT in step 8B, specify the Sysout Class.
9 Select your options for generating the baseline.

A To create the input, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

B To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

C To create the baseline, select one of the following options:

- (foreground run type) Type S to select Run.
- (batch run type) Type S to select Submit JCL.

D Press Enter.

Creating a full-recovery baseline

You create a full-recovery baseline to establish a point-in-time snapshot of an application’s data structure and data. This type of baseline can be used as a recover point in cases in which structure changes are implemented or data is lost. Baselines that are taken at two different points in time can be compared to determine what changes have been applied to the data structure definitions for that specific time frame.

Before you begin

To exclude XML or LOB data from the full-recovery baseline, modify one of the following profiles and create scope rules to exclude the tables that contain the XML or LOB columns:

- baseline profile that you used to build the full-recovery baseline
- outbound migrate profile that the baseline profile references

To create a full-recovery baseline

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.
2 Specify an existing alter-type WORKID and select **Analyze WORKID and create a worklist**. Then, press **Enter**.

The Analysis Alter Worklist Interface panel is displayed.

---

**NOTE**

You can only establish a full-recovery baseline by requesting it through Analysis.

---

3 Select **Generate a worklist which will convert and reload current data and establish a full recovery baseline**. Then, select the **Run Type** and press **Enter**.

The Analysis Baseline Interface panel is displayed.

4 Provide the information that is needed to build the full-recovery baseline.

   A Specify the name of the **Baseline Profile**.

   B Specify the **Baseline Name**, or leave the field blank and use the profile’s baseline name template to name the baseline automatically.

   C To protect the baseline, type **Y** for **Protect**.

   D Press **Enter**.

   The Analysis JCL Processing Interface panel is displayed.

5 Specify the data set names to be used in the baseline.

   A *(batch run type)* Type the data set name for the **JCL** that is generated by this process.

   B Type the data set name for the **Worklist** that is generated by this process.

   C Type one of the following options for **Diagnostics**:

      - *(foreground run type)* To display the diagnostics on the terminal, type **TERM**.
      - To write the diagnostics to a sequential file, type the name of the data set.
      - To write the diagnostics to a print data set, type **SYSOUT**.

   D *(foreground run type)* If you specified **SYSOUT** in step **5C**, specify the **Sysout Class**.

6 Select your processing options.
A To override the options for analyzing the work ID, type S to select Override. The Analysis Options panels are displayed. For more information about overriding options for analysis, see “Overriding the default processing options” on page 61.

B To create the input to Analysis, select one of the following options:

- (foreground run type) Type S to select Create analysis input.
- (batch run type) Type S to select Create JCL.

C To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit analysis input.
- (batch run type) Type S to select Edit JCL.

D To create a worklist, select one of the following options:

- (foreground run type) Type S to select Run analysis.
- (batch run type) Type S to select Submit JCL.

E To modify the worklist, type S to select Edit Worklist.

F Press Enter.

Analysis generates the steps in a worklist for creating a full-recovery baseline.

7 Press END.

The WORKID Action Menu is displayed.

8 Select Execute WORKID worklist created by Analysis.

The Execution JCL Build Interface panel is displayed.

9 Specify the information to build and run the JCL.

A Select Build Initial JCL.

B Specify a Run Type.

C Press Enter.

The Execution Pre- and Post-Processing Interface panel is displayed, as shown in Figure 46. Note that a full-recovery baseline has been specified.
Creating a full-recovery baseline

Figure 46  Execution Pre- and Post-Processing Interface panel

ACMFEXC1  Execution Pre- and Post-Processing Interface
Command ===>

WORKID . . . : RDACRJ.ALT03
Specify additional steps to be included in this job. Then press Enter.
Pre-Execution Compare
  1. Do not do a compare before execution
  2. Compare two previous baselines
  3. Compare a previous baseline to the current catalog
Full Recovery Baseline has been specified.

Post-Execution Compare
  1. Do not do a compare after execution
  2. Compare a previous baseline to the new catalog structures
  3. Compare the baseline built during a previous job step to the new
     catalog structures
  4. Create CDL to fall back to a previous baseline
Post-Execution Baseline
  1. Do not build a baseline after execution
  2. Build a baseline after execution
Commands: HELP PREVIOUS END

10  Press Enter.

The Execution JCL Processing Interface panel is displayed.

11  Specify the data set names to be used by Execution.

  a  Type the data set name for the Execution JCL that is generated by this process.

  b  Type the data set name for the Worklist that is used by this process.

  c  Type one of the following options for Diagnostics:
      ■ To write the diagnostics to a sequential file, type the name of the data set.
      ■ To write the diagnostics to a print data set, type SYSOUT.

  d  (batch run type) Type the name of the data set to contain the Batch JCL Job. The
      batch job is used to generate the Execution JCL.
Select the processing options. When Execution processes the worklist, the full-recovery baseline is established.

A To override the options for executing the worklist, type S to select Override Defaults. The Execution Options panels are displayed. For more information about overriding options for execution, see “Overriding the default processing options” on page 61.

B To edit the worklist that was generated by Analysis, type S to select Edit Worklist.

C To create the input to Execution, select one of the following options:

- (foreground run type) Type S to select Build Execution JCL.
- (batch run type) Type S to select Build Batch JCL.

D To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Execution JCL.
- (batch run type) Type S to select Edit Batch JCL.

E Select one of the following options:

- (foreground run type) To execute the worklist, type S to select Submit Execution JCL.
- (batch run type) To submit the batch JCL, type S to select Submit Batch JCL.

F Press Enter.

Using a script to create a full-recovery baseline

You can use the CREATE_FULL_RCV_BL script in the CM/PILOT component of CHANGE MANAGER to generate an Analysis worklist to create a full-recovery baseline. A full-recovery baseline provides you with a recovery point for old data structures and old data. You can also use it to recover your data structures with current data.

You can also create your own script to create a baseline and then generate a baseline report. For more information about scripts, see Chapter A, “Using scripting tools to automate change management.”

In this task, you will create and execute a task ID.
**Before you begin**

Using CHANGE MANAGER, create a catalog baseline profile.

---

**TIP**

To create a catalog baseline profile, see page 198.

---

**To create a task ID**

1. On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.

   The CM/PILOT Main Menu is displayed.

2. Select **TASKIDs** and press **Enter**.

   The TASKID Action Menu is displayed.

3. Type the name of a new **TASKID** and select **Create a TASKID**. Press **Enter**.

   The Script Selection List is displayed.

4. Select **Create a Full-Recovery Baseline**, and press **Enter**.

   The Create TASKID panel is displayed.

5. Specify the information for a task ID.

   A **CM/PILOT** uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for **WORKID** or type the name of a work ID name template.

   B *(optional)* Type the name of a CM/PILOT **Application**. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C *(optional)* Specify a **Comment** to describe the task ID.

   D Press END.

   The task ID is created and the TASKID Action Menu is displayed.
To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

   A Type the data set name for the CM/PILOT Worklist.

   B Type the data set name for the CM/PILOT JCL.

   C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

   A Type S to select Override CHANGE MANAGER options and then press Enter.

   The Override CHANGE MANAGER Options panel is displayed.

   B Select Override CHANGE MANAGER Options.

   C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

   For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

   WARNING

   BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

   D Press END until the TASKID Interface panel is displayed.
Using a script to create a full-recovery baseline

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select **Create Worklist**. Then, press **Enter**.

The Step Settings panel is displayed.

B To modify the execution step in the script, type S to select **Step 2 Execution step**. Then, press **Enter**.

The TASKID Execution Step Options panel is displayed.

C After you specify your step options, press **Enter** until the TASKID Baseline Profiles panel is displayed.

D On the TASKID Baseline Profiles panel, type the name of an existing **Baseline Profile** to be used in this task ID or type a wildcard pattern and select a profile from the list. Then, press **Enter**.

The TASKID New Baselines panel is displayed.

E On the TASKID New Baselines panel, specify the information for the new baseline.

1. Type the name of the baseline.

2. To protect the baseline from deletion, type Y to select **Protect**.

3. Press **Enter**.

   The CHANGE MANAGER Datasets panel is displayed.

F On the CHANGE MANAGER Datasets panel, specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press END until the TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select **Edit Worklist**.

B To create the JCL to run the worklist, type S to select **Create JCL**.

C To review or modify the JCL, type S to select **Edit JCL**.

D To submit the JCL to run the CM/PILOT worklist, type S to select **Submit JCL**.
**Browsing a baseline**

You can view the information for a baseline. The information includes the type, status, and comments for the selected baseline. In addition, the date and time the baseline was created is provided. The total number of objects and authorizations for each object type is also provided.

1. On the CHANGE MANAGER Main Menu, select **Baselines**, and press **Enter**.

   The Baseline Action Menu is displayed.

2. Type the name of a **Baseline** and select **Browse a Baseline**. Then, press **Enter**.

   The Browse Baseline panel is displayed.

3. Press **END** to return to the Baseline Action Menu.

**NOTE**

To list the existing baselines, specify a wildcard pattern for **Baseline** and press **Enter**. The Baseline Selection List is displayed. Type **B** in the **Act** column adjacent to the baseline that you want to browse, and press **Enter**. The Browse Baseline panel is then displayed.
Generating a baseline report

The baseline report contains information that is specific to a baseline, including the DDL for the objects in the baseline.

1. On the CHANGE MANAGER Main Menu, select Baselines, and press Enter.

   The Baseline Action Menu is displayed.

2. Type the Baseline name and select Generate a report for a Baseline. Then, press Enter.

   The Baseline Report Panel 1 panel is displayed.

   **NOTE**

   If you do not know the name of the baseline, you can type a wildcard pattern for the Baseline name and press Enter. The Baseline Selection List is displayed. Type S in the Act column adjacent to the baseline for which you want to generate a report, and press Enter. The Baseline Report Panel 1 is displayed.

3. On the Baseline Report Panel 1, select the Run Type and press Enter.

   The Baseline Report Panel 2 panel is displayed.

4. Specify the data set names to be used in the report.

   A  *(batch run type)* Type the data set name for the JCL that is generated by this process.

   B  Type the data set name for the Report that is generated.

   C  Type one of the following options for Diagnostics:

      - *(foreground run type)* To display the diagnostics on the terminal, type TERM.
      - To write the diagnostics to a sequential file, type the name of the data set.
      - To write the diagnostics to a print data set, type SYSOUT.

   D  *(foreground run type)* If you specified SYSOUT in step 4C, specify the Sysout Class.

5. Select your options for generating the report.

   A  To create the input, select one of the following options:

      - *(foreground run type)* Type S to select Create Input.
      - *(batch run type)* Type S to select Create JCL.
Deleting baselines automatically

You can delete baselines automatically, without first creating a baseline, by specifying a value for the DELETEAGE or RETAINMAX keywords in the baseline profile. When you specify a value for one of these keywords, you can delete all of the baselines that are outside a specified range of dates or a range of numbers.

1. On the CHANGE MANAGER Main Menu, select Baselines, and press Enter.

   The Baseline Action Menu is displayed.

2. Type the name of a Baseline and select Create a Baseline from a Baseline Profile. Then, press Enter.

   The Baseline Interface panel is displayed.

3. Specify the Baseline Profile name that you want to use and the Run Type. Then, press Enter.

   The Baseline JCL Processing Interface panel is displayed.

4. To delete baselines without first creating a baseline, type Y for NoBaseline.

5. Specify values for Deleteage and Retainmax. These values will override the values that are specified in the baseline profile.

   **NOTE**
   You cannot specify nonzero values for both Deleteage and Retainmax.
6 Specify the data set names to be used.

A (batch run type) Type the data set name for the JCL that is generated by this process.

B Type one of the following options for Diagnostics:

- (foreground run type) To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

C (foreground run type) If you specified SYSOUT in step 6B, specify the Sysout Class.

7 Select your options for deleting the baselines.

A To create the input, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

B To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

C To delete the baselines, select one of the following options:

- (foreground run type) Type S to select Run Report.
- (batch run type) To submit the batch JCL, type S to select Submit JCL.

D Press Enter.

Deleting baselines using a baseline profile

You can delete baselines that are created with baseline profiles.

1 On the CHANGE MANAGER Main Menu, select Baseline Profiles, and press Enter.

The Baseline Profile Action Menu is displayed.

2 Type a wildcard pattern for the Baseline Profile and select Delete BASELINES based on Retainmax or Deleteage. Then, press Enter.

The Baseline Profile Selection List panel is displayed.
Deleting a baseline explicitly

3 Type S in the Act column adjacent to the profile that is used to create the baseline that you want to delete. Then, press Enter.

The Baseline Deletion List panel is displayed, as shown in Figure 47.

NOTE
A baseline must meet the Deleteage or Retainmax criteria for deletion to be included in the Baseline Deletion List.

Figure 47 Baseline Deletion List panel

4 Specify one of the Delete Options and press Enter.

Deleting a baseline explicitly

You can delete a baseline explicitly from a list of baselines.

1 On the CHANGE MANAGER Main Menu, select Baselines, and press Enter.

The Baseline Action Menu is displayed.

2 Type a wildcard pattern for Baseline and select Delete a Baseline. Then, press Enter.

The Baseline Selection List panel is displayed.
Deleting baselines in batch

**NOTE**
The Rec column indicates whether the baseline is recoverable and the Prot column indicates whether the baseline is protected. You should carefully consider whether to delete a full-recovery baseline.

3 Type S in the Act column adjacent to the baseline that you want to delete. Then, press Enter.

The Baseline Delete Confirmation panel is displayed.

4 To delete the baseline, select Delete Baseline, and press Enter.

Deleting baselines in batch

When you create the input for a batch run type, you can specify to delete the baseline specified for a baseline profile.

1 On the CHANGE MANAGER Main Menu, select Baselines, and press Enter.

The Baseline Action Menu is displayed.

2 Type the name of a Baseline and select Create a Baseline from a Baseline Profile. Then, press Enter.

The Baseline Interface panel is displayed.

3 Specify the Baseline Profile name that you want to use and the Run Type. Then, press Enter.

The Baseline JCL Processing Interface panel is displayed.

4 Type Y for NoBaseline.

5 Specify values for Deleteage and Retainmax. These values will override the values that are specified in the baseline profile.

**NOTE**
You cannot specify nonzero values for both Deleteage and Retainmax.
6 Specify the data set names to be used.

A Type the data set name for the JCL that is generated by this process.

B Type one of the following options for Diagnostics:
   - To write the diagnostics to a sequential file, type the name of the data set.
   - To write the diagnostics to a print data set, type `SYSOUT`.

7 Select your options for deleting the baselines.

A To create the input, type `S` to select **Create JCL**.

B To review or modify input to the process, type `S` to select **Edit JCL**.

C Edit the JCL as follows:

1. Ensure that the BLPROFILE and BLNAME keywords do not specify a template.
2. Ensure that the NOBASELINE keyword is included in the ALUIN DD.
3. Insert the OVERRIDE( BASELINE-DELETE ) keyword in the ALUIN DD, as shown in the following example:

   ```
   //ALUIN DD *
   SSID DEGA
   COMMIT 500
   SOURCETYPE CATALOG
   BLPROFILE AGH.A1DB
   BLNAME MYTEST.DEMOAPP
   NOBASELINE
   OVERRIDE( BASELINE-DELETE )
   DELETEAGE 0
   RETAINMAX 0
   /*
   ```

D To delete the baselines and to submit the batch JCL, type `S` to select **Submit JCL**.

E Press Enter.
Now that you have created baselines, you can use them to compare data structures or use them as a recover point when you alter your data structures. For more information, see Chapter 3, “Comparing database environments,” or Chapter 4, “Altering data structures in a database environment.”
Comparing database environments

This chapter presents the following topics:

Before you begin ................................................................. 220
Overview ................................................................. 220
Comparing data structures ................................................. 223
  Synchronizing data structures ........................................ 223
  Versioning data structures ........................................... 229
Deciding which type of comparison to perform ..................... 235
Defining the scope of a comparison .................................... 236
  Using scope rules as the scope ..................................... 238
  Using the objects in the input data structures as the scope 241
Resolving changes to names ............................................. 245
  Using an outbound migrate profile to override the rules 247
Specifying the options for a comparison ................................. 248
  Specifying the types of objects .................................... 248
  Specifying the object attributes .................................... 250
  Specifying an outbound migrate profile ............................ 253
Overriding the default values for a comparison ...................... 254
  Allocating data sets .................................................. 255
  Including statements and dependent objects ........................ 255
  Displaying change rules .............................................. 257
  Keeping the original owner name for an object .................. 257
Generating the JCL or the input for the comparison .................. 258
Generating CDL .............................................................. 258
  Validation of object text dependencies ............................. 258
  Management of renamed objects .................................... 259
Results of a comparison .................................................. 261
  Diagnostic log ......................................................... 262
  CDL file ............................................................... 262
Generating a CDL file from an alter-type work ID ................... 263
Applying the CDL ........................................................... 264
Improving the performance of a comparison ......................... 264
Accomplishing your goals ............................................... 266
  Comparing a DB2 catalog to a DB2 catalog ....................... 266
  Using a profile to scope a catalog to a catalog comparison ..... 272
  Comparing a DB2 catalog to a DDL file ............................ 276
Before you begin

Before you compare data structures, you might need to create a baseline of your environment. For information about creating baselines, see Chapter 2, “Taking a snapshot of a database environment.”

Overview

Updating data structures is an integral part of the change management process. You can use the Compare component of the CHANGE MANAGER product to generate Change Definition Language (CDL) commands in a file that show the differences between two sets of data structures. You can then import the CDL file to a work ID and process the file as a set of change requests for the current subsystem. You can also import the file to a work ID on a different subsystem to update a separate version of the data structures.

The comparison process enables you to perform the following tasks:

- record changes made to a local or remote subsystem
- understand structure changes between two subsystems to help determine changes to be migrated from one subsystem to the other
- move application changes from one subsystem to another subsystem using the same application, without losing local modifications to the data structures
- roll back changes made to a subsystem
The process of identifying structural differences between two copies of the same data structure, and then making the data structures identical, is referred to as *synchronization*. You might want to synchronize your data structures if, for example, you have separate development groups who independently modify several copies of a set of data structures. At various times, you need to synchronize the copies to ensure that all of the developers are using the same structure definitions.

When you perform a comparison so that you can change a version of a data structure, you compare baselines that represent the same set of structures at different points in time. This process is referred to as *versioning*. You might want to version your data structures so that you can properly maintain versions of your applications.

This chapter discusses how you can use CHANGE MANAGER to compare your data structures. Using the Compare component of CHANGE MANAGER to generate CDL enables you to automate the process of updating data structures. The benefit of using CDL rather than DDL is that CDL allows more types of modifications to data structures and can retain local modifications to those structures that would otherwise be lost.

**NOTE**

For more information, view the Quick Course Using Compare. You must have a BMC Support ID to view the Quick Course.

**Figure 48** illustrates the process of comparing your data structures using CHANGE MANAGER.
Figure 48 Workflow for the comparison process

Start

Do you need to create a baseline for the comparison?
  yes → Go to baseline process
  no →

Do you need to create a worklist for the comparison?
  yes → Go to migration process
  no →

Do you want to use or do one of the following?
  yes → Create an outbound migrate profile
  no →

Do you want to use a DB2 catalog, DDL file, or a worklist to perform a comparison?
  yes →
  no →

Do you want to synchronize two sets of data structures?
  yes →
  no →

Use a baseline to identify version differences between data structures at different points in time

Do you want to import the CDL to alter the data structures?
  yes → Go to alter process
  no →

Stop
Comparing data structures

The Compare component requires two inputs: the primary set of input (also called Compare1) and the secondary set of input (also called Compare2). The CDL that Compare generates shows the changes for the objects in the primary input that are required to create data structures like those that are defined in the secondary input. In other words, the primary input is the environment to which the resulting CDL will be applied. The secondary input is the environment that contains the objects that already have the structure definitions that you want.

The input structures for a comparison can be contained in the DB2 catalog, a baseline, a migrate-type worklist, or a DDL file. In addition, Compare can generate CDL from change requests stored in an alter-type work ID.

Synchronizing data structures

When you perform comparisons to synchronize data structures, you compare two separate sets of structures. Each set of structures represents the definition of that instance of the structures for the current point in time.

You can use Compare to keep multiple copies of data structures synchronized. Comparisons to synchronize data structures are often performed at the first stage of the application development cycle. For example, you might have several copies of the same data structures, each of which is modified independently by a separate group of developers. At various times, you need to synchronize each copy to ensure that all developers are using the same structure definitions.

The following methods are recommended for synchronizing your data structures:

- comparing a DB2 catalog to a DB2 catalog
- comparing a DB2 catalog to a DDL file (or a migrate-type worklist)
- comparing a DDL file (or a migrate-type worklist) to a DDL file (or a migrate-type worklist)

Comparing a DB2 catalog to a DB2 catalog

A DB2 catalog to a DB2 catalog comparison enables you to compare a specified set of DB2 objects from a DB2 catalog to DB2 objects of the same type from another catalog (see Figure 49). You can also compare objects of the same type from the same catalog.
If you want to use scope rules to specify any collection of unrelated objects and their dependents, you can use a catalog baseline profile or an outbound migrate profile. If you want to use change-type or suppress-type change rules, or generate the CDL files for different locations, you can also use an outbound migrate profile. After you synchronize these data structures, you can use the change migration process to manage the changes among them.

Figure 49  Comparing a DB2 catalog to a DB2 catalog

When you compare two DB2 catalogs, CHANGE MANAGER automatically resolves differences in the object names between the catalogs, if only one object name and its dependent object types were used to define the scope of the comparison. That is, a baseline profile or an outbound migrate profile was not used to specify the scope. The resulting CDL shows the changes for the objects at the primary input location that are required to create data structures like those defined at the secondary input location. The primary input objects retain their names.
You can compare DB2 objects using the following catalog combinations:

- local catalog to local catalog
- local catalog to remote catalog
- remote catalog to local catalog
- remote catalog to remote catalog

_TIP_
To perform a catalog to catalog comparison, see page 266.

**Requirements for the comparison for a remote location**

Before you perform a DB2 catalog to a DB2 catalog comparison for a remote location, ensure that the following requirements are met:

- Remote locations must be defined in the SYSIBM.LOCATIONS table of the local DB2 subsystem, connected using the Distributed Data Facility (DDF) of DB2, and usable with SPUFI or the Query Management Facility (QMF).

To determine whether the location is usable with SPUFI, use the following query in SPUFI on your local subsystem:

```sql
SELECT NAME FROM remoteSubsystem.SYSIBM.SYSTABLES
WHERE CREATOR = 'SYSIBM' AND NAME LIKE 'SYSCH%';
```

An SQLCODE=0 return code indicates that you can perform a comparison for a remote location.

- You must completely install CHANGE MANAGER on the local and remote DB2 subsystems.

- The version, release, and maintenance level of the CHANGE MANAGER installations must be the same.

**Considerations for the comparison**

You should consider the following information when you plan to perform a DB2 catalog to a DB2 catalog comparison:

- If you do not resolve conflicts for the object names and creators in the comparison, automatic name resolution is performed.
You do not retain local modifications to data structures if you directly apply the CDL that results from a DB2 catalog to a DB2 catalog comparison. For example, if you compare a table that has a column added specifically for your location (Compare1) to a similar table from another location (Compare2) that does not contain this column, Compare generates CDL to drop the column from your table (Compare1).

**NOTE**

If you want to retain this type of local modifications to your data structures, you must manually edit the CDL file.

No support is provided for renamed objects. Compare only resolves renamed objects when a baseline is involved in the comparison and the renamed objects are correctly recorded in the CHANGE MANAGER rename table. For more information about managing renamed objects, see “Management of renamed objects” on page 259.

You can specify a remote SSID for a baseline profile that contains scope rules in the SCOPE1 and SCOPE2 ALUIN keywords. The remote SSID is the first part of a three-part name for the profile.

### Comparing a DB2 catalog to a DDL file or a migrate-type worklist

A DB2 catalog to a DDL file or a migrate-type worklist comparison (see Figure 50) is generally used to synchronize data structures in a development environment. DDL that is created with a data modeling tool or a migrate-type worklist that is created by CHANGE MANAGER can be used in the comparison process to apply incremental changes to an application.
With a DB2 catalog to a DDL file or migrate-type worklist comparison, you can

- use DDL created outside of CHANGE MANAGER
- control the set of input data structures to be compared
- control the contents of the CDL file

With a DB2 catalog to a DDL file or migrate-type worklist comparison, you cannot

- determine when objects have been renamed, unless you specify an outbound migrate profile for Compare2 (CHANGERULESIN2)

If you specify an outbound migrate profile, the change rules can resolve the names of objects and creators before the comparison. For example, you might need to apply change rules when the names that are specified by the vendor of the DDL are different from those that are used in the DB2 catalog. As a result, the renamed objects appear as ALTER statements in the CDL file.

If you do not specify an outbound migrate profile, any name changes cause renamed objects to appear as DROP and CREATE statements in the CDL file.
always determine whether objects will be dropped

Whether or not objects are dropped depends on how you specify the set of input data structures to be compared.

**TIP**
To compare a DB2 catalog to a DDL file or worklist, see page 276.

---

**Comparing a DDL file or migrate-type worklist to a DDL file or migrate-type worklist**

Versions of an application’s data structures are often contained in DDL files or migrate-type worklists. When you generate CDL by using a DDL file or a migrate-type worklist, the CDL shows the changes between the two versions of the data structures.

A DDL file (or a migrate-type worklist) to another DDL file (or a migrate-type worklist) comparison (see Figure 51) is generally used to maintain versions of your DDL. This comparison scenario assumes that you are comparing an older DDL file to a newer DDL file.

**NOTE**
The DDL file or migrate-type worklist should define the application’s data structures completely. For example, the file cannot contain an ALTER statement that modifies the primary quantity of a table space without containing a CREATE statement that establishes the table space.
With a DDL file to a DDL file comparison, you can generate CDL for all changes that have occurred to your DDL. Any name changes will be eliminated either before or after the comparison is actually performed (by using an outbound migrate profile) or will cause renamed objects to appear as DROP and CREATE statements in the CDL file.

**TIP**

To compare a DDL file or worklist to a DDL file or worklist, see page 280.

### Versioning data structures

You can perform comparisons to migrate structure changes from one version of an application to another, or to roll back changes to an application’s data structures. When you maintain versions of an application, you often have to migrate changes from development to test, or from test to production. When you roll back data structure changes, you might also have to roll back data to a previous version of the application.
To properly maintain versions of an application, you must take baselines of your application that represent the state of the data structures (and possibly data) at specific points in time. Baselines are usually taken before an application’s data structures are put into production, and then periodically over time. Baselines provide a way to discern how a set of data structures has changed over time. You then use these baselines as input to the comparison process.

You can use the following types of comparisons to migrate or roll back changes to an application’s data structures:

- baseline to baseline (migrating and rolling back changes)
- baseline to catalog (migrating changes)
- catalog to baseline (rolling back changes)

**Comparing a baseline to a baseline**

A baseline captures a set of data structure definitions at a point in time. A baseline to a baseline comparison (see Figure 52) is used to maintain versions of an application. A baseline can be either a catalog baseline or a DDL baseline. A catalog baseline is a baseline that is established on part of the DB2 catalog. A DDL baseline is a baseline that is established on a file that contains DDL. An example of a file that contains DDL is a worklist that is generated in CHANGE MANAGER.

You can use the comparison as one step towards migrating changes from a development environment to a test environment, or from a test environment to a production environment. This comparison scenario assumes that you are comparing an older baseline (Compare1) to a newer baseline (Compare2).
When you use a baseline as a primary input (Compare1) for a comparison, you are using the baseline as the referenced set of objects to be modified. When a baseline is the secondary input (Compare2), it contains the final preferred set of structures.

With a baseline to a baseline comparison, you can

- generate CDL for all changes that have occurred to a set of objects at any point in time for which you have a baseline
- identify name changes that have occurred between two baselines

You can import the CDL file that this comparison generates to another subsystem. For example, after you import and analyze the CDL file, CHANGE MANAGER generates an alter-type worklist. When you execute the worklist on the receiving subsystem, the changes in the worklist update the data structures on the receiving subsystem without losing any local modifications.

_TIP_
To compare a baseline to a baseline, see page 283.
Comparing a catalog baseline to a DB2 catalog

A catalog baseline to a DB2 catalog comparison (see Figure 53) is used to maintain versions of an application’s data structures. It can be used as one step towards migrating changes from a development environment to a test environment, or from a test environment to a production environment.

This type of comparison is similar to a baseline to a baseline comparison, except that the secondary input (Compare2) is the current catalog rather than another baseline. When you specify the DB2 catalog as the secondary input to Compare, you typically generate CDL for one of the following reasons:

- to record changes made to the catalog
- to transmit changes made on the local subsystem to other subsystems
- to update copies of an application’s data structures on the same subsystem

**NOTE**

When you update copies of data structures on the same subsystem, the duplicates will have different names. You must tailor the generated CDL for the duplicate names either by using change rules in an outbound migrate profile or by editing the CDL file itself.
With a catalog baseline to a DB2 catalog comparison, you can

- generate CDL that expresses the exact changes that have been applied to the baseline objects since the baseline was taken
- identify name changes that have occurred to the baseline objects since the baseline was taken
- include statistical information for objects in the CDL file

_TIP_

To compare a baseline to a catalog, see page 287.

**Comparing a DB2 catalog to a catalog baseline**

A DB2 catalog to a catalog baseline comparison (see Figure 54) is used to roll back changes that are made to an application. If you have taken full-recovery baselines, you can roll back data as well as the data structures.

**Figure 54  Comparing a DB2 catalog to a catalog baseline**
Comparing the DB2 catalog to a baseline shows all of the changes that have been made since the baseline was taken. These changes include dependent object changes as well as the high-level changes that were requested in the Work ID. For example, if you change the name of a table, CDL generated from a Work ID shows only that change. If you compare the DB2 catalog to a baseline that was established before the Work ID was executed, the CDL shows changes to dependent indexes, views, synonyms, and aliases as well.

When you specify the DB2 catalog as the primary input (Compare1) to Compare, you generate CDL that restores the catalog to the state that is specified in the secondary input (Compare2). With a DB2 catalog to a catalog baseline comparison, you can generate CDL that expresses the changes necessary to convert the current catalog to the selected catalog baseline definition.

_TIP_

To compare a catalog to a baseline, see page 290.

**Comparing a baseline to a DDL file or migrate-type worklist**

A catalog or DDL baseline to a DDL file or migrate-type worklist comparison (see Figure 55) is used to identify changes that need to be made to DB2 catalog objects that are represented by the baseline. A baseline to migrate-type worklist comparison can also be used to accomplish a baseline to remote DB2 catalog comparison.

_TIP_

To compare a baseline to a DDL file, see page 294.
Deciding which type of comparison to perform

Because the Compare component is so flexible, it can be difficult to understand which type of comparison that you should use. Table 20 summarizes the different types of comparisons that can help you accomplish your goals.

Table 20  Types of comparisons  (part 1 of 2)

<table>
<thead>
<tr>
<th>Goal</th>
<th>Type of comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare two structures quickly</td>
<td>DB2 catalog to DB2 catalog</td>
</tr>
<tr>
<td>Analyze the differences between two sets of data structures</td>
<td>DB2 catalog to a migrate-type worklist</td>
</tr>
<tr>
<td>Compare two objects in the same subsystem’s catalog, or across subsystems where DDF is implemented</td>
<td>DB2 catalog to a DDL file</td>
</tr>
<tr>
<td>Synchronize objects</td>
<td>DB2 catalog to a catalog baseline (baseline established earlier)</td>
</tr>
<tr>
<td>Use DDL from a third-party vendor</td>
<td>DB2 catalog to a DDL file</td>
</tr>
<tr>
<td>Receive DDL from data modeling tools</td>
<td></td>
</tr>
<tr>
<td>Roll back changes</td>
<td></td>
</tr>
</tbody>
</table>
Defining the scope of a comparison

The set of input data structures to be compared is called the scope of the comparison. The scope of a comparison can consist of scope rules (or specifications) that are defined in a baseline profile or an outbound migrate profile, scope rules that are dynamically created from a DDL file or migrate-type worklist, or the objects or a subset of the objects in each input.

Table 21 describes the scope for comparing different types of input structures.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Type of comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply changes to environments to implement a specific version</td>
<td>Baseline to baseline</td>
</tr>
<tr>
<td>Fall back to a previous version of an application’s data structures</td>
<td></td>
</tr>
</tbody>
</table>

Table 20  Types of comparisons (part 2 of 2)

<table>
<thead>
<tr>
<th>Goal</th>
<th>Type of comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply changes to environments to implement a specific version</td>
<td>Baseline to baseline</td>
</tr>
<tr>
<td>Fall back to a previous version of an application’s data structures</td>
<td></td>
</tr>
</tbody>
</table>

Defining the scope of a comparison

The set of input data structures to be compared is called the scope of the comparison. The scope of a comparison can consist of scope rules (or specifications) that are defined in a baseline profile or an outbound migrate profile, scope rules that are dynamically created from a DDL file or migrate-type worklist, or the objects or a subset of the objects in each input.

Table 21 describes the scope for comparing different types of input structures.

Table 21  Scope of a comparison (part 1 of 2)

<table>
<thead>
<tr>
<th>Primary input (Compare1)</th>
<th>Secondary input (Compare2)</th>
<th>DDL baseline</th>
<th>Catalog baseline</th>
<th>DDL or migrate-type worklist</th>
<th>DB2 catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL Baseline</td>
<td>Compare uses all of the objects captured in both baselines. The subsystems can be either local or remote.</td>
<td>Compare uses all of the objects captured in both baselines. The subsystems can be either local or remote.</td>
<td>Compare uses all of the objects in both the baseline and the DDL file or worklist.</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This type of comparison cannot be performed.</td>
<td></td>
</tr>
<tr>
<td>Catalog Baseline</td>
<td>Compare uses all of the objects captured in both baselines. The subsystems can be either local or remote.</td>
<td>Compare uses all of the objects captured in both baselines. The subsystems can be either local or remote.</td>
<td>Compare uses all of the objects in both the baseline and the DDL file or worklist.</td>
<td>Compare uses all of the objects in the baseline and the catalog objects that are defined by the scope rules in the catalog baseline profile that was used to establish the baseline.</td>
<td></td>
</tr>
</tbody>
</table>
Defining the scope of a comparison

### Table 21  Scope of a comparison (part 2 of 2)

<table>
<thead>
<tr>
<th>Primary input (Compare1)</th>
<th>Secondary input (Compare2)</th>
<th>DDL baseline</th>
<th>Catalog baseline</th>
<th>DDL or migrate-type worklist</th>
<th>DB2 catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL or Migrate-Type Worklist</td>
<td></td>
<td>Compare uses all of the objects in both the baseline and the DDL file or worklist.</td>
<td>Compare uses all of the objects in both the baseline and the DDL file or worklist.</td>
<td>Compare uses all of the objects in the DDL files or worklists.</td>
<td>Compare uses the objects in the DDL file or the worklist. The objects that Compare uses from the catalog are determined by the SCOPETYPE and SCOPE$n$ keywords. The catalogs can be either local or remote.</td>
</tr>
<tr>
<td>DB2 Catalog</td>
<td></td>
<td><em>Error</em></td>
<td>Compare uses all of the objects in the baseline and the catalog objects that are defined by the scope rules in the catalog baseline profile that was used to establish the baseline.</td>
<td>Compare uses the objects in the DDL file or the worklist. The objects that Compare uses from the catalog are determined by the SCOPETYPE and SCOPE$n$ keywords. The catalogs can be either local or remote.</td>
<td>Compare uses the catalog objects that are specified with the SCOPERULE$n$ or SCOPE$n$ keywords. The DB2 catalogs can be either local or remote. You can define the scope explicitly by specifying the object type and object name. You can also define the scope by specifying scope rules in a catalog baseline profile or an outbound migrate profile. You can specify a remote SSID for a baseline profile that contains scope rules in the SCOPE1 and SCOPE2 ALUIN keywords. The remote SSID is the first part of a three-part name for the profile.</td>
</tr>
</tbody>
</table>

---

**TIP**

To use a profile to scope a comparison, see page 272.
Using scope rules as the scope

When you perform a comparison that involves the DB2 catalog and a DDL file or migrate-type worklist, you can use one of the following scope types to control the scope of the comparison:

- baseline profile
- outbound migrate profile
- DDL file or migrate-type worklist

The type of scope that you select determines which scope rules are used in the comparison:

- If you select a scope type of baseline profile or outbound migrate profile, the scope rules that are defined for a specific profile are used to select a subset of the DB2 catalog objects. Compare compares the catalog objects that match the scope rules to all of the objects in the DDL file, migrate-type worklist, or DB2 catalog.

  By using a profile, you can ensure that all objects that match the scope rules are retrieved from the catalog. In other words, you can use scope rules to retrieve objects that are not included in the DDL file, migrate-type worklist, or DB2 catalog.

- If you select a scope type of a DDL file or a migrate-type worklist, the objects that are contained in the DDL file or migrate-type worklist are used to build a dynamic set of scope rules. These dynamically created scope rules are used to select objects from the catalog that match the objects in the DDL file or migrate-type worklist.

  When you specify the scope type as DDL or a migrate-type worklist, you do not have to rely on the profile scope rules to select a subset of objects from the catalog. You can use a data modeling tool to generate DDL for changed or new objects and have Compare use only those objects as the implicit scope for the comparison. Using this scope type speeds the comparison process because the entire object set does not have to be generated using the data modeling tool, and unchanged objects are not retrieved from the catalog. However, objects to be created or dropped might not be identified because they are not included in the scope of comparison.

Specifying the scope with a baseline profile

A catalog baseline profile might be used for the scope of a comparison when the DB2 catalog is compared with a DDL file, migrate-type worklist, or DB2 catalog. Compare uses the scope rules that are defined in the catalog baseline profile to select the objects from the DB2 catalog.
The SCOPETYPE and SCOPE\textit{n} keywords are used in the ALUIN input stream to indicate the use of the baseline profile and the name of the profile. The SCOPE1 keyword is used in a DB2 catalog to a DDL file, migrate-type worklist, or DB2 catalog comparison. The SCOPE2 keyword is also used in a DDL file, migrate-type worklist, or DB2 catalog to a DB2 catalog comparison. The example in Figure 56 shows the keywords used for comparison of a DDL file to a DB2 catalog.

**Figure 56  Keywords used in a DDL file to a DB2 catalog comparison**

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>DDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE2</td>
<td>LOCAL</td>
</tr>
<tr>
<td>SCOPETYPE</td>
<td>BASELINE</td>
</tr>
<tr>
<td>SCOPE2</td>
<td>RDACRJ.BLPROF1</td>
</tr>
</tbody>
</table>

The subsystem on which the baseline profile resides can be either local or remote. When you compare a DB2 catalog to a DB2 catalog, you can specify a remote SSID in the first part of a three-part name for the profile in the SCOPE1 or SCOPE2 ALUIN keywords.

---

**NOTE**

When you change the names of objects that are used in baselines, ensure that the scope rules that are used to create the baseline still apply to the new names of the objects. For example, if the name of table DCH.PRODTB is changed to DCH.TESTTB, a scope rule that specifies DCH.*TB for tables is valid. A scope rule specified as DCH.PROD*, however, will not select the table with its new name. To avoid this problem, specify scope rules that select a high-level object, such as database or table space, and include all of the dependent objects.

---

**Specifying the scope with an outbound migrate profile**

An outbound migrate profile, like the baseline profile, might be used for the scope of a comparison when the DB2 catalog is compared with a DDL file, migrate-type worklist, or DB2 catalog. Compare uses the scope rules in an outbound migrate profile to select a subset of objects from the catalog.

---

**NOTE**

For more information, view the Quick Course Creating Migrate Profiles. You must have a BMC Support ID to view the Quick Course.

---

The SCOPETYPE and SCOPE\textit{n} keywords are used in the ALUIN input stream to indicate the use of the outbound migrate profile and the name of the profile. The SCOPE1 keyword is used in a DB2 catalog to a DDL file, migrate-type worklist, or DB2 catalog comparison. The SCOPE2 keyword is used in a DDL file, migrate-type worklist, or DB2 catalog to a DB2 catalog comparison. The example in Figure 57 shows the keywords that are used for a DB2 catalog to a DB2 catalog comparison.
Using scope rules as the scope

Figure 57  Keywords used for a DB2 catalog to a DB2 catalog comparison

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>LOCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE2</td>
<td>LOCAL</td>
</tr>
<tr>
<td>SCOPETYPE</td>
<td>PROFILES:MIGRATE,MIGRATE</td>
</tr>
<tr>
<td>SCOPE1</td>
<td>RDACRJ.OMP1</td>
</tr>
<tr>
<td>SCOPE2</td>
<td>RDACRJ.OMP1</td>
</tr>
</tbody>
</table>

**NOTE**
If a scope rule for a database, table space, or table that includes auxiliary objects is specified in an outbound migrate profile, the Compare component ignores the rule.

Specifying the scope with a DDL file or a migrate-type worklist

When you compare a DDL file or a migrate-type worklist to a DB2 catalog, you can use the objects that are contained in the DDL file or the objects that are defined by the -SQL statements in the migrate-type worklist as the scope of the comparison, instead of using the scope rules in an outbound migrate profile or in a baseline profile. If you use the DDL file or the migrate-type worklist to define the scope of the comparison, Compare extracts only the matching objects from the catalog. Compare uses the CREATE statements in the DDL file or migrate-type worklist to dynamically build scope rules when the comparison is run. These rules are not saved after the comparison is performed.

Only the objects that are named in CREATE or ALTER statements are used as the scope of comparison for both the DDL file and the migrate-type worklist scopes. Any DROP statements in the file are ignored. Compare attempts to select objects from the catalog that match the object names that appear in the CREATE or ALTER statements in the DDL file or migrate-type worklist. In addition, the ALTER statements must reference objects that are named in the CREATE statements.

The SCOPETYPE keyword is used in the ALUIN input stream to indicate the type of scope, as shown in the example shown in Figure 58.

Figure 58  Keywords used for a migrate-type worklist to a DB2 catalog comparison

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>WORKLIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE2</td>
<td>LOCAL</td>
</tr>
<tr>
<td>SCOPETYPE</td>
<td>WORKLIST</td>
</tr>
</tbody>
</table>

Consider the following items when you use the objects contained in a DDL file or migrate-type worklist for the scope of a comparison with the DB2 catalog:

- Changes to subordinate objects that are not in the DDL file or migrate-type worklist will not be in the resulting CDL.
Foreign keys that do not have generated constraint names might not resolve correctly in the CDL if the column list that defines the key has changed.

If the parent table is not included in the scope of the comparison, foreign keys and unique constraints will not be verified.

If you compare a DB2 catalog to a DDL file or a migrate-type worklist, only the ALTER and CREATE statements appear in the resulting CDL; DROP statements do not appear.

If you compare a DDL file or a migrate-type worklist to a DB2 catalog, only the ALTER and DROP statements appear in the resulting CDL; CREATE statements do not appear.

You must specify nonprintable and nonviewable characters, such as null and control characters, in limit keys, view text, trigger text, or check constraint text in an external hexadecimal format.

For more information, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

If you want DROP or CREATE statements to be generated in the CDL, choose a baseline profile or an outbound migrate profile for the scope.

Using the objects in the input data structures as the scope

The objects in a baseline, DDL file, and migrate-type worklist are fixed. When you compare two input data structures in which the set of objects is fixed, the scope is simply the objects in each input. For example, if you compare two DDL files, the scope is simply the objects contained in each file.

Using a baseline in the comparison

When you compare two local baselines, the data structures that are contained in each baseline constitute the scope of the comparison. BMC recommends that you take each baseline using the same baseline profile, even though the different scope rules might be in effect. CHANGE MANAGER warns you when the scope rules of a baseline profile or the scope rules of an outbound migrate profile that are referenced by a baseline profile have been changed since the baseline was created.

When you perform a catalog baseline to a DB2 catalog comparison, you cannot specify the scope. Compare uses the baseline name to look up the baseline profile that was used to create the baseline. It then uses the scope rules that are associated with the profile to retrieve objects from the catalog that match the scope rules.
Using the objects in the input data structures as the scope

**NOTE**
The scope rules that are used to create the original baseline might need to be modified to select the current definition of the objects. This modification might be necessary if you rename the objects in such a way that the original scope rules no longer match the objects' new names.

The CMPTYPE\textsubscript{n} and CMPIN\textsubscript{n} keywords are used in the ALUIN input stream to indicate the use of a baseline in a comparison. The example in Figure 59 shows the use of these keywords for a baseline to baseline comparison.

**NOTE**
The LOCATION\textsubscript{n} keyword will be added if either of the baselines is a remote baseline.

Figure 59  Keywords used for a baseline to a baseline comparison

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>BASELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1</td>
<td>RDACRJ.BLUNLD201102110001</td>
</tr>
<tr>
<td>LOCATION1</td>
<td>DEBA</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>BASELINE</td>
</tr>
<tr>
<td>CMPIN2</td>
<td>RDACRJ.BLUNLD201101250002</td>
</tr>
<tr>
<td>LOCATION2</td>
<td>DEAH</td>
</tr>
</tbody>
</table>

Using a DDL file or a migrate-type worklist in the comparison

When a DDL file or migrate-type worklist is used as part of a comparison, Compare considers all of the DB2 objects contained in the DDL file or migrate-type worklist. When you use a DB2 catalog as one input and a DDL file or migrate-type worklist as the other input, Compare requires a scope to determine the objects to use. For more information, see “Using scope rules as the scope” on page 238.

The DDL file should meet the following criteria:

- DDL files can be either a sequential, 80-column data set or a member of a partitioned data set (PDS).
- The DDL statements must appear in columns 1-72. Columns 73-80 are ignored.
- The DDL commands in columns 1-72 are free format.
- If multiple DDL commands appear in the file, they can be separated by semicolons.
Comments in a DDL file are ignored by Compare. Comments begin with two dashes (--) and continue until the end of the line. No other comment characters are recognized.

**NOTE**
The DDL statements can be separated by a character other than the semicolon if the --#SET TERMINATOR value control statement is inserted into the DDL file. The --#SET TERMINATOR statement is the convention that is used by the IBM DSNTEP2 sample dynamic SQL program and SPUFI.

The Compare component uses the pound sign (#) as the value. The value can be any character except a blank, comma (,), double quotation mark ("), single quotation mark (‘), left parenthesis ([], right parenthesis []), or an underscore (_).

**NOTE**
The --#SET TERMINATOR statement is not treated as a comment.

**Using a DB2 catalog in the comparison**

When one of the comparison inputs is the DB2 catalog, CHANGE MANAGER implicitly or explicitly selects a subset of objects from the catalog. You can define the scope of a DB2 catalog to DB2 catalog comparison explicitly by specifying the object type and object name for the primary input (Compare1) and the secondary input (Compare2) (see Figure 60). The object type that you specify for the primary input must match the object type that you specify for the secondary input. As an alternative, you can use the scope that is defined for an outbound migrate profile or a catalog baseline profile to select a subset of objects from the catalog for comparison. (For more information, see “Using scope rules as the scope” on page 238.)

**NOTE**
You cannot use the scope of a profile when a DB2 catalog and a baseline are used in a comparison.
Using the objects in the input data structures as the scope

Figure 60  Compare Catalog to Catalog panel

Listing the objects

On the Compare Catalog to Catalog panel, you can display a list of objects to choose from, based on object type and name. If you specified to display a list for both the primary input and the secondary input locations, the primary objects are shown first and then the secondary objects are shown. You can use action codes to select objects in the list and to display dependent objects as you can with lists of objects in Specification.

You can use wildcard characters in the first or second part of the object name to find a requested object. For information about using wildcard patterns, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

**WARNING**

For the PIECESIZE attribute of a nonpartitioned index to be compared correctly, you must include the associated table space for the nonpartitioned index in the scope of the comparison.

Specifying a table owner

You can optionally specify a table owner to limit the scope of database and table space comparisons. Then, only tables with the specified owner and their dependent objects are used in the comparison.
The SCOPERULEn keywords are used in the ALUIN input stream to indicate the owner and the name of the object. For example, if you specified to compare only the tables that have specific owners in two databases, Compare would add the following keywords to the ALUIN input stream:

<table>
<thead>
<tr>
<th>SCOPERULE1</th>
<th>MAF8220 DB MAF8D22A</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOPERULE2</td>
<td>DSN8220 DB DSN8D22A</td>
</tr>
</tbody>
</table>

In this example, the location of the primary input (Compare1) corresponds to the SCOPERULE1 keyword and the location of the secondary input (Compare2) corresponds to the SCOPERULE2 keyword. You could also specify all table owners by using an asterisk (*) wildcard character.

**Specifying remote DB2 locations**

You can generate a list of remote DB2 locations by using wildcard characters or you can explicitly specify the name of a DB2 remote location. Remote locations must be accessible using the DDF of DB2. If you display a list, you can specify an action code on the list to select a remote location. The default location is the local catalog.

**Figure 61** shows an example of the ALUIN input stream keywords that Compare uses for a local catalog to remote catalog comparison. CMPTYPE1 specifies that the source location is a local catalog, and CMPTYPE2 specifies that the target location is a remote catalog. LOCATION2 specifies the SSID of the remote DB2 subsystem.

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>LOCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE2</td>
<td>REMOTE</td>
</tr>
<tr>
<td>LOCATION2</td>
<td>DBDC</td>
</tr>
</tbody>
</table>

**Resolving changes to names**

Object names or creator names might not always match within the scope of a comparison. In a DB2 catalog to a DB2 catalog comparison, CHANGE MANAGER resolves the differences in the object owners, schemas, and database names by applying automatic change rules. These change rules make the objects in Compare2 match the objects in Compare1.
CHANGE MANAGER applies automatic change rules only when the following conditions are met:

- You are comparing a DB2 catalog to a DB2 catalog.

- You have explicitly defined the scope by specifying the object type and object name for Compare1 and Compare2 on the Compare Catalog to Catalog panel.

- You have not specified an outbound migrate profile that contains change rules that should be applied before the comparison is performed (CHANGERULESIN2 keyword).

For example, suppose that you want to compare all of the production tables in the PROD database named like POWNER*. (Compare1) to all of the test tables in the TEST database named like TOWNER*. (Compare2). Figure 62 illustrates the input data structures in the comparison.

**Figure 62 Comparing two DB2 catalogs**

Before CHANGE MANAGER builds the automatic change rules, it ensures that none of the tables have the same table name and it ensures that no duplicate objects exist. If a duplicate object exists, the comparison process results in an error because CHANGE MANAGER does not know how to build the automatic change rule. If no duplicate objects exist, CHANGE MANAGER ignores the owner part and matches the object name of the Compare1 object to the object name of the Compare2 object. It then builds the automatic change rules to make the owner of the Compare2 object match the owner of the Compare1 object.

Using the example tables listed in Figure 62, CHANGE MANAGER builds the following automatic change rules:

- change TOWNERA to POWNERX for table TOWNERA.TB1
- change TOWNERB to POWNERY for table TOWNERB.TB2
- change TOWNERC to ####### for table TOWNERC.TB3
Using an outbound migrate profile to override the rules

Specifying an outbound migrate profile for the comparison enables you to accomplish the following tasks:

- override the automatic change rules
- use the scope that is defined for the profile to select a subset of objects from the DB2 catalog
- improve performance

The outbound migrate profile can contain change rules to resolve the names of the objects and creators; it cannot contain locations. When you specify an outbound migrate profile, Compare inserts the CHANGERULESIN2 keyword into the ALUIN input stream.

CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison. However, only the change rules for object owners and names are applied. The change rules for attributes are ignored.

For a DB2 catalog to a DB2 catalog comparison, if the object names and creators are the same in different subsystems in a comparison, you can specify CHANGERULESIN2 NULL on the Compare Catalog to Catalog panel, or in the ALUIN input stream to override the change rules.

**NOTE**

When CHANGE MANAGER builds a CDL CREATE statement and cannot determine the owner of an object or cannot determine the database name for a table space or table, the owner or database name is replaced with a set of pound (#) signs. If you select to retain the original owner, CHANGE MANAGER inserts the DEFAULTOFF keyword into the ALUIN stream and builds the automatic change rule to keep table TOWNERC.TB3.

In addition, if the object is a table space or a table, the database attribute is replaced with the primary input database name. To specify the correct value, you can edit the CDL file.

**WARNING**

When you perform a DB2 catalog to DB2 catalog comparison, you can use wildcards to select databases for Compare1 or Compare2 on the Compare Catalog to Catalog panel. If you do not specify an outbound migrate profile or do not specify CHANGERULESIN2 NULL, automatic change rules are generated. You must specify the NODATABASE keyword in the ALUIN input stream. CHANGE MANAGER cannot assure the resolution of object names for databases when you use wildcards in the scope for a database.
For more information about improving the performance of a comparison, see “Improving the performance of a comparison” on page 264.

**TIP**
To create an outbound migrate profile, see page 146.

**Specifying the options for a comparison**

After you define the scope of a comparison, you can select the values that are used to process the comparison. You can specify the values for the following Compare options:

- types of dependent objects to include in a CDL file
- object attributes to use in a comparison
- optional outbound migrate profile that contains change rules (which CHANGE MANAGER applies after the comparison)

**Specifying the types of objects**

Compare enables you to specify the object types to include in the comparison. When you specify an object, any changes for that object will be included in the generated CDL. By default, for comparisons other than a DB2 catalog to a DB2 catalog comparison, the following objects are included:

- storage group
- database
- table space
- table
- check constraint
- foreign key
- index
- unique constraint
- view
- trigger
- synonym
- alias
- auxiliary objects
By default, for a DB2 catalog to a DB2 catalog comparison, database, table space, synonym, and alias objects are excluded.

**NOTE**

When you compare an auxiliary table to a table, Compare generates DROP and CREATE statements.

Although you will usually use Compare to evaluate the changes to all of the major object types, situations might arise in which you do not want to compare certain object types. For example, you might have catalog baselines that include all of the object types. This type of baseline is normally used for recovery or change migration. If you want to only compare the objects that an application uses, such as tables, views, synonyms, and aliases, select to include those objects and exclude the others.

When an object type is excluded from the comparison, Compare inserts a keyword in the ALUIN input stream. The following keywords can be included:

- NOSTOGROUP
- NODATABASE
- NOTABLESPACE
- NOTABLE
- NOCHECKCNST
- NOFOREIGNKEY
- NOINDEX
- NOUNIQUECNST
- NOVIEW
- NOTRIGGER
- NOSYNONYM
- NOALIAS
- NOAUXILIARY

**NOTE**

The NOSTOGROUP, NODATABASE, and NOTABLESPACE keywords refer to both the object and the attribute.
Improving performance

Excluding certain object types from the comparison improves performance in the following situations:

- object types have not changed
- changes to the object types are not significant for a comparison
- objects do not need to be kept synchronized

Including storage groups, table spaces, tables, foreign keys, indexes, and views in a comparison requires Compare to access multiple catalog tables. For example, when you specify tables as the object type for a DB2 catalog to a DB2 catalog comparison, Compare retrieves table information from the SYSIBM.SYSTABLES catalog table and table column information from the SYSIBM.SYSCOLUMNS catalog table.

Retrieving table column information can take considerable time with a large catalog. If you do not include tables in the comparison, Compare will not retrieve table column information. Compare only retrieves the table names from the catalogs and uses those names for retrieving dependent object information. Therefore, access to the SYSIBM.SYSCOLUMNS catalog tables is not needed and the performance of the comparison is improved.

**WARNING**

You can specify any combination of object types to include in a comparison. However, ensure that you do not eliminate changes to the object types which you need to synchronize.

**NOTE**

Global temporary tables are included in the comparison if the table (TB) object type is selected.

For more information about improving the performance of a comparison, see “Improving the performance of a comparison” on page 264.

Specifying the object attributes

Compare also enables you to specify the object attributes to include in a comparison. When you select the option **Display a list of object attributes to exclude from the Compare** on the Compare CDL Options panel, the Compare Object Attributes panel is displayed, as shown in Figure 63.
Specifying the object attributes

Figure 63  Compare Object Attributes panel

![Compare Object Attributes panel](image)

**NOTE**

Any changes that you make in the Compare Object Attributes panel are not saved to your ISPF profile.

From the Compare Object Attributes panel, you can specify the following attributes. You must specify the object attributes to include or exclude for each comparison, unless you specify the attributes as suppress-type change rules. One advantage of using change rules is that they can be stored and used again. For more information about using suppress-type change rules, see “Specifying an outbound migrate profile” on page 253.

- **Comments/Remarks** indicates changes to comments and labels for an attribute. If this attribute is excluded, Compare inserts the NOREMARKS keyword into the ALUIN input stream.

- **Locksize/Lockmax** indicates the LOCKSIZE attribute for table spaces. If this attribute is excluded, Compare inserts the NOLOCKSIZE keyword into the ALUIN input stream. This keyword suppresses both the LOCKMAX and the LOCKSIZE attributes.

- **Segsize**, which indicates the SEGSIZE attribute for segmented table spaces. If this attribute is excluded, Compare inserts the NOSEGSIZE keyword into the ALUIN input stream.

- **Define** indicates the DEFINE attribute for table spaces and indexes. If this attribute is excluded, Compare inserts the NODEFINE keyword into the ALUIN input stream.
Specifying the object attributes

- **Bufferpool** indicates the BUFFERPOOL attribute for databases, table spaces, and indexes. If this attribute is excluded, Compare inserts the NOBUFFERPOOL keyword into the ALUIN input stream.

- **Check Option** indicates the CHECK attribute for views. If this attribute is excluded, Compare inserts the NOVIEWCHECK keyword into the ALUIN input stream.

- **Drop Restrict** indicates the DROPRESTRICT attribute for tables. If this attribute is excluded, Compare inserts the NODROPRESTRICT keyword into the ALUIN input stream.

- **Path** indicates the PATH attribute for check constraints, views, and triggers. If this attribute is excluded, Compare inserts the NOPATH keyword into the ALUIN input stream.

- **Startvalue** indicates the START attribute for an identity column in a table. If this attribute is excluded, Compare inserts the NOSTARTVALUE keyword into the ALUIN input stream.

- **Identity** indicates the IDENTITY attribute for tables. If this attribute is excluded, Compare inserts the NOIDENTITY keyword into the ALUIN input stream.

- **Partition attributes** indicates the following types of attributes for partitioned and nonpartitioned table spaces, tables, and indexes:
  
  — attributes that are part of the USING clause
  — NUMPARTS and STOGROUP attributes
  — (partitioned indexes only) index limit key ranges

If you exclude the Partition attributes option, Compare inserts the NOPARTITION keyword into the ALUIN input stream, and the comparison excludes these attributes.

---

**NOTE**

For more information, view the Quick Course Using NOPARTITION. You must have a BMC Support ID to view the Quick Course.

---

If partition attributes are excluded from the comparison, Compare does not compare attributes of the following items:

- any table space, table, or index that are a part of the USING clause

- the NUMPARTS and STOGROUP attributes of any partitioned table space, table, or index

- index limit key range values of any partitioned index
In contrast, when you include Partition attributes, you can choose to exclude individual attributes from the comparison (for example, COMPRESS, FREEPAGE, PCTFREE, PRIQTY, SECQTY, and TRACKMOD). In that case, Compare inserts the appropriate keyword into the ALUIN input stream:

— NOCOMPRESS
— NOFREEPAGE
— NOPCTFREE
— NOPRIQTY
— NOSECQTY
— NOTRACKMOD

Specifying an outbound migrate profile

The outbound migrate profile that you specify in the Compare CDL Options panel can contain change rules and locations. These rules and locations can be used to filter out specific changes in a CDL file, generate multiple CDL files, and change the attributes of objects. CHANGE MANAGER applies the rules in the outbound migrate profile when it generates CDL (after the comparison is performed). When you specify an outbound migrate profile, Compare inserts the CDLCHANGERULES keyword into the ALUIN input stream.

Tip

To create an outbound migrate profile, see page 146.

Filtering changes

You can filter out specific changes in a CDL file by using suppress-type change rules in an outbound migrate profile. Suppress-type rules affect only altered objects and attributes.

One advantage of using suppress-type rules is that you can tailor the types of changes that you want to filter out for each receiving location. For example, you can suppress a change to the primary quantity of a partition for one location while allowing it for another location. You can use a suppress-type change rule that specifies not to generate CDL for a primary quantity change to a particular table space or index. Because CHANGE MANAGER generates CDL that specifies renamed table spaces or indexes with ALTER commands rather than with DROP and CREATE commands, it can apply the suppress-type change rule to the table space or index and show only the name change in the CDL. Changes to the primary quantity are not specified.
You can only use suppress-type change rules with altered objects. Using these change rules preserves modifications to your local subsystem. For a list of the objects and the attributes of the objects that you can use with change rules, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

### Generating multiple CDL files and changing object attributes

You can also use an outbound migrate profile to generate multiple CDL files (one per location) or to change the attributes of objects through the use of change-type change rules. The CDL for each location uses the ddname CDLnnn, where nnn is a three-digit, sequential number beginning with 001 for the first file. For example, if the profile contains three locations, the ddnames are CDL001, CDL002, and CDL003. If locations are grouped together using the group ID in the outbound migrate profile, Compare generates a single CDL file per group ID. If some locations have a group ID while other locations do not, Compare generates one CDL file per group ID and one file for each location whose group ID is blank. You can also use the MIGLOCATIONS keyword to restrict the set of CDL files that are generated.

---

**NOTE**

If you generate CDL using an outbound migrate profile with locations, you cannot edit the CDL in *CHANGE MANAGER*.

---

### Overriding the default values for a comparison

Before you create the JCL or the input for a comparison, you can override the default values for the following options:

- data set allocation parameters
- types of objects and statements to include in a CDL file
- whether the change rules are displayed
- whether to keep the original owner of an object when *CHANGE MANAGER* builds automatic change rules

You use the options on the Compare Override Options panel to override the default values. You can also change some of these options by using keywords in the ALUIN input stream.
Allocating data sets

You can override the default data set allocation parameters for the CDL file. The parameters, which can only be used for new data sets, are the name of the unit to which the CDL will be written, the volume, and the primary and secondary quantities (in tracks).

Including statements and dependent objects

The Compare component can include specific CDL statement types and changes to dependent objects in the CDL file. When you specify to include these items, Compare inserts the INCLUDE keyword in the ALUIN input stream, along with the appropriate parameters.

Including CDL statement types

When you specify to include a CDL statement type, Compare includes the -CDL commands for created, altered, or dropped objects. Compare also inserts the INCLUDE keyword with the CREATE, ALTER, or DROP parameter into the ALUIN input stream, as shown in Figure 64.

Figure 64  Use of ALTER, CREATE, and DROP in the ALUIN input stream

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>BASELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1</td>
<td>BMC.BLUND201102110001</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>LOCAL</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>(ALTER CREATE DROP)</td>
</tr>
</tbody>
</table>

Compare includes -CDL commands for altering objects when you select the ALTER statements option, except for object dependencies (those objects that are only generated due to a name change of a parent object). For example, if the column name in two tables is different, then base views for those tables will not be included in the CDL. DROP and CREATE commands for some foreign keys and index changes are generated in the CDL file when ALTER is specified.
Analysis propagates object name changes to some dependent objects. Compare always includes the -CDL commands for altering the following objects:

- storage groups
- databases, table spaces, and indexes

If a change is made to the name of a storage group that affects these objects, Compare includes the -CDL commands.

views

Including dependent objects

When you specify to include dependent objects, Compare includes -CDL commands for dependent objects and the commands that are only generated due to a name change of a parent object. Compare also generates a complete record of the changes to the objects (and their dependents) that are specified within the scope of the comparison. For example, if you select this option and a table name within the scope of comparison is changed, every dependent object for the specified table is included in the generated CDL file.

**NOTE**

If you do not specify this option, -CDL commands are not generated for object changes that the Analysis component propagates. As a result, you will not obtain a complete record of changes until the worklist is generated.

Compare inserts the INCLUDE keyword with the CDLDEP parameter into the ALUIN input stream when this option is selected. An example of the ALUIN input stream is shown in Figure 65.

**Figure 65  Use of CDLDEP in the ALUIN input stream**

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>BASELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1</td>
<td>RDACRJ.BL01</td>
</tr>
<tr>
<td>LOCATION1</td>
<td>DEBA</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>BASELINE</td>
</tr>
<tr>
<td>CMPIN2</td>
<td>RDACRJ.BL02</td>
</tr>
<tr>
<td>LOCATION2</td>
<td>DEAH</td>
</tr>
<tr>
<td>CHANGERULESIN2</td>
<td>RDACEL.C7LBA1$C7LBD1</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>(CDLDEP)</td>
</tr>
<tr>
<td>MIGLOCATIONS</td>
<td>(DALLAS CDL001, HOUSTON CDL002)</td>
</tr>
</tbody>
</table>
Displaying change rules

You can choose whether to display change rules when you have specified an outbound migrate profile that contains change rules (indicated by the CHANGERULESIN2 keyword) for the secondary input (Compare2) in a comparison. Selecting this option also displays the automatic change rules that are created when you do not specify an outbound migrate profile in a DB2 catalog to a DB2 catalog comparison. When you select this option, Compare inserts the SHOWRULES keyword into the ALUIN input stream.

Keeping the original owner name for an object

If you are comparing a DB2 catalog to another DB2 catalog and you do not specify an outbound migrate profile (indicated by the CHANGERULESIN2 keyword), CHANGE MANAGER might be unable to resolve the owner of an object or the database name for a table space or table when it builds automatic change rules. (For information about automatic change rules, see “Resolving changes to names” on page 245.)

Select the Retain original owners in Compare2 objects option to retain the database or owner name in Compare2 if CHANGE MANAGER cannot determine it. When you select this option, Compare inserts the DEFAULTOFF keyword into the ALUIN input stream. If you do not select this option, the owner or database name for the created objects is replaced with a set of pound (#) signs.
Generating the JCL or the input for the comparison

After you specify the options for your comparison, you generate the JCL for the comparison. Table 22 lists the DD names that are used in the JCL.

Table 22  DD Names used in the comparison JCL

<table>
<thead>
<tr>
<th>DD Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1</td>
<td>(optional) the names of the DDL or migrate-type worklist input data sets</td>
</tr>
<tr>
<td>ALUPRINT</td>
<td>the name of the Compare diagnostic output log</td>
</tr>
<tr>
<td>CDL001</td>
<td>the name of the CDL file that is created by Compare</td>
</tr>
<tr>
<td>ALUIN</td>
<td>the name of the parameter input stream</td>
</tr>
</tbody>
</table>

The CDL file can be either a sequential data set or a member of a partitioned data set. If more than one output CDL data set is generated, DD names are created sequentially, starting with CDL001. Compare requires output CDL data sets to use 80-character records.

Generating CDL

To generate the CDL, either submit the JCL or run Compare. When CDL is generated, Compare validates view and trigger text and manages renamed objects.

Validation of object text dependencies

When CHANGE MANAGER generates CDL, Compare validates text for the following objects when included in the scope of the comparison:

- views
- triggers
- check constraints
- indexes on an expression
- materialized query tables (MQTs)
This feature performs the following functions:

- parses the object for object and attribute dependencies
- applies change rules for name changes to objects and columns

For more information, see “Validity and performance checks” on page 73.

Management of renamed objects

When a baseline is used as one of the inputs in a comparison, Compare uses the rename table to resolve renamed objects. The CDL that CHANGE MANAGER generates correctly retains data with objects and table columns and propagates name changes to columns of dependent views. Compare resolves renamed objects, regardless of how many times the names are changed, as long as the name changes were made through either ALTER or CHANGE MANAGER. In this case, the rename information is recorded in the CHANGE MANAGER rename table.

Several CHANGE MANAGER components are involved in managing renamed objects. You can use Specification or Import to create change requests in an alter-type work ID that specify changes to object names, table columns, or view columns. When Analysis processes this alter-type work ID, it generates an -RNAM command in the worklist for each renamed object, table column, or view column. When Execution processes the worklist, it uses the information provided by the -RNAM commands, along with time stamps of when name changes occurred (or were executed). A record of all of the name changes that have been applied and processed is kept.

When a comparison is run with a baseline as one input, the comparison uses the time stamps of both sets of input to select the rename table entries that have occurred within the time frame of the two inputs. The rename information is applied as needed to resolve name changes. A time stamp is stored when a baseline is created and when an -RNAM command is executed.

**NOTE**

If you have baselines that were created with a prior version of CHANGE MANAGER and that contain views whose column names have changed since the baseline was taken, it is recommended that you create a new baseline with the current version. You can also use the -RNAM worklist command (specifying VC for the object type) for updating rename information. For more information, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

When you use the DB2 catalog, a DDL file, or a migrate-type worklist as one of the inputs, the time stamp of these sources is considered to be the current time, and name changes are applied from the time the baseline was created up to the current time.
Resolution of renamed objects

The following example shows how renamed objects are resolved:

You create a baseline that includes a table named DCH.TEST. At a later time, you rename the table to DCH.PROD using an alter-type work ID. You then generate CDL that shows the changes to DCH.TEST since the baseline was created. The ALUIN input stream contains the keywords shown in Figure 66 for a catalog baseline to a DB2 catalog comparison:

Figure 66  Example ALUIN input stream

<table>
<thead>
<tr>
<th>CMPTYPE1</th>
<th>BASELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1</td>
<td>ACM.BLUNLD201102120001</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>LOCAL</td>
</tr>
</tbody>
</table>

Using this input stream, Compare performs the following functions:

1. loads the baseline and the time stamp that is named ACM.BLUNLD200104170001

2. retrieves the current time stamp and object definitions from the DB2 catalog by using the same scope that was specified to create the baseline

3. using the baseline time stamp and the current time stamp, Compare retrieves the rename information about the table from the CHANGE MANAGER rename table that falls between the two time stamps and applies it to the table definition that was previously retrieved from the DB2 catalog

   The changes to the name of the objects are maintained by CHANGE MANAGER using the rename table.

4. generates an ALTER command in the CDL file that shows the renaming of the table, plus any additional changes that might have been applied to the table since the baseline was created

If CHANGE MANAGER was not used to rename the table, Compare would treat DCH.TEST and DCH.PROD as two different objects. It would then generate a DROP command for DCH.TEST and a CREATE command for DCH.PROD in the CDL file. Data will be lost unless you manually update the rename table.

Rename resolution is performed regardless of the time of the comparison. That is, you can perform a roll back comparison that uses the DB2 catalog as the primary input (Compare1) and a baseline as the secondary input (Compare2). If this comparison includes name changes, the objects are renamed back to their old names as specified in the baseline. If either of the Compare inputs are DDL baselines, the correct rename information is probably not in the rename table unless it was manually inserted.
Rename support is not as significant for some objects because no possibility for data loss exists. For example, CDL could be generated to specify changes to index names with DROP and CREATE commands rather than with ALTER commands. But then you would not know if other index attributes also changed because the CREATE command specifies all of the index attributes. The ALTER command specifies only the changed index attributes, which usually results in generating less CDL and might be important when Compare is used for impact analysis.

**Considerations for renamed objects**

Consider the following items when you plan to rename objects or when you work with renamed objects:

- Use CHANGE MANAGER to change object names. Using CHANGE MANAGER enables Compare to resolve renamed objects when they are used in a comparison and eliminates the need to manually update the rename table.

- If you have a set of data structures that contain renamed objects and you have a choice of sending either CDL or DDL to another subsystem, send CDL. Assuming that Compare generates CDL ALTER commands for the renamed objects, CDL is easier to apply on the receiving subsystem. If you send DDL that contains renamed objects, the receiver of the DDL must know the objects that have been renamed and the receiver must manually perform the process of retaining the data that is associated with the renamed objects. If you send DDL, your local modifications might not be retained.

- If you receive DDL that is generated from data modeling tools and that contains renamed objects, you need to create and retain baselines of the DDL and manually update the CHANGE MANAGER rename table for the objects that have been renamed. If possible, request that the sender of the DDL provide a listing of the renamed objects in a worklist format.

- Compare only resolves renamed objects when a baseline is involved in the comparison and the renamed objects were correctly recorded in the CHANGE MANAGER rename table.

**Results of a comparison**

After you perform a comparison of two input structures, Compare can generate the following output:

- a diagnostic log, which shows the status of each phase of a comparison

- a CDL file, which contains commands that show the differences between the two input structures
Optionally, the CDL file can contain comparison report information (as comments), which can be used to determine how the objects in a structure will be affected if the CDL is used to synchronize or version your data structures.

**Diagnostic log**

Compare produces diagnostic output in the ALUPRINT file. The diagnostic output might be assigned to TERM (in foreground), assigned to SYSOUT, or sent to a sequential MVS file. It cannot be assigned to a member of a partitioned data set.

The diagnostic log contains header information, keywords in the ALUIN input stream, and maintenance information. The log also contains messages and return codes which indicate the status of the various processing phases. Table 23 lists some of the return codes that might appear in the diagnostic log.

<table>
<thead>
<tr>
<th>Return Code (RC)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no errors or warnings were issued</td>
</tr>
<tr>
<td>1</td>
<td>no differences were found in the comparison process; CDL was not generated</td>
</tr>
<tr>
<td>4</td>
<td>warnings were issued</td>
</tr>
<tr>
<td>8 or 12</td>
<td>errors were issued; Compare terminated</td>
</tr>
</tbody>
</table>

**CDL file**

A CDL file contains the CDL commands that show the differences between two sets of data structures. The file is a record of the changes that would be made if you import the file to a work ID, run Analysis, and then run Execution to update a version of the data structures.

The CDL is contained in an 80-column sequential data set or a PDS member. Every CDL file begins with the -TIME, -ORGN, and -DEST commands followed by zero or more -CDL commands. Each -CDL command contains a CDL statement. The *ALTER and CHANGE MANAGER for DB2 Reference Manual* contains a complete description of the CDL statements, as well as information about the format of the CDL file.
A CDL file can also contain header, detailed, and summary report information as comments. The comments include all of the ALTER statements for both old and new values. Comments are indicated by an asterisk and eight dashes (* - - - - - - - -). The comparison report information can help you evaluate the effects of applying the CDL that a comparison generates. The comments are included in the CDL when the REPORT (DETAIL) keyword and parameter are included in the ALUIN input stream.

A CDL file can also contain header, detailed, and summary report information as comments. The comments include all of the ALTER statements for both old and new values. Comments are indicated by an asterisk and eight dashes (* - - - - - - - -). The comparison report information can help you evaluate the effects of applying the CDL that a comparison generates. The comments are included in the CDL when the REPORT (DETAIL) keyword and parameter are included in the ALUIN input stream.

You can edit a CDL file with the standard ISPF editor or with any other text editor.

Auxiliary objects and nonauxiliary objects are listed in the same section of the file. For example, auxiliary table space and nonauxiliary table space commands are listed in the table space commands section.

Generating a CDL file from an alter-type work ID

When Compare generates CDL from an alter-type work ID, the CDL acts as a record of the changes made to the local subsystem. The CDL can also be used to transmit those changes to another subsystem or to transmit inter-version changes to remote subsystems. Only those changes that are specified in the work ID and that result in CDL statements are included in the CDL file. CDL statements that result from change propagation by Analysis are not included. This process enables you to retain local modifications.
Applying the CDL

For example, if you change the name of a table column in a work ID on a local subsystem, the CDL file contains only a single ALTER TABLE ... COLUMN statement for that change. If the column was used in an index, the column name change is propagated to the index by Analysis and it is not included in the CDL file. However, more than one index on the remote subsystem might be affected by this change. In this case, Analysis will propagate the name change to the indexes. The CDL file will not include the name changes for the indexes.

**TIP**
To generate CDL from a work ID, see page 297.

### Applying the CDL

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem. You can also import the CDL file to an alter-type work ID on a different subsystem to update a separate version of the data structures. To import the CDL file and update your data structures, see Chapter 4, “Altering data structures in a database environment.”

### Improving the performance of a comparison

If the scope of an application contains about 20,000 or more objects, performing a comparison can consume a considerable amount of CPU time, memory, and disk space. To minimize the amount of resources used for the comparison, perform the following steps:

1. Create or verify indexes on the DB2 catalog tables shown in Table 24.

<table>
<thead>
<tr>
<th>Table</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIBM.SYSFIELDS</td>
<td>TBCREATOR, TBNAME, NAME</td>
</tr>
<tr>
<td>SYSIBM.SYSFOREIGNKEYS</td>
<td>CREATOR, TBNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSSYNONYMS</td>
<td>TBCREATOR, TBNAME</td>
</tr>
<tr>
<td>SYSIBM.SYSTABLES</td>
<td>DBNAME, TSNAME, TYPE</td>
</tr>
</tbody>
</table>
Creating these indexes enables Compare to use indexes to retrieve specific objects. If the indexes do not exist, Compare might need to perform table space scans, which can consume a great deal of CPU time. For more information about creating indexes, see the Installation System User Guide and the BMC Products and Solutions for DB2 Configuration Guide.

2. Consider excluding any DB2 object types that are not necessary from the comparison. For example, on the Compare CDL Options panel, you can type N for VW to exclude views from the comparison.

3. Avoid the use of automatic change rules in a DB2 catalog to DB2 catalog comparison. Instead, specify change rules in an outbound migrate profile that is applied before the comparison (with the CHANGERULESIN2 keyword) or specify CHANGERULESIN2 NULL.

   - If the database, creator, owner, and schema names are the same for Compare1 and Compare2, specify CHANGERULESIN2 NULL.

   - If the database, creator, owner, and schema names are different for Compare1 and Compare2, specify an existing outbound migrate profile with change rules (CHANGERULESIN2) to be applied to Compare2.

   When CHANGE MANAGER creates automatic change rules, it generates at least one change rule for each object in the comparison. Message BMC56388I in the diagnostic log provides you with the number of change rules that are generated. While the memory that is used to store these change rules is somewhat significant, the CPU time that is expended to apply these rules is considerable.

4. Modify the value of the THRESHOLD keyword in the ALUIN input stream to increase the threshold for the number of objects that are pulled from the DB2 catalog.

   The THRESHOLD keyword specifies the integer that limits the number of objects that are pulled from the DB2 catalog using the nonunique catalog indexes. If the number of objects that are requested exceeds the limit, the nonunique catalog indexes are ignored. The relevant table space in the catalog is then scanned for the requested objects. For example, if the THRESHOLD is set to 175, and you request 200 objects, the product fully scans the table space instead of opening 200 DB2 cursors.

   In general, the THRESHOLD keyword can be set to a very high value. The value is set to 100,069 by default.
5. If possible, use the fastest input source that is available for your comparisons. The following sources of input are available (shown in order of speed, with the fastest listed first):

- DDL file or migrate-type worklist
- baseline
- DB2 catalog

Accomplishing your goals

CHANGE MANAGER enables you to compare two data structures so that you can synchronize or version those structures. Table 25 lists the tasks that you can perform to synchronize or version your data structures.

Table 25  Comparison tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Comparing a DB2 catalog to a DB2 catalog”</td>
<td>page 266</td>
</tr>
<tr>
<td>“Using a profile to scope a catalog to a catalog comparison”</td>
<td>page 272</td>
</tr>
<tr>
<td>“Comparing a DB2 catalog to a DDL file”</td>
<td>page 276</td>
</tr>
<tr>
<td>“Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist”</td>
<td>page 280</td>
</tr>
<tr>
<td>“Comparing a baseline to a baseline”</td>
<td>page 283</td>
</tr>
<tr>
<td>“Comparing a baseline to a DB2 catalog”</td>
<td>page 287</td>
</tr>
<tr>
<td>“Comparing a DB2 catalog to a baseline”</td>
<td>page 290</td>
</tr>
<tr>
<td>“Comparing a baseline to a DDL file”</td>
<td>page 294</td>
</tr>
<tr>
<td>“Generating a CDL file from a work ID”</td>
<td>page 297</td>
</tr>
<tr>
<td>“Using a script to perform a comparison”</td>
<td>page 300</td>
</tr>
</tbody>
</table>

Comparing a DB2 catalog to a DB2 catalog

You can compare two sets of data structures that are contained in two different catalogs. You can use this type of comparison to synchronize copies of an application under development or to compare a small subset of an application’s objects or individual objects. The example used in this set of steps compares a local DB2 catalog to a remote DB2 catalog. In addition, the scope of the comparison was defined explicitly by specifying the object type and object name for Compare1 and Compare2 on the Compare Catalog to Catalog panel.
1 On the CHANGE MANAGER Main Menu, select **Compare (CDL Build)**, and press Enter.

   The Compare Type Selection panel is displayed.

2 Select **Catalog** for both **Compare1 Type** and **Compare2 Type**. Press Enter.

   The Compare Catalog to Catalog Scope Selection panel is displayed.

3 Select **One catalog object and all of its dependents** as the source of the scope. Press Enter.

   **NOTE**

   To select multiple catalog objects, specify scope rules in a baseline profile or an outbound migrate profile and select to use the profile as the source of the scope. For information, see “Using a profile to scope a catalog to a catalog comparison” on page 272.

   The Compare Catalog to Catalog panel is displayed, as shown in **Figure 67**.

**Figure 67  Compare Catalog to Catalog panel**

| ACMF9CMC ---------------- Compare Catalog to Catalog ------------------ |
| Command ===> |
| CDL will be generated to make Compare1 look like Compare2. |
| Object Type ( SG DB TS TB IX VW SY AL SP )   DB |
| Display list of objects (Y/N) . . . . . . . N |
| Type the values for Compare1, where the resulting CDL will be applied. |
| Name - part 1 . . . . . . . . . . DEMOCJ |
| Name - part 2 . . . . . . . . . . |
| Name - part 3 . . . . . . . . . . |
| Optional table owner (to limit scope)... * |
| Optional location name for remote DB2 .. |

   Type the values for Compare2, where the objects already have the desired structure definitions. Then press Enter.

   | Name - part 1 . . . . . . . . . . DEMOME |
   | Name - part 2 . . . . . . . . . . |
   | Name - part 3 . . . . . . . . . . |
   | Optional table owner (to limit scope)... * |
   | Optional location name for remote DB2 .. |
   | Optional outbound migrate profile (owner.name) containing change rules to apply BEFORE the Compare (CHANGERULESIN2) |

4 Type the abbreviation for an object type.

5 If you plan to use a wildcard for **Name - part 1**, **Name - part 2**, or **Name - part 3**, type **Y** to display a list of objects in a mixed list.
Comparing a DB2 catalog to a DB2 catalog

6 Specify the values for Compare1.

A Type the name of the object.

- If the name of the object has only one part, for Name - part 1, type the name or owner, or type a wildcard to specify an object from a list of objects.

- If the name of the object has two parts, specify the following information:

  - For Name - part 1, type the name or owner, or type a wildcard to specify an object from a list of objects.

  - For Name - part 2, type the name of a single object or type a wildcard to specify an object from a list of objects.

- If the name of the object has three parts (such as a stored procedure), specify the following information:

  - For Name - part 1, type the name or owner, or type a wildcard to specify an object from a list of objects.

  - For Name - part 2, type the name of a single object or type a wildcard to specify an object from a list of objects.

  - For Name - part 3, type the name of an object or type a wildcard to specify an object from a list of objects.

B (optional) To limit the scope of the comparison, type the name of the table owner.

C (optional) Specify the name of the location of the remote DB2 subsystem. As an alternative, you can type a wildcard for the location name to generate a list of remote locations from which you can select.

7 Specify the values for Compare2.

A Repeat step 6.

B (optional) Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

8 Press Enter.
If you chose to display a list of objects in a mixed list in step 7, the Mixed List panel is displayed. Otherwise, the Compare CDL Options panel is displayed (proceed to step 11).

9 From the Mixed List panel, you can perform one of the following actions:

- Type **S** to select a single object to use in the comparison. Then, press **Enter**.
- Press **END** to exit the mixed list. All of the objects listed on the mixed list will be included in the comparison.

10 Press **Enter**.

The Compare CDL Options panel is displayed.

11 On the Compare CDL Options panel, specify the object types and attributes for the comparison.

- Type **Y** or **N** for each object type that you want to include in or exclude from the comparison.
- Type **S** to display a list of the object attributes to include in or exclude from the comparison.
- Type **S** to generate comparison report information as comments in the CDL file.
- **(optional)** Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).
- Select a Run Type.

Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

12 On the Compare Object Attributes panel, specify the object attributes.

- To include an object attribute in the comparison, type **Y** adjacent to the attribute.
- If you specified to include the partition attributes in step 12A, you can exclude one or more of the partition attributes. Type **N** adjacent to the attributes that you want to exclude.
Comparing a DB2 catalog to a DB2 catalog

C Press Enter to return to the Compare CDL Options panel.

13 Press Enter.

The Compare JCL Processing Interface panel is displayed.

14 On the Compare JCL Processing Interface panel, specify the data set names to be used in the comparison.

A *(batch run type)* Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.

C Type one of the following options for Diagnostics:

- *(foreground run type)* To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D *(foreground run type)* If you specified SYSOUT in step 14C, specify the Sysout Class.

15 Type S to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process.

16 Press Enter.

The Compare Override Options panel is displayed.

17 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the Unitname, Volume, Priqty, or Secqty.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select CDLDEP.

D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for Display change rules applied to Compare2 BEFORE the Compare.

E Press END to return to the Compare JCL Processing Interface panel.
Select your options for the comparison.

A To create the input for the comparison, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

B To review or modify input to the comparison process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

Figure 68 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

Figure 68  ALUIN input stream—DB2 catalog to DB2 catalog

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>LOCAL</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>REMOTE</td>
</tr>
<tr>
<td>LOCATION2</td>
<td>DBDC</td>
</tr>
<tr>
<td>SCOPETYPE</td>
<td>RULE</td>
</tr>
<tr>
<td>INCLUDE(</td>
<td>ALTER CREATE DROP)</td>
</tr>
<tr>
<td>SCOPERULE1</td>
<td>(* IX ACMB3.*) ( )</td>
</tr>
<tr>
<td>SCOPERULE2</td>
<td>(* IX ACMB3.*) ( )</td>
</tr>
<tr>
<td>CHANGERULESIN2</td>
<td>NULL</td>
</tr>
<tr>
<td>DEFAULTOFF</td>
<td></td>
</tr>
<tr>
<td>NOSTOGROUP</td>
<td></td>
</tr>
<tr>
<td>NODATABASE</td>
<td></td>
</tr>
<tr>
<td>NOTABLESPACE</td>
<td></td>
</tr>
<tr>
<td>*-NOINDEX</td>
<td></td>
</tr>
<tr>
<td>NOTABLE</td>
<td></td>
</tr>
<tr>
<td>NOCHECKCNST</td>
<td></td>
</tr>
<tr>
<td>NOFOREIGNKEY</td>
<td></td>
</tr>
<tr>
<td>NOUNIQUECNST</td>
<td></td>
</tr>
<tr>
<td>NOVIEW</td>
<td></td>
</tr>
<tr>
<td>NOSYNONYM</td>
<td></td>
</tr>
<tr>
<td>NOALIAS</td>
<td></td>
</tr>
<tr>
<td>NOTRIGGER</td>
<td></td>
</tr>
<tr>
<td>NOAUXILIARY</td>
<td></td>
</tr>
<tr>
<td>NODAONLY</td>
<td></td>
</tr>
</tbody>
</table>
Using a profile to scope a catalog to a catalog comparison

C To run the comparison, select one of the following options:

- (foreground run type) Type S to select Run Compare.
- (batch run type) Type S to select Submit JCL.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

D To edit the CDL file that is generated by the comparison, type S to select Edit CDL File.

---

WARNING

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

---

E Press Enter.

Where to go from here

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.

TIP

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

---

Using a profile to scope a catalog to a catalog comparison

You can compare two sets of data structures that are contained in two different catalogs. The example provided in this set of steps uses the scope that is defined for an outbound migrate profile or a catalog baseline profile to select a collection of unrelated objects and their dependents.

1 On the CHANGE MANAGER Main Menu, select Compare (CDL Build), and press Enter.

The Compare Type Selection panel is displayed.

2 Select Catalog for both Compare1 Type and Compare2 Type. Press Enter.

The Compare Catalog to Catalog Scope Selection panel is displayed.
3 Select **Baseline profile** or **Outbound migrate profile** as the source of the scope. Press **Enter**.

The Compare Catalog to Catalog Using Profiles panel is displayed.

4 Specify the values for the inputs to the comparison.

A Type the name of the location of a catalog for Compare1. The local DB2 catalog is the default.

B Type the name of the baseline profile for Compare1 or type a wildcard pattern to display a list of profiles.

You can specify a remote SSID for the baseline profile. To do so, you must edit the ALUIN input stream (see step 14B on page 275); you cannot specify the SSID on the Compare Catalog to Catalog Using Profiles panel. In the SCOPE1 keyword, specify the SSID in the first part of a three-part name.

C Type the name of the location of a catalog for Compare2. The local DB2 catalog is the default.

D Type the name of the baseline profile for Compare2 or type a wildcard pattern to display a list of profiles.

You can specify a remote SSID for the baseline profile. To do so, you must edit the ALUIN input stream (see step 14B on page 275); you cannot specify the SSID on the Compare Catalog to Catalog Using Profiles panel. In the SCOPE2 keyword, specify the SSID in the first part of a three-part name.

E *(optional)* Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

F Press **Enter**.

The Compare CDL Options panel is displayed.

5 Type **Y** or **N** for each object type that you want to include in or exclude from the comparison.

6 Type **S** to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.
7 If you selected to display a list of the object attributes, specify the object attributes.

A To include an object attribute in the comparison, type Y adjacent to the attribute.

B If you specified to include the partition attributes in step 12A, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

C Press END to return to the Compare CDL Options panel.

8 Type S to generate comparison report information as comments in the CDL file.

9 (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

10 Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

The Compare JCL Processing Interface panel is displayed.

11 Specify the data set names to be used in the comparison.

A (batch run type) Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.

C Type one of the following options for Diagnostics:

   ■ (foreground run type) To display the diagnostics on the terminal, type TERM.
   ■ To write the diagnostics to a sequential file, type the name of the data set.
   ■ To write the diagnostics to a print data set, type SYSOUT.

D (foreground run type) If you specified SYSOUT in step 11C, specify the Sysout Class.

12 Type S to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

The Compare Override Options panel is displayed.
13 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the Unitname, Volume, Priqty, or Secqty.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select CDLDEP.

D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for Display change rules applied to Compare2 BEFORE the Compare.

E Press END to return to the Compare JCL Processing Interface panel.

14 Select your options for the comparison.

A To create the input for the comparison, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

B To review or modify input to the comparison process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

Figure 69 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

Figure 69   ALUIN input stream—DB2 catalog to DB2 catalog with profiles

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>LOCAL</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>LOCAL</td>
</tr>
<tr>
<td>SCOPE1</td>
<td>RDACRJ.OMP1</td>
</tr>
<tr>
<td>SCOPE2</td>
<td>RDACRJ.OMP2</td>
</tr>
<tr>
<td>INCLUDE (ALTER CREATE DROP )</td>
<td></td>
</tr>
<tr>
<td>NODEDEFINE</td>
<td></td>
</tr>
</tbody>
</table>
Comparing a DB2 catalog to a DDL file

C To run the comparison, select one of the following options:

- (foreground run type) Type S to select Run Compare.
- (batch run type) Type S to select Submit JCL.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

D To edit the CDL file that is generated by the comparison, type S to select Edit CDL File.

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

E Press Enter.

Where to go from here

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.

**TIP**

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Comparing a DB2 catalog to a DDL file

This task uses a DDL scope type to compare a local DB2 catalog to a DDL file. The DDL scope type allows only those objects that match the objects in the DDL to be retrieved from the catalog for comparison. The comparison of a DB2 catalog to a DDL file is useful in generating CDL for incremental changes.

1 On the CHANGE MANAGER Main Menu, select Compare (CDL Build), and press Enter.

The Compare Type Selection panel is displayed.

2 Select Catalog for the Compare1 Type and select DDL for the Compare2 Type.

The Compare Scope Selection panel is displayed.
3 To specify the type of scope to use for the comparison, select **DDL or Worklist file**.

Press *Enter*.

The Compare Catalog to DDL panel is displayed.

4 Specify the values for the inputs to the comparison.

A Type the name of the location of a catalog for Compare1. The local DB2 catalog is the default.

B Type the data set name of the DDL file for Compare2.

C *(optional)* Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

D Press *Enter*.

The Compare CDL Options panel is displayed.

5 Type **Y** or **N** for each object type that you want to include in or exclude from the comparison.

6 Type **S** to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

7 If you selected to display a list of the object attributes, specify the object attributes.

A To include an object attribute in the comparison, type *Y* adjacent to the attribute.

B If you specified to include the partition attributes in step 7A, you can exclude one or more of the partition attributes. Type *N* adjacent to the attributes that you want to exclude.

C Press END to return to the Compare CDL Options panel.

8 Type **S** to generate comparison report information as comments in the CDL file.

9 *(optional)* Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).
Comparing a DB2 catalog to a DDL file

10 Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

The Compare JCL Processing Interface panel is displayed.

11 Specify the data set names to be used in the comparison.

A (batch run type) Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.

C Type one of the following options for Diagnostics:

- (foreground run type) To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D (foreground run type) If you specified SYSOUT in step 11C, specify the Sysout Class.

12 Type S to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

The Compare Override Options panel is displayed.

13 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the Unitname, Volume, Priqty, or Secqty.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select CDLDEP.

D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for Display change rules applied to Compare2 BEFORE the Compare.

E Press END to return to the Compare JCL Processing Interface panel.
14 Select your options for the comparison.

A To create the input for the comparison, select one of the following options:

- *foreground run type* Type S to select Create Input.
- *batch run type* Type S to select Create JCL.

B To review or modify input to the comparison process, select one of the following options:

- *foreground run type* Type S to select Edit Input.
- *batch run type* Type S to select Edit JCL.

Figure 70 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

Figure 70 ALUIN input stream—DB2 catalog to a DDL file

<table>
<thead>
<tr>
<th>SSID DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1 LOCAL</td>
</tr>
<tr>
<td>CMPTYPE2 DDL</td>
</tr>
<tr>
<td>SCOPETYPE DDL</td>
</tr>
<tr>
<td>INCLUDE (ALTER CREATE DROP )</td>
</tr>
<tr>
<td>NODEFINE</td>
</tr>
</tbody>
</table>

C To run the comparison, select one of the following options:

- *foreground run type* Type S to select Run Compare.
- *batch run type* Type S to select Submit JCL.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

D To edit the CDL file that is generated by the comparison, type S to select Edit CDL File.

---

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

E Press Enter.
Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist

**Where to go from here**

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.

**TIP**
To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist**

You can compare two DDL files. You can use this type of comparison to maintain versions of your DDL or to see the differences in your DDL files.

1 On the CHANGE MANAGER Main Menu, select **Compare (CDL Build)**, and press **Enter**.

   The Compare Type Selection panel is displayed.

2 Select **DDL** for both the **Compare1 Type** and the **Compare2 Type**. Press **Enter**.

   The Compare DDL to DDL panel is displayed.

3 Specify the values for the inputs to the comparison.

   A Type the data set name of the DDL file for Compare1.

   B Type the data set name of the DDL file for Compare2.

   C *(optional)* Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 **before** the comparison.

   D Press **Enter**.

   The Compare CDL Options panel is displayed.

4 Type **Y** or **N** for each object type that you want to include in or exclude from the comparison.
5 Type S to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

6 If you selected to display a list of the object attributes, specify the object attributes.

   A To include an object attribute in the comparison, type Y adjacent to the attribute.

   B If you specified to include the partition attributes in step 6A, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

   C Press END to return to the Compare CDL Options panel.

7 Type S to generate comparison report information as comments in the CDL file.

8 (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

9 Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

   The Compare JCL Processing Interface panel is displayed.

10 Specify the data set names to be used in the comparison.

   A (batch run type) Type the data set name for the JCL that is generated by this process.

   B Type the data set name for the CDL that is generated by this process.

   C Type one of the following options for Diagnostics:

      ■ (foreground run type) To display the diagnostics on the terminal, type TERM.

      ■ To write the diagnostics to a sequential file, type the name of the data set.

      ■ To write the diagnostics to a print data set, type SYSOUT.

   D (foreground run type) If you specified SYSOUT in step 10C, specify the Sysout Class.

11 Type S to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

   The Compare Override Options panel is displayed.
Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist

If you selected to override the defaults, specify the data set and CDL options.

A. To specify the data set allocation parameters for new data sets, type a value for the **Unitname**, **Volume**, **Pri qty**, or **Sec qty**.

B. To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C. To generate CDL that shows change propagation to dependent objects, type S to select **CDLDEP**.

D. To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for **Display change rules applied to Compare2 BEFORE the Compare**.

E. Press END to return to the Compare JCL Processing Interface panel.

Select your options for the comparison.

A. To create the input for the comparison, select one of the following options:

   - (foreground run type) Type S to select **Create Input**.
   - (batch run type) Type S to select **Create JCL**.

B. To review or modify input to the comparison process, select one of the following options:

   - (foreground run type) Type S to select **Edit Input**.
   - (batch run type) Type S to select **Edit JCL**.

   **Figure 71** shows an example of the ALUIN input stream.

   Press END to return to the Compare JCL Processing Interface panel.

**Figure 71**  **ALUIN input stream—DDL file to a DDL file**

<table>
<thead>
<tr>
<th>SSID DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1 DDL</td>
</tr>
<tr>
<td>CMPTYPE2 DDL</td>
</tr>
<tr>
<td>INCLUDE (ALTER CREATE DROP )</td>
</tr>
<tr>
<td>NODEFINE</td>
</tr>
</tbody>
</table>

C. To run the comparison, select one of the following options:

   - (foreground run type) Type S to select **Run Compare**.
   - (batch run type) Type S to select **Submit JCL**.

   CHANGE MANAGER generates a CDL file and a diagnostic output file.
Comparing a baseline to a baseline

You can compare a local catalog baseline to a remote catalog baseline. The CDL file that CHANGE MANAGER generates for this type of comparison can be used to migrate changes.

1. On the CHANGE MANAGER Main Menu, select **Compare (CDL Build)**, and press Enter.

   The Compare Type Selection panel is displayed.

2. Select **Baseline** for both the **Compare1 Type** and the **Compare2 Type**. Press Enter.

   The Compare Baseline to Baseline panel is displayed.

3. Specify the values for the inputs to the comparison.

   A. Type the name of the location of a catalog for Compare1. The local DB2 catalog is the default.

   B. Type the name of the baseline for Compare1 or type a wildcard pattern to display a list of baselines.

---

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

---

**Where to go from here**

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.

**TIP**

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

---

**Comparing a baseline to a baseline**

D. To edit the CDL file that is generated by the comparison, type **S** to select **Edit CDL File**.
Type the name of the location of a catalog for Compare2. The local DB2 catalog is the default.

Type the name of the baseline for Compare2 or type a wildcard pattern to display a list of baselines.

(optional) Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

Press Enter.

The Compare CDL Options panel is displayed.

Type Y or N for each object type that you want to include in or exclude from the comparison.

Type S to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

If you selected to display a list of the object attributes, specify the object attributes.

To include an object attribute in the comparison, type Y adjacent to the attribute.

If you specified to include the partition attributes in step 6A, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

Press END to return to the Compare CDL Options panel.

Type S to generate comparison report information as comments in the CDL file.

(optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

The Compare JCL Processing Interface panel is displayed.
10 Specify the data set names to be used in the comparison.

A *(batch run type)* Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.

C Type one of the following options for Diagnostics:
   - *(foreground run type)* To display the diagnostics on the terminal, type `TERM`.
   - To write the diagnostics to a sequential file, type the name of the data set.
   - To write the diagnostics to a print data set, type `SYSOUT`.

D *(foreground run type)* If you specified `SYSOUT` in step 10C, specify the Sysout Class.

11 Type S to select **Override Compare defaults** to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

The Compare Override Options panel is displayed.

12 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the `Unitname`, `Volume`, `Priqty`, or `Secqty`.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select `CDLDEP`.

D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for **Display change rules applied to Compare** BEFORE the Compare.

E Press END to return to the Compare JCL Processing Interface panel.

13 Select your options for the comparison.

A To create the input for the comparison, select one of the following options:
   - *(foreground run type)* Type S to select **Create Input**.
   - *(batch run type)* Type S to select **Create JCL**.
Comparing a baseline to a baseline

B To review or modify input to the comparison process, select one of the following options:

- *(foreground run type)* Type S to select Edit Input.
- *(batch run type)* Type S to select Edit JCL.

Figure 72 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

![Figure 72 ALUIN input stream—baseline to a baseline](image)

C To run the comparison, select one of the following options:

- *(foreground run type)* Type S to select Run Compare.
- *(batch run type)* Type S to select Submit JCL.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

D To edit the CDL file that is generated by the comparison, type S to select Edit CDL File.

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

E Press Enter.

Where to go from here

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.
Comparing a baseline to a DB2 catalog

You can compare a catalog baseline to a DB2 catalog. You can use this type of comparison to maintain versions of an application’s data structures. The baseline is a snapshot of the object definitions in the DB2 catalog at a particular point in time.

1. On the CHANGE MANAGER Main Menu, select Compare (CDL Build), and press Enter.

   The Compare Type Selection panel is displayed.

2. Select Baseline for the Compare1 Type and select Catalog for the Compare2 Type. Press Enter.

   The Compare Baseline to Catalog panel is displayed.

3. Specify the values for the inputs to the comparison.

   A. Type the name of the catalog baseline for Compare1 or type a wildcard pattern to display a list of baselines.

   B. Type the name of the location of a catalog for Compare2. The local DB2 catalog is the default.

   C. (optional) Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

   D. Press Enter.

   The Compare CDL Options panel is displayed.

4. Type Y or N for each object type that you want to include in or exclude from the comparison.

5. Type S to display a list of the object attributes to include in or exclude from the comparison.

   The Compare Object Attributes panel is displayed.

---

**TIP**

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
6 If you selected to display a list of the object attributes, specify the object attributes.

A To include an object attribute in the comparison, type Y adjacent to the attribute.

B If you specified to include the partition attributes in step 6A, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

C Press END to return to the Compare CDL Options panel.

7 Type S to generate comparison report information as comments in the CDL file.

8 (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

9 Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

The Compare JCL Processing Interface panel is displayed.

10 Specify the data set names to be used in the comparison.

A (batch run type) Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.

C Type one of the following options for Diagnostics:

- (foreground run type) To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D (foreground run type) If you specified SYSOUT in step 10C, specify the Sysout Class.

11 Type S to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

The Compare Override Options panel is displayed.
12 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the Unitname, Volume, Priqty, or Secqty.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select CDLDEP.

D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for Display change rules applied to Compare2 BEFORE the Compare.

E Press END to return to the Compare JCL Processing Interface panel.

13 Select your options for the comparison.

A To create the input for the comparison, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

B To review or modify input to the comparison process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

Figure 73 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

Figure 73 ALUIN input stream—baseline to a DB2 catalog

| SSID DEBA
| CMPTYPE1 BASELINE
| CMPIN1 RDACRJ.BLUNLD200704240001
| CMPTYPE2 LOCAL
| INCLUDE (ALTER CREATE DROP )
| NODEFINE

C To run the comparison, select one of the following options:

- (foreground run type) Type S to select Run Compare.
- (batch run type) Type S to select Submit JCL.
Comparing a DB2 catalog to a baseline

CHANGE MANAGER generates a CDL file and a diagnostic output file.

D To edit the CDL file that is generated by the comparison, type S to select **Edit CDL File**.

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

E Press Enter.

Where to go from here

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.

**TIP**

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Comparing a DB2 catalog to a baseline

You can compare a DB2 catalog to a catalog baseline. You can use this type of comparison to roll back changes that are made to an application.

1 On the CHANGE MANAGER Main Menu, select **Compare (CDL Build)**, and press Enter.

   The Compare Type Selection panel is displayed.

2 Select **Catalog** for the Compare1 Type and select **Baseline** for the Compare2 Type. Press Enter.

   The Compare Catalog to Baseline panel is displayed.

3 Specify the values for the inputs to the comparison.

   A Type the name of the location of a catalog for Compare1. The local DB2 catalog is the default.

   B Type the name of the baseline for Compare2.
C *(optional)* Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

D  Press Enter.

The Compare CDL Options panel is displayed.

4 Type Y or N for each object type that you want to include in or exclude from the comparison.

5 Type S to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

6 If you selected to display a list of the object attributes, specify the object attributes.

A  To include an object attribute in the comparison, type Y adjacent to the attribute.

B  If you specified to include the partition attributes in step 6A, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

C  Press END to return to the Compare CDL Options panel.

7 Type S to generate comparison report information as comments in the CDL file.

8 *(optional)* Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

9 Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

The Compare JCL Processing Interface panel is displayed.

10 Specify the data set names to be used in the comparison.

A *(batch run type)* Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.
Comparing a DB2 catalog to a baseline

C Type one of the following options for **Diagnostics**:
- *(foreground run type)* To display the diagnostics on the terminal, type `TERM`.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type `SYSOUT`.

D *(foreground run type)* If you specified `SYSOUT` in step 10C, specify the **Sysout Class**.

11 Type S to select **Override Compare defaults** to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

The Compare Override Options panel is displayed.

12 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the **Unitname**, **Volume**, **Priqty**, or **Secqty**.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select **CDLDEP**.

D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for **Display change rules applied to Compare2 BEFORE the Compare**.

E Press END to return to the Compare JCL Processing Interface panel.

13 Select your options for the comparison.

A To create the input for the comparison, select one of the following options:

- *(foreground run type)* Type S to select **Create Input**.
- *(batch run type)* Type S to select **Create JCL**.

B To review or modify input to the comparison process, select one of the following options:

- *(foreground run type)* Type S to select **Edit Input**.
- *(batch run type)* Type S to select **Edit JCL**.
Figure 74 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

**Figure 74  ALUIN input stream—DB2 catalog to a baseline**

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>LOCAL</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>BASELINE</td>
</tr>
<tr>
<td>CMPIN2</td>
<td>RDACRJ.BL200704260001</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>(ALTER CREATE DROP )</td>
</tr>
<tr>
<td>NODEFINE</td>
<td></td>
</tr>
</tbody>
</table>

**C** To run the comparison, select one of the following options:

- *(foreground run type)* Type S to select **Run Compare**.
- *(batch run type)* Type S to select **Submit JCL**.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

**D** To edit the CDL file that is generated by the comparison, type S to select **Edit CDL File**.

---

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

---

**E** Press Enter.

**Where to go from here**

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem.

---

**TIP**

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
Comparing a baseline to a DDL file

You can compare a catalog baseline to a DDL file. You can use this type of comparison to identify changes that need to be made to DB2 catalog objects that are represented by a baseline.

1 On the CHANGE MANAGER Main Menu, select **Compare (CDL Build)**, and press **Enter**.

   The Compare Type Selection panel is displayed.

2 Select **Baseline** for the **Compare1 Type** and select **DDL** for the **Compare2 Type**. Press **Enter**.

   The Compare Baseline to DDL panel is displayed.

3 Specify the values for the inputs to the comparison.

   A Type the name of the baseline for Compare1.

   B Type the data set name of the DDL file for Compare2.

   C *(optional)* Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

   D Press **Enter**.

   The Compare CDL Options panel is displayed.

4 Type Y or N for each object type that you want to include in or exclude from the comparison.

5 Type S to display a list of the object attributes to include in or exclude from the comparison.

   The Compare Object Attributes panel is displayed.

6 If you selected to display a list of the object attributes, specify the object attributes.

   A To include an object attribute in the comparison, type Y adjacent to the attribute.

   B If you specified to include the partition attributes in step 6A, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.
Press END to return to the Compare CDL Options panel.

7 Type S to generate comparison report information as comments in the CDL file.

8 *(optional)* Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules *after* it performs the comparison (when it generates the CDL).

9 Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

The Compare JCL Processing Interface panel is displayed.

10 Specify the data set names to be used in the comparison.

A *(batch run type)* Type the data set name for the JCL that is generated by this process.

B Type the data set name for the CDL that is generated by this process.

C Type one of the following options for Diagnostics:

- *(foreground run type)* To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D *(foreground run type)* If you specified SYSOUT in step 10C, specify the Sysout Class.

11 Type S to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

The Compare Override Options panel is displayed.

12 If you selected to override the defaults, specify the data set and CDL options.

A To specify the data set allocation parameters for new data sets, type a value for the Unitname, Volume, Priqty, or Secqty.

B To include CDL statements for created, altered, or dropped objects, type S to select the statements.

C To generate CDL that shows change propagation to dependent objects, type S to select CDLDEP.
To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for Display change rules applied to Compare2 BEFORE the Compare.

Press Enter to return to the Compare JCL Processing Interface panel.

Select your options for the comparison.

To create the input for the comparison, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

To review or modify input to the comparison process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

Figure 75 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

To run the comparison, select one of the following options:

- (foreground run type) Type S to select Run Compare.
- (batch run type) Type S to select Submit JCL.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

To edit the CDL file that is generated by the comparison, type S to select Edit CDL File.

__WARNING__

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.
Generating a CDL file from a work ID

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.

   The WORKID Action Menu panel is displayed.

2. Type the name of a **WORKID** and select **Convert Alter WORKID to CDL**. Then, press **Enter**.

   The Compare CDL Options panel is displayed.

3. Type **Y** or **N** for each object type that you want to include in or exclude from the comparison.

4. Type **S** to display a list of the object attributes to include in or exclude from the comparison.

   The Compare Object Attributes panel is displayed.

5. If you selected to display a list of the object attributes, specify the object attributes.

   **A** To include an object attribute in the comparison, type **Y** adjacent to the attribute.

   **B** If you specified to include the partition attributes in step 5A, you can exclude one or more of the partition attributes. Type **N** adjacent to the attributes that you want to exclude.

---

**TIP**

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
C  Press END to return to the Compare CDL Options panel.

6  Type **S** to generate comparison report information as comments in the CDL file.

7  *(optional)* Specify the name of an outbound migrate profile. The outbound migrate profile must contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

8  Select a Run Type and press Enter. Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

   The Compare JCL Processing Interface panel is displayed.

9  Specify the data set names to be used in the comparison.

   A  *(batch run type)* Type the data set name for the JCL that is generated by this process.

   B  Type the data set name for the CDL that is generated by this process.

   C  Type one of the following options for Diagnostics:

      ■  *(foreground run type)* To display the diagnostics on the terminal, type **TERM**.

      ■  To write the diagnostics to a sequential file, type the name of the data set.

      ■  To write the diagnostics to a print data set, type **SYSOUT**.

   D  *(foreground run type)* If you specified **SYSOUT** in step 9C, specify the **Sysout Class**.

10 Type **S** to select Override Compare defaults to change data set allocation parameters or to control the content of the CDL that is generated by the comparison process. Press Enter.

   The Compare Override Options panel is displayed.

11 If you selected to override the defaults, specify the data set and CDL options.

   A  To specify the data set allocation parameters for new data sets, type a value for the **Unitname, Volume, Priqty, or Secqty**.

   B  To include CDL statements for created, altered, or dropped objects, type **S** to select the statements.

   C  To generate CDL that shows change propagation to dependent objects, type **S** to select **CDLDEP**.
D To display, in the CDL, the change rules from an outbound migrate profile or the automatic change rules, type S for **Display change rules applied to Compare2 BEFORE the Compare**.

E Press END to return to the Compare JCL Processing Interface panel.

12 Select your options for the comparison.

A To create the input for the comparison, select one of the following options:

- *(foreground run type)* Type S to select **Create Input**.
- *(batch run type)* Type S to select **Create JCL**.

B To review or modify input to the comparison process, select one of the following options:

- *(foreground run type)* Type S to select **Edit Input**.
- *(batch run type)* Type S to select **Edit JCL**.

*Figure 76* shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

---

**Figure 76**  ALUIN input stream—convert work ID to CDL

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>WORKID</td>
</tr>
<tr>
<td>CMPIN1</td>
<td>RDACRJ.ALT01</td>
</tr>
<tr>
<td>INCLUDE (ALTER CREATE DROP )</td>
<td></td>
</tr>
<tr>
<td>NODEFINE</td>
<td></td>
</tr>
</tbody>
</table>

C To run the comparison, select one of the following options:

- *(foreground run type)* Type S to select **Run Compare**.
- *(batch run type)* Type S to select **Submit JCL**.

CHANGE MANAGER generates a CDL file and a diagnostic output file.

D To edit the CDL file that is generated by the comparison, type S to select **Edit CDL File**.

---

**WARNING**

Do not attempt to edit the CDL file while the comparison is in progress, or the file will be locked, and the Compare component will not be able to access it.

E Press Enter.
Where to go from here

After you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID on the same or on a different DB2 subsystem and process it as a set of change requests for a subsystem.

**TIP**
To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

---

Using a script to perform a comparison

The CMP_STRUCTURE_CHG script in the CM/PILOT component of CHANGE MANAGER contains the steps to perform a comparison and generate a CDL file if differences exist between the data structures. If the CHANGE MANAGER Compare component finds no differences, CM/PILOT worklist processing stops and an Analysis worklist is not created. If differences exist, a CDL file is created and imported, and an Analysis worklist is created to apply the changes. This example assumes that the comparison, import, analysis, and execution are performed on the same subsystem. For more information about scripts, see Chapter A, “Using scripting tools to automate change management.”

In this task, you will create and execute a task ID.

**Before you begin**

Depending on the inputs specified for the comparison, any of the following items can be required:

<table>
<thead>
<tr>
<th>Item</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL file</td>
<td>“Using a DDL file or a migrate-type worklist in the comparison” on page 242</td>
</tr>
<tr>
<td>CHANGE MANAGER worklist</td>
<td>“Using a DDL file to create a DDL baseline” on page 181</td>
</tr>
<tr>
<td>DDL baseline</td>
<td>“Using a DB2 catalog to create a catalog baseline” on page 180</td>
</tr>
<tr>
<td>catalog baseline</td>
<td>“Creating a catalog baseline profile” on page 198</td>
</tr>
<tr>
<td>catalog baseline profile containing scope rules</td>
<td>“Creating an outbound migrate profile” on page 146</td>
</tr>
</tbody>
</table>

---

To import a file, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
Using a script to perform a comparison

### To create a task ID

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   
   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press Enter.
   
   The TASKID Action Menu is displayed.

3. Type the name of a new TASKID and select Create a TASKID. Then, press Enter.
   
   The Script Selection List is displayed.

4. Select Change data structures using a Compare trigger, and press Enter.
   
   The Create TASKID panel is displayed.

5. Specify the information for a task ID.

   A. CM/PILOT uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for WORKID or type the name of a work ID name template.

   B. (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C. (optional) Specify a Comment to describe the task ID.

   D. Press END.

   The task ID is created and the TASKID Action Menu is displayed.

### To execute the task ID

1. Select Execute a TASKID and press Enter.
   
   The TASKID Interface panel is displayed.
2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT **Worklist**.

B Type the data set name for the CM/PILOT **JCL**.

C Type the data set name for **Diagnostics** or type **SYSOUT**.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in **CHANGE MANAGER**.

A Type **S** to select **Override CHANGE MANAGER options** and then press **Enter**.

The Override CHANGE MANAGER Options panel is displayed.

B Select **Override CHANGE MANAGER Options**.

C Type **S** to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press **Enter**.

For more information about the CHANGE MANAGER options, see the **ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1**.

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type **S** to select **Create Worklist**. Then, press **Enter**.

The Step Settings panel is displayed.

B Type **S** to select **Step 1 Compare**, and press **Enter**.

The TASKID Compare Step Options panel is displayed.
C Select the structures that you want to compare, and press Enter. (For this example, Catalog was selected for Compare1 Type and Compare2 Type.)

The Step Settings panel is displayed.

D (optional) To modify the Analysis step, type S to select Step 2 Analysis step and press Enter.

E (optional) To modify the Execution step, type S to select Step 3 Execution step and press Enter.

To specify the inputs for the comparison

6 Press Enter.

The Compare Catalog to Catalog Scope Selection panel is displayed.

7 Select One catalog object and all of its dependents as the source of the scope. Press Enter.

--- NOTE ---
To select multiple catalog objects, specify scope rules in a baseline profile or an outbound migrate profile and select to use the profile as the source of the scope. For information, see “Using a profile to scope a catalog to a catalog comparison” on page 272.

The Compare Catalog to Catalog panel is displayed.

8 Type the abbreviation for an object type.

9 If you plan to use a wildcard for Name - part 1, Name - part 2, or Name - part 3, type Y to display a list of objects in a mixed list.

10 Specify the values for Compare1.

A Type the name of the object.

- If the name of the object has only one part, for Name - part 1, type the name or owner, or type a wildcard to specify an object from a list of objects.

- If the name of the object has two parts, specify the following information:

  — For Name - part 1, type the name or owner, or type a wildcard to specify an object from a list of objects.

  — For Name - part 2, type the name of a single object or type a wildcard to specify an object from a list of objects.
If the name of the object has three parts (such as a stored procedure), specify the following information:

- For Name - part 1, type the name or owner, or type a wildcard to specify an object from a list of objects.

- For Name - part 2, type the name of a single object or type a wildcard to specify an object from a list of objects.

- For Name - part 3, type the name of an object or type a wildcard to specify an object from a list of objects.

B (optional) To limit the scope of the comparison, type the name of the table owner.

C (optional) Specify the name of the location of the remote DB2 subsystem. As an alternative, you can type a wildcard for the location name to generate a list of remote locations from which you can select.

11 Specify the values for Compare2.

A Repeat step 10.

B (optional) Specify the name of an outbound migrate profile for Compare2. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects in Compare2 before the comparison.

12 Press Enter.

If you chose to display a list of objects in a mixed list in step 9, the Mixed List panel is displayed. Otherwise, the Compare CDL Options panel is displayed (proceed to step 13).

13 From the Mixed List panel, you can perform one of the following actions:

- Type S to select a single object to use in the comparison. Then, press Enter.
- Press END to exit the mixed list. All of the objects listed on the mixed list will be included in the comparison.

14 Press Enter.

The Taskid CDL Options panel is displayed.

15 On the Taskid Compare CDL Options panel, specify the object types and attributes for the comparison.
A Type Y or N for each object type that you want to include in or exclude from the comparison.

B Type S to display a list of the object attributes to include in or exclude from the comparison.

C Press Enter.

If you chose to display a list of the object attributes in step 15B, the Compare Object Attributes panel is displayed. Otherwise, the TASKID Migrate Profiles panel is displayed (proceed to step 18).

16 On the Compare Object Attributes panel, specify the object attributes.

A To include an object attribute in the comparison, type Y adjacent to the attribute.

B If you specified to include the partition attributes, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

C Press Enter to return to the Taskid Compare CDL Options panel.

17 Press Enter.

The TASKID Migrate Profiles panel is displayed.

18 On the TASKID Migrate Profiles panel, specify the migrate profiles.

A (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

B (optional) Specify the name of an inbound migrate profile. The inbound migrate profile can contain change rules. It can be used when the CDL that is generated from the comparison is imported.

C Press Enter.

The CHANGE MANAGER Datasets panel is displayed.

19 Specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press Enter.

The TASKID Interface panel is displayed.
20 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

21 To run the comparison and to import the resulting CDL into an alter-type work ID, execute the Analysis worklist.

---

**NOTE**

The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.

---

**TIP**

To execute a worklist, see page 562.

---

**Where to go from here**

Now that you have compared your data structures and have generated a CDL file, you can import the CDL file to an alter-type work ID and process it as a set of change requests for a subsystem. You can also import the CDL file to an alter-type work ID on a different subsystem to update a separate version of the data structures. To import the CDL file and update your data structures, see Chapter 4, “Altering data structures in a database environment.”
Chapter 4  Altering data structures in a database environment

This chapter presents the following topics:

Before you begin .......................................................... 310
Overview ................................................................. 310
Altering data structures .................................................. 314
Specifying changes to data structures .................................. 314
  Identifying changes to data structures ............................ 316
  Specifying changes to auxiliary objects for a LOB column .... 317
  Converting table spaces ............................................ 320
  Considerations for changing data structures .................. 326
Importing files ................................................................ 328
  Deciding what type of file to import ............................. 329
  Deciding the type of work ID to use ............................. 332
  Validity checks ....................................................... 334
  Reviewing the results of the import ............................. 335
Estimating space .......................................................... 336
  Estimating space for a table space .............................. 337
  Estimating space for an index .................................... 341
Estimating space requirements based on user-specified values .... 344
Analyzing changes ....................................................... 345
  Overriding the default processing options ................. 346
  Validity and performance checks ............................. 347
  Reviewing the results of Analysis ............................ 347
Generating JCL ........................................................... 348
Executing changes ....................................................... 349
  Specifying options for Execution .............................. 349
  Controlling authorizations ..................................... 352
  Processing a worklist .............................................. 352
  Reviewing the results of your changes ..................... 353
Accomplishing your goals ............................................... 354
  Creating an alter-type work ID ................................. 357
  Creating an alter-type work ID in batch mode ............ 358
  Sorting work IDs .................................................. 359
Deleting a work ID .......................................................... 360
Creating an inbound migrate profile in CHANGE MANAGER .......... 361
Replicating multiple work IDs .......................................... 362
Modifying a work ID with values in a user-defined table ............... 366
Changing column definitions for imported DDL .......................... 370
Deleting multiple work IDs ............................................. 373
Deleting sync table entries for multiple work IDs .................... 376
Importing a CDL, DDL, or DML file in CHANGE MANAGER ............ 379
Importing a DDL file in ALTER ......................................... 381
Specifying changes to data structures ................................... 383
Creating a primary constraint for a table .............................. 384
Creating a unique constraint for a table ............................... 386
Editing the columns for a unique constraint ............................ 390
Editing a primary key ................................................... 392
Naming a primary key .................................................... 395
Changing the attributes of an index .................................... 397
Creating indexes for ROWID GENERATED ALWAYS columns by importing DML ........................................ 400
Adding a column to a table .............................................. 402
Dropping a column from a table ....................................... 404
Changing the data type and length of a column ....................... 405
Moving multiple tables using Quick Edit ................................ 406
Changing data structures by using an existing work ID ................ 408
Receiving DDL to create data structures ............................... 411
Changing data structures by using DML ................................ 415
Adding columns to the DB2 PLAN_TABLE table by importing DML 420
Deleting aliases by importing DML ..................................... 422
Copying a column by importing DML ................................... 423
Updating index partitions for a database by importing DML ........... 424
Adding WITH RESTRICT ON DROP to tables by importing DML .... 426
Updating table spaces or indexes with existing data sets by importing DML ........................................ 427
Creating synonyms for selected tables by importing DML .......... 428
Converting VCAT-defined partitions to STOGROUP-defined partitions by importing DML ........................................ 429
Creating a materialized query table (MQT) ............................ 431
Creating an application-period temporal table ......................... 433
Creating a system-period temporal table .............................. 435
Creating a history table for a system-period temporal table .......... 437
Creating a history table from an existing history table ............... 440
Creating a history table from a base table ............................ 443
Creating an external stored procedure ................................. 446
Creating a native SQL stored procedure ................................ 447
Creating a new version of a native SQL stored procedure ........... 449
Updating options for a native SQL stored procedure by importing DML ..... 451
Updating parameters for a native SQL stored procedure by importing DML ........................................ 452
Estimating the space requirements for a table space ................. 453
Estimating the space requirements for an index ........................ 460
Estimating space requirements for a table space based on user-specified values ........................................ 464
Estimating space requirements for an index based on user-specified values  
Setting space estimation parameters by importing DML  
Estimating the primary and secondary quantities in table space and index partitions by importing DML  
Setting a secondary quantity in table space and index partitions by importing DML  
Creating objects for the SHRLEVEL CHANGE HSSC process  
Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC process  
Converting table spaces to partition-by-growth table spaces by importing DML  
Converting a nonpartitioned table space to an index-controlled partitioned table space  
Converting a nonpartitioned table space to a table-controlled partitioned table space  
Converting a nonpartitioned table space to a range-partitioned table space  
Converting a nonpartitioned table space to a partition-by-growth table space  
Converting a partitioned table space to a nonpartitioned table space  
Changing a partitioned table space to a range-partitioned table space by importing DML  
Converting a table-controlled partitioned table space to an index-controlled partitioned table space  
Converting table-controlled partitioned table spaces to range-partitioned table spaces by importing DML  
Converting a range-partitioned table space to an index-controlled partitioned table space  
Converting an index-controlled partitioned table space to a table-controlled partitioned table space  
Converting index-controlled partitioned table spaces to table-controlled partitioned table spaces by importing DML  
Converting an index-controlled partitioned table space to a range-partitioned table space  
Converting index-controlled partitioned table spaces to range-partitioned table spaces by importing DML  
Converting a partitioned table space to a partition-by-growth table space  
Converting a partition-by-growth table space to an index-controlled partitioned table space  
Converting a partition-by-growth table space to a table-controlled partitioned table space  
Converting a partition-by-growth table space to a range-partitioned table space  
Changing a single-table table space to a partition-by-growth table space by importing DML  
Changing tables in an explicit database to an implicit database by importing DML  
Converting explicit databases and implicit table spaces to implicit databases and table spaces by importing DML  
Changing the encoding scheme for a table space by importing DML  
Adding a ROWID and a LOB column to create a base table
Before you begin

Before you create or modify data structures, you need to ensure that the default and user options for the ALTER or CHANGE MANAGER products have been set up properly. To set or change the options, see the _ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1_.

You might want to create a baseline of your environment before you alter your data structures. A baseline is a snapshot of an application’s data structures at a given point in time. For information about creating baselines, see Chapter 2, “Taking a snapshot of a database environment.”

You also might want to compare your data structures so that you can generate Change Definition Language (CDL) commands in a file that show the differences between two sets of data structures. For information about performing a comparison, see Chapter 3, “Comparing database environments.”

Overview

ALTER and CHANGE MANAGER enable you to create, modify, or drop data structures within a DB2 subsystem. For example, you can change the name of a data structure and modify attributes such as limit keys and storage groups. This process of creating, modifying, or dropping data structures is known as altering the data structures.
With ALTER and CHANGE MANAGER, you can automate the alter process, as well as the following tasks:

- determine the dependencies on changed data structures and preserve those dependencies and their associated data
- analyze the impact of changes by validating the changes against the DB2 catalog
- import data definition language (DDL) files and CDL files from another subsystem and apply the changes to your subsystem

This chapter describes the alter process, and the tasks that you can perform to create, modify, and drop your data structures.

The workflow in Figure 77 illustrates the alter process.
Figure 77  Workflow for the alter process (part 1 of 2)

Start

Do you want to create a structure only baseline? yes

Go to baseline process

no

Create an alter-type Work ID

Do you want to import CDL or DDL? yes

Do you want to use change rules? yes

Define the change rules in an inbound migrate profile

no

no

import CDL or DDL to the alter-type Work ID

Specify the changes to the data structures no

no

no
Figure 77  Workflow for the alter process (part 2 of 2)
Altering data structures

To alter the data structures, you must first create an alter-type work ID. The alter-type work ID is the basic unit of work in the alter process. After you create the alter-type work ID, you can either specify changes to the data structures (see page 314) or import external files (see page 328). If you import changes to the data structures from DDL files or CDL files, you can also use an inbound migrate profile to customize those changes.

**TIP**
To create an alter-type work ID, see page 357.

**NOTE**
For more information, view the Quick Course Creating Work IDs. You must have a BMC Support ID to view the Quick Course.

Specifying changes to data structures

You specify the changes to data structures through the Specification component. You can select objects from the DB2 catalog and describe changes to be performed on those objects. You can modify and delete objects by using the ALTER and DROP commands. You can also create new objects by using the LIKE or CREATE command. Specification stores the changes in the CD tables using the work ID name as the identifier.

**NOTE**
For more information, view the Quick Course Performing Specification. You must have a BMC Support ID to view the Quick Course.

You specify changes to the data structures in the same way that you specify the data structures for a migration. You can specify the names of the data structures that you want to appear in a mixed list on the Object Specification panel.

From the mixed list, you can specify the objects that you want to change. The Mixed List panel (see Figure 78) displays lists of different types of DB2 data structures.
Specifying changes to data structures

Chapter 4 Altering data structures in a database environment

Figure 78 Mixed List panel

For a list of the abbreviations for object types that Specification uses in the Mixed List panel, as well as the data types for the object names, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

You can use action codes (or Command line commands) on the Mixed List panel to indicate the type of action that you want to perform. Some of the actions that you can perform in the Mixed List panel are described in Table 26.

Table 26 Mixed List actions (part 1 of 2)

<table>
<thead>
<tr>
<th>Act</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX</td>
<td>Auxiliary Object (LOB)</td>
<td>displays the auxiliary objects for LOB columns that are associated with a base table</td>
</tr>
<tr>
<td>D</td>
<td>Drop</td>
<td>excludes the data structure that is explicitly marked for migration (or implicitly marked in the migrate options) from being included in the migration or marks the data structure to be dropped</td>
</tr>
<tr>
<td>E</td>
<td>Edit</td>
<td>specifies the creation of data structures on the destination subsystem that are a modified version of those on the origin subsystem</td>
</tr>
<tr>
<td>L</td>
<td>Like</td>
<td>specifies the creation of a single data structure on the destination subsystem using the data structure from the origin subsystem as a template</td>
</tr>
</tbody>
</table>

Dependent structures are not created. You can also use the C (Create) action code to create a data structure without a template.
Identifying changes to data structures

When you specify a change to an object, an asterisk (*) and a change-level indicator appears on a list panel adjacent to the Act column. Table 27 describes each of the indicators that relate to changing data structures.

### Table 27 Change-level indicators (part 1 of 2)

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>indicates that you changed one or more object attributes</td>
</tr>
<tr>
<td>D</td>
<td>indicates that you dropped an object</td>
</tr>
<tr>
<td>L</td>
<td>indicates that you LIKEd, copied, inserted, or added an object</td>
</tr>
</tbody>
</table>

You can also use the following command-line commands on the Mixed List panel:

- **DROPALL** marks all of the data structures to be dropped. This command is the equivalent of typing **D** in the Act column for all data structures.

- **UNDOALL** reverses the action on all data structures. This command is the equivalent of typing **U** in the Act column for all data structures.

For more information about using a mixed list, see “Specifying the data structures for a migration” on page 41. For more information about the commands that you can use in the Mixed List panel, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**Tip**

To specify changes to data structures, see page 383.
Specifying changes to auxiliary objects for a LOB column

When you request a change to a data structure, only the definition of the data structure and the requested changes are saved in the Change Definition (CD) tables. The specified changes do not occur in the DB2 catalog until you analyze the changes, generate a worklist, and execute the worklist.

### Specifying changes to auxiliary objects for a LOB column

To edit, like, or drop auxiliary objects for a LOB column, you use the Auxiliary Objects List panel (see Figure 79). The panel lists only auxiliary table spaces, tables, and indexes.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>indicates that you changed at least one auxiliary object that is associated with a base table. If you change the base table, the appropriate indicator overrides the @ indicator. For more information about auxiliary objects, see “Specifying changes to auxiliary objects for a LOB column” on page 317.</td>
</tr>
<tr>
<td>$</td>
<td>indicates that you estimated the space for an object. <strong>Note:</strong> This indicator is not applicable to auxiliary objects.</td>
</tr>
</tbody>
</table>
Specifying changes to auxiliary objects for a LOB column

Figure 79  Auxiliary Objects List panel

The auxiliary objects are grouped by each partition and are sorted by the LOB column name and partition number. Orphaned auxiliary objects are displayed in the Unattached Objects section at the end of the list. The Auxiliary Objects List panel uses the same object-type abbreviations as the Mixed List panel.

As in the Mixed List panel, you can use action codes (or Command line commands) in the Auxiliary Objects List panel to indicate the type of action that you want to perform. You can only edit, like, drop, or undo an action on an auxiliary object for a LOB column in the Auxiliary Objects List panel.

For more information about the commands that you can use in the Auxiliary Objects List panel, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Creating auxiliary objects for a LOB column

To create an auxiliary object for a base table in a base table space, you can use the CAX action code (or Command line command) on the Table Column Detail or the Table Columns List panel. The CAX action code can be used with partitioned or nonpartitioned table spaces.

For partitioned table spaces, you use the CAX action code to use templates to create the auxiliary table spaces, auxiliary tables, and auxiliary indexes that are required for a base table that contains a LOB column. The templates enable you to specify the definition for an auxiliary object and to replicate that definition to the other auxiliary objects in each partition of the base table.
You can use the templates to name the auxiliary objects in a partitioned base table in one of the following ways:

- with name prefixing, which causes the products to automatically and uniquely name the auxiliary objects, based on a prefix that you provide
- without name prefixing, which causes the products to name all of the auxiliary objects based on a name that you provide

After you provide the name, you must edit the new auxiliary objects and make the names unique.

When you exit the last panel that is displayed for the templates, an auxiliary objects list panel displays the list of replicated objects. You can also name a single set of auxiliary objects (table space, table, and index) for a specific partition.

For nonpartitioned table spaces, the CAX command enables you to create all of the auxiliary objects that are required for a base table. Templates are not used. You must provide a name for each of the objects.

To create auxiliary objects, see page 525 and page 538.

Considerations for changing auxiliary objects for a LOB column

Consider the following items when you specify changes to auxiliary objects:

- The products do not allow you to drop an auxiliary table space when it contains an auxiliary table. If you drop the base table, the products also drop the auxiliary table and auxiliary index. After the auxiliary table and the auxiliary index are dropped, you can drop the auxiliary table space.

- You can drop an auxiliary table if it is empty. If it is not empty, you must drop the base table. Alternatively, you can delete all of the rows in a base table after it is unloaded and then drop the auxiliary table.

- The products do not allow you to drop an auxiliary index when it contains data. To drop the index, you must drop the auxiliary table. Alternatively, you can delete all of the rows in a base table after it is unloaded and then drop the auxiliary index.

- If you created the auxiliary objects with CURRENT RULES = STD, the products drop the auxiliary table space, auxiliary table, and auxiliary index when they drop the base table.
If you create a LOB column and the associated auxiliary objects in a base table and then cancel the creation of the LOB column (by typing CANCEL on the Table Columns List), the products do not create the LOB column. However, the products have already saved the definition of the auxiliary objects. In the Auxiliary Objects List panel, the column that is associated with the auxiliary objects is listed as &lt;NOT FOUND&gt;. If you do not remove the auxiliary objects, you will receive an error when you execute the worklist, because no association exists between the auxiliary objects and the base table. To remove the auxiliary objects, undo the actions on the Auxiliary Objects List panel.

If a base table is marked with a *@ change-level indicator in the Mixed List panel, you cannot undo the requested changes for the table. To remove the indicator, you must undo the changes to the auxiliary objects on the Auxiliary Objects List panel.

If you change the definition of an auxiliary index from STOGROUP to VCAT, you might need to rename the index. The first eight characters of the name of a VCAT-defined index must be unique. Similarly, if you create an auxiliary index and you are using name prefixing, you must specify a five-character prefix for the index name to ensure the uniqueness of the name.

You must create an auxiliary table space, table, and index for each LOB column for each partition of a partitioned table space.

**Converting table spaces**

DB2 provides several types of table spaces in which you can store various types of data. As your needs change, you might need to change the type of table space that holds your data. ALTER and CHANGE MANAGER enable you to easily convert from one type of table space to another. Table 28 shows the supported types of conversions and page references either to instructions for performing the conversions or to more information about the conversion.
### Table 28  Supported types of conversions

<table>
<thead>
<tr>
<th>From this type</th>
<th>nonpartitioned simple (DB2 Version 8) implicit simple table space</th>
<th>nonpartitioned segmented</th>
<th>index-controlled partitioned</th>
<th>table-controlled partitioneda</th>
<th>range-partitioned</th>
<th>partition-by-growth</th>
<th>implicit database and table spaceb</th>
<th>explicit database and implicit table spaceb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Supported</td>
<td>page 480</td>
<td>page 486</td>
<td>page 490</td>
<td>page 491</td>
<td>Supported</td>
<td>page 491</td>
<td></td>
</tr>
<tr>
<td>nonpartitioned segmented</td>
<td>page 480</td>
<td>page 486</td>
<td>page 490</td>
<td>page 491</td>
<td>Supported</td>
<td>page 491</td>
<td></td>
<td></td>
</tr>
<tr>
<td>index-controlled partitioned</td>
<td>page 493</td>
<td>page 500</td>
<td>page 505</td>
<td>page 507</td>
<td>Supported</td>
<td>page 491</td>
<td></td>
<td></td>
</tr>
<tr>
<td>table-controlled partitioned</td>
<td>page 493</td>
<td>page 497</td>
<td>page 498</td>
<td>page 507</td>
<td>Not supported</td>
<td>Not supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>range-partitioned</td>
<td>page 493</td>
<td>page 500</td>
<td>Supported</td>
<td>page 507</td>
<td>Not supported</td>
<td>Not supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>partition-by-growth</td>
<td>page 493</td>
<td>page 510</td>
<td>page 512</td>
<td>page 513</td>
<td>Not supported</td>
<td>page 325</td>
<td>page 325</td>
<td></td>
</tr>
<tr>
<td>(DB2 Version 9) implicit database and table space</td>
<td>page 493</td>
<td>page 510</td>
<td>page 512</td>
<td>page 513</td>
<td>page 325</td>
<td>page 325</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(DB2 Version 9) explicit database and implicit table space</td>
<td>page 493</td>
<td>page 510</td>
<td>page 512</td>
<td>page 513</td>
<td>page 325</td>
<td>page 325</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

a  For more information about converting a table space to a table-controlled partitioned table space or a range-partitioned table space, see page 322.

b  For more information about implicit databases and table spaces, see “Working with implicit objects in partition-by-growth table spaces” on page 325.
Converting a table space to a table-controlled partitioned table space or a range-partitioned table space

Analysis performs different actions depending on the type of table space that you choose to convert to a table-controlled partitioned table space or a range-partitioned table space. Table 29 describes these actions and lists the commands that Analysis generates in the worklist.

Table 29  Converting to table-controlled partitioning

<table>
<thead>
<tr>
<th>If you convert this type of table space...</th>
<th>Specification performs the following action...</th>
<th>and Analysis generates these commands in the worklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>a partitioned table space with a partitioning index</td>
<td>uses the DB2 catalog to populate the limit key and the partitioning key columns</td>
<td>ALTER TABLE ALTER PARTITION or ALTER INDEX NOT CLUSTER</td>
</tr>
<tr>
<td>a nonpartitioned table space, or a partitioned table space that does not have a partitioning index</td>
<td>does not populate the limit key and the partitioning key columns</td>
<td>DROP TABLESPACE CREATE TABLESPACE CREATE TABLE (using table-controlled partitioning)</td>
</tr>
<tr>
<td>a partitioned table space that you have LIKEd</td>
<td>might use an index that was created in the work ID to populate the limit key and the partitioning key columns</td>
<td>CREATE TABLESPACE CREATE TABLE (using table-controlled partitioning) CREATE INDEX CLUSTER</td>
</tr>
</tbody>
</table>

**NOTE**

If you are migrating a table that you are converting to table-controlled partitioning, the products include CREATE TABLESPACE and CREATE TABLE statements in the migrate-type worklist, even if the table space was not selected for migration.
Preserving the values of limit keys

**NOTE**

Preserving the values of limit keys is relevant only for table spaces that are not defined as LARGE and in which the DSSIZE of the table space is equal to zero.

IBM recommends that you convert a table in an index-controlled partitioned table space to table-controlled partitioning by altering the definition of the partitioning index from CLUSTER to NOT CLUSTER, and then back to CLUSTER. When you perform these actions, DB2 automatically sets the limit key values for the last partition to the following values:

- *(DB2 Version 9 or DB2 Version 8 with IBM APAR PK33529 applied)* MAXVALUE or MINVALUE
- *(DB2 Version 8)* high or low values (for example, X'FF' or X'00')

If a future change to the table requires that the table be dropped and rebuilt, DB2 might not be able to rebuild the table because the high values might be invalid values for data types for the limit key.

When you choose to convert an index-controlled partitioned table space to a table-controlled partitioned or range-partitioned table space, ALTER and CHANGE MANAGER use one of the following methods to accomplish the task:

- alter the definition of a partitioning index from CLUSTER to NOT CLUSTER and then back to CLUSTER
  
  This method is used if the table spaces are defined as LARGE and in which the DSSIZE of the table space is greater than zero (0), or if you choose not to preserve the values of the limit keys in the last partition of the table space.

- alter the value of the limit key for the last partition in the table space to the value that existed for the last partition

  This method, in which you preserve the values of the limit keys in the last partition of the table space, prevents the limit key from being set to MAXVALUE (or high values). However, this method requires the reorganization of the last partition.

  — If you preserve the values, ALTER and CHANGE MANAGER require that a REORG be run against the last partition. (If a REORG is not run, DB2 leaves the table space partition in a REORG PENDING status.) If the data stored in the last partition is beyond the limit key value, DB2 discards rows when it reorganizes the partition. To verify whether the data is beyond the limit key value, issue a SELECT MAX statement on the first column of the partitioning key.
If you do not preserve the values, DB2 automatically assigns the limit key values for the last partition to MAXVALUE or MINVALUE (or high or low values). All data that is currently stored in the last partition remains in the partition, even if the data was actually beyond the limit key value that was originally specified for the partition. In addition, ALTER and CHANGE MANAGER do not allow you to rotate or add partitions. Allowing DB2 to set high values for your limit key has the following disadvantages:

- The original limit key value specified for that partition will be lost.
- If you add or rotate a partition to this table, you will have to change the limit key for the existing last partition. When you change a limit key, DB2 puts the table space partition in a REORG PENDING status.

Before you run Analysis, you can change whether to preserve the values of the limit keys on the Tablespace Parts List panel.

**NOTE**

- Because of restrictions with the DB2 commands to alter limit keys of tables that contain large object (LOB) columns, ALTER and CHANGE MANAGER are unable to preserve the values of the limit keys if a table contains a LOB column, and if the table space is not defined as LARGE and the DSSIZE is equal to zero.
- If a table contains a LOB column, BMC recommends that you specify a value for the DSSIZE, and drop and create the table space before you convert it to table-controlled partitioning.

### Rotating partitions

You can identify table space partitions by both their physical and logical partition numbers. It is important to know the partition numbers because DB2 modifies the logical partition numbers when it rotates partitions of a table space. Table 30 shows how the components of ALTER and CHANGE MANAGER reference the partition numbers.

<table>
<thead>
<tr>
<th>Component/Object</th>
<th>Reference to partition numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>uses the logical partition number</td>
</tr>
<tr>
<td>Specification</td>
<td>Specification displays the physical partition number for information only.</td>
</tr>
<tr>
<td>Baseline</td>
<td>saves the physical and logical partition numbers</td>
</tr>
<tr>
<td>Baseline Report</td>
<td>uses the logical partition number</td>
</tr>
<tr>
<td></td>
<td>The baseline report includes the physical partition number as a comment.</td>
</tr>
<tr>
<td>Worklist</td>
<td>uses the physical partition number</td>
</tr>
</tbody>
</table>
ALTER and CHANGE MANAGER enable you to rotate the first partitions in a table-controlled partitioned or a range-partitioned table space to the last partitions in the table space. You specify the number of partitions to rotate in the Tablespace Parts List panel. For each partition, Analysis builds an ALTER TABLE ROTATE PARTITION statement in the worklist. When you execute the worklist, DB2 deletes all of the data in the rotated partitions of the table. The products allow you to change the value of the limit keys so that they are valid for the last partitions.

**NOTE**

If a table in the table space has a foreign key with a delete rule of RESTRICT, ALTER and CHANGE MANAGER remove the relationship before rotating the partitions and then add the relationship after rotating the partitions.

### Working with implicit objects in partition-by-growth table spaces

ALTER and CHANGE MANAGER support the automatic creation of a database or a table space. The products also support the use of the `<DEFLT>` keyword as the database name or table space name, as shown in Table 31.

#### Table 30 Referencing physical and logical partition numbers (part 2 of 2)

<table>
<thead>
<tr>
<th>Component/Object</th>
<th>Reference to partition numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL</td>
<td>uses the physical partition number</td>
</tr>
<tr>
<td>CDL</td>
<td>uses the logical partition number</td>
</tr>
</tbody>
</table>

#### Table 31 Specifying `<DEFLT>` for the database or table space name

<table>
<thead>
<tr>
<th>Database name</th>
<th>Table space name</th>
<th>CREATE syntax</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;DEFLT&gt;</code></td>
<td><code>&lt;DEFLT&gt;</code></td>
<td>CREATE TABLE <code>tableName</code>;</td>
<td>(DB2 Version 9) DB2 creates the table in an implicit database (named DSNnnnnn) and an implicit table space.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(DB2 Version 8) DB2 creates the table in a database defined as DSNDB04 and an implicit table space.</td>
</tr>
<tr>
<td>user-defined name DSNDB04</td>
<td><code>&lt;DEFLT&gt;</code></td>
<td>CREATE TABLE <code>tableName</code> IN DATABASE <code>databaseName</code>;</td>
<td>DB2 creates the table in an explicit database and an implicit table space.</td>
</tr>
<tr>
<td><code>&lt;DEFLT&gt;</code></td>
<td>user-defined name</td>
<td>ERROR</td>
<td>You cannot explicitly create a table space in an implicit database in ALTER and CHANGE MANAGER.</td>
</tr>
</tbody>
</table>
For reference, the guidelines in Table 32 apply when you work with implicit table spaces.

**Table 32 Working with implicit table spaces**

<table>
<thead>
<tr>
<th>Situation</th>
<th>DB2 or product response</th>
</tr>
</thead>
<tbody>
<tr>
<td>You create a table that has a ROWID GENERATED BY DEFAULT column in an implicit table space.</td>
<td>DB2 automatically creates an enforcing index.</td>
</tr>
<tr>
<td>You create a base table in an implicit table space.</td>
<td>DB2 automatically creates auxiliary objects to support the table’s large object (LOB) columns.</td>
</tr>
<tr>
<td>You add a ROWID GENERATED BY DEFAULT column (or a LOB column) to a table that exists in an implicit table space within an implicit database and your changes result in a DROP TABLE and a CREATE TABLE statement.</td>
<td>The product re-creates the table space implicitly. <strong>Note:</strong> You do not need to create the enforcing index for the ROWID and the auxiliary objects. However, if you want to maintain control over the names and attributes of the required enforcing index and auxiliary objects, you must re-create the table space explicitly.</td>
</tr>
</tbody>
</table>
| You add a ROWID GENERATED BY DEFAULT column (or a LOB column) to a table that exists in an implicit table space within an explicit database (defined by the user or defined as DSNDB04). | **If your changes result in a DROP TABLE and a CREATE TABLE statement,** the product drops and re-creates the table in an explicit table space. You must create the enforcing index for the ROWID and the auxiliary objects.  
**If your changes result in an ALTER TABLE ADD COLUMN statement,** the product alters the table in the implicit table space. You do not need to create the enforcing index for the ROWID and the auxiliary objects. |
| The table exists in an implicit table space, and you do not create an enforcing index for the primary key or unique constraints. | DB2 automatically creates the index for you. |

**Considerations for changing data structures**

Consider the following items when you specify changes to data structures:

- When you type E in the **Act** column adjacent to an object, the detail panel displays the SQL CREATE statement attributes for that object or element. For example, the **Table Owner** and **Tablespace Name** are attributes of a table. When you change the definition of an object, the detail panel for the object type displays the attributes of the object and a **New Values** column. You type new attribute values in the **New Values** column.
Detail panels sometimes display a value in both the **Current Values** column and the **New Values** column, which indicates that the field is a copied field. A copied field is displayed when a blank value or a 0 (zero) for a numeric attribute has meaning. For example, if an EDITPROC exists for a table, the name of the EDITPROC appears in both the **Current Values** column and the **New Values** column. To eliminate the EDITPROC, you must specify a blank in the **New Values** column. In this example, a blank indicates a deliberate action or value.

If you alter a table in a table space that was created implicitly, and those changes require the table to be rebuilt, Analysis compares the primary quantity for the table space in the DB2 catalog to the default primary quantity value for an implicit table space. If the primary quantity in the catalog is larger than the default value, Analysis re-creates the table space explicitly. If the default value is larger, Analysis re-creates the table space implicitly.

If you have defined an identity column in your table with GENERATED ALWAYS, note that if the table must be dropped and rebuilt the unique values that are assigned to the rows in the column might change when the values are reloaded. You can have the utilities preserve the existing values in the identity column, or you can enable DB2 to generate new unique values for the identity column.

— To have the BMC utilities (BASIC UNLOAD, UNLOAD PLUS, and LOADPLUS) preserve the existing values in the identity column, simply run Analysis and Execution.

By default, the ALUIN input stream includes the NOREGENIDENTITY keyword. NOREGENIDENTITY tells Analysis not to allow DB2 to generate new values for an identity column that is defined as GENERATED ALWAYS if the table is dropped and created. In addition, Analysis adds the IDENTITYOVERRIDE YES and UPDATEMAXA YES parameters to the -BMCL (LOADPLUS) worklist command. IDENTITYOVERRIDE YES tells LOADPLUS to load identity column values from the input file. UPDATEMAXA YES tells LOADPLUS to update the MAXASSIGNEDVAL column of SYSIBM.SYSSEQUENCES when loading identity column values from an input file.

— To have the IBM utilities preserve the existing values in the identity column, perform the following tasks:

1. Specify the GENERATED BY DEFAULT parameter for the identity column.

2. Determine the highest value that is currently assigned to a row. You can use SQL to retrieve the value in the counter in the SYSIBM.SYSSEQUENCES table, or you can enter the MAX command-line command on the Identity Column Detail panel in Specification.
3. Increase the value specified in the START WITH parameter to accommodate inserting additional rows.

4. Run Analysis and Execution. Analysis unloads and loads the current values.

— To enable DB2 to generate new unique values when using the BMC or IBM utilities, manually change the NOREGENIDENTITY ALUIN keyword to REGENIDENTITY. Then, run Analysis and Execution.

The ALTER and CHANGE MANAGER products support various data conversions for existing data. You can change any of the column attributes for existing data, provided that the data can be converted to the new definition.

CHANGE MANAGER supports the conversion of a VARCHAR or LONG VARCHAR data type to a BLOB or CLOB data type. If a column is already defined as a LOB data type, it cannot be converted.

**TIP**

To convert the data type, see “Converting a VARCHAR data type to a LOB data type” on page 543.

**TIP**

For a list of tasks that you can perform to alter data structures, see Table 41 on page 354.

### Importing files

You can use ALTER and CHANGE MANAGER to process files that are received from the same or another subsystem and apply the changes. The products convert the data structures that are stored in the files into change requests in a work ID. After the file is imported, the products create entries in the CD tables as if the entries had been requested in the Specification component. You can then modify the work ID in Specification, analyze it, and execute the resulting worklist to implement the changes.
Deciding what type of file to import

With CHANGE MANAGER, you can import CDL, DDL, and CM/PILOT data manipulation language (DML) files. With ALTER, you can import only DDL files.

Importing CDL

CDL is a BMC proprietary language that supports the creation, dropping, or alteration of DB2 data structures. You can create CDL by using the Compare component of CHANGE MANAGER or by manually creating the CDL commands in a file. (For more information about creating CDL commands, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.) The CDL statements show the differences between two sets of data structures. The CDL file can be either a sequential, 80-column data set or a member of a PDS.

If you want to quickly apply a change to an application, you can convert change requests for an alter-type work ID into CDL statements. You can then use the resulting CDL to alter another instance of the application’s data structures. CHANGE MANAGER converts LIKE and DROP entries into CREATE and DROP CDL commands. When CHANGE MANAGER processes ALTER entries, it generates a CDL ALTER command. If the data structure has not actually been altered, CHANGE MANAGER ignores the entry and generates a message in the diagnostic output file.

NOTE

If the CDL includes a CREATE PROCEDURE statement and other procedures exist in the catalog with the same name, the product imports the statement as a CREATE PROCEDURE ADD VERSION statement.

TIP

To import a CDL file, see page 379.

NOTE

For more information, view the Quick Course Importing CDL. You must have a BMC Support ID to view the Quick Course.

Importing DDL

DDL is a category of Structured Query Language (SQL) statements that create DB2 objects. You can import DDL statements to create your DB2 objects. Then, use the space estimation feature to determine the amount of space that a table space or index will require.
Deciding what type of file to import

The DDL file should meet the following criteria:

- DDL files can be either a sequential, 80-column data set or a member of a partitioned data set (PDS).
- The DDL statements must appear in columns 1 through 72 in free format. The products ignore columns 73 through 80.
- If multiple DDL statements appear in the file, you can use semicolons to separate them.

**NOTE**
The DDL statements can be separated by a character other than the semicolon if the --#SET TERMINATOR value control statement is inserted into the DDL file. The --#SET TERMINATOR statement is the convention that is used by the IBM DSNTEP2 sample dynamic SQL program and SPUFI.

The products use the pound sign (#) as the value. The value can be any character except a blank, comma (,), double quotation mark ("), single quotation mark (‘), left parenthesis ([, right parenthesis (]), or an underscore (_).

**NOTE**
Products ignore comments in a DDL file, such as the authorizations in a baseline report. Comments begin with two dashes (--) and continue until the end of the line. No other comment characters are recognized.

**NOTE**
The --#SET TERMINATOR statement is not treated as a comment.

You should not attempt to import incremental DDL from a file. Incremental DDL modifies objects by dropping and rebuilding them. If you want to make changes to your data structures, it is more efficient to migrate the changes by using the change migration process, which generates CDL. For information, see Chapter 5, “Maintaining database environments.”

**NOTE**
If the DDL file includes a CREATE PROCEDURE statement and other procedures exist in the catalog with the same name, the product imports the statement as a CREATE PROCEDURE statement.

**TIP**
To import a DDL file, see page 381.
Importing DML

DML is a SQL-like data manipulation language that updates, deletes, and migrates data structures, or estimates space. You can create DML statements by using the CM/PILOT component of CHANGE MANAGER, or by manually creating the statements in a file. You can also use the example DML that BMC provides.

The DML file should meet the following criteria:

- The DML files can be either a sequential, 80-column data set or a member of a PDS.

- The DML file can consist of DML statements that are extracted from a CM/PILOT worklist, or the file can be one of the following members in the HLQ.DBCNTL data set:
  - ACMDMLD1
  - ACMDMLD2
  - ACMDMLD3
  - ACMDMLL1
  - ACMDMLL2
  - ACMDMLM1
  - ACMDMLM2
  - ACMDMLP1
  - ACMDMLP2
  - ACMDMLP3
  - ACMDMLT1
  - ACMDMLT2
  - ACMDMLUA
  - ACMDMLUB
  - ACMDMLUC
  - ACMDMLUD
  - ACMDMLUE
  - ACMDMLUF
  - ACMDMLU1
  - ACMDMLU2
  - ACMDMLU3
  - ACMDMLU4
  - ACMDMLU5
  - ACMDMLU6
  - ACMDMLU7
  - ACMDMLU8
  - ACMDMLU9
Deciding the type of work ID to use

A DML file can contain one or more of the following statements:

- DELETE
- LIKE
- MIGRATE
- SET SPACE ESTIMATION
- UPDATE

CHANGE MANAGER does not support importing the DELETE SYNTAXTABLE, DELETE WORKID, REPLICATE WORKID, and SET OMITUNICODE statements.

A DML file cannot contain a MIGRATE statement and a DELETE, LIKE, or UPDATE statement. The statements must occur in separate files.

**TIP**
To import a DML file, see page 379.

**NOTE**
For more information, view the Quick Course CM/PILOT Overview. You must have a BMC Support ID to view the Quick Course.

---

**Deciding the type of work ID to use**

You can import your file into a new alter-type work ID or into an existing alter-type work ID.

**NOTE**
If your DML file contains a MIGRATE statement, CHANGE MANAGER automatically changes the alter-type work ID to a migrate-type work ID.

**Using a new alter-type work ID**

When you import your CDL or DDL file into a new alter-type work ID, CHANGE MANAGER can use change rules in an inbound migrate profile to apply changes to imported data structures. CHANGE MANAGER cannot use an inbound migrate profile when importing a DML file. Inbound migrate profiles are not available for the ALTER product.
Inbound migrate profiles modify the input that is imported from another system to match the receiving subsystem’s version (or local modifications) of the application. Inbound migrate profiles can contain change rules that modify the attributes of data structures. Unlike an outbound migrate profile that is used in the migration process, you cannot specify scope rules or locations with an inbound migrate profile.

The change rules in an inbound migrate profile match the object type and attributes to those in the imported data structures, and apply the rule when a match is found. Change rules enable you to easily specify and repeat common changes. The change rules that you can specify in an inbound migrate profile for an alter-type work ID are the same as those that you would specify for an outbound migrate profile for a migrate-type work ID. For information about specifying change rules, see “Specifying values for change rule attributes” on page 49.

CHANGE MANAGER creates the CD table entries after you import the file. For example, if a change rule specifies to change the database name DEMO* to ACM*, the product reads the DDL statement CREATE DATABASE DEMOHRS and the Import component applies the change rule. The product creates the CD table entries to CREATE DATABASE ACMHRS.

**TIP**
To create an inbound migrate profile, see page 361.

### Using an existing alter-type work ID

When you import a CDL or a DDL file into an existing alter-type work ID, the changes that are in the work ID can be either replaced by or merged with the changes in the file that you import. CHANGE MANAGER cannot use the change rules that are defined in an inbound migrate profile for an existing work ID to change the data structures. When you import a DML file, the changes are replaced by the changes in the file.

If you choose to replace the changes in the work ID, the product replaces the entries in the product CD tables. In addition, the product inserts the REPLACEWORKID keyword into the ALUIN input stream. To manually specify to replace the changes in the work ID, edit the JCL that is created. Change the WORKID keyword to REPLACEWORKID.

If you want to merge the changes that are in the work ID with the changes in the CDL or DDL file, the product imports the changes for objects that are not already stored in the work ID. If the object is already in the work ID, the product ignores changes for that object and issues an error.
Validity checks

When generating change requests from the CDL, DDL, and DML files, the products perform validity and performance checks. Complete validity and performance checks are performed when a worklist is generated by Analysis. Table 33 lists the rules that the products follow for the SQL statements in the CDL or DDL files.

Table 33 Import input rules for CDL and DDL files

<table>
<thead>
<tr>
<th>SQL statement</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE</td>
<td>Import creates a LIKE entry in the CD tables for the object.</td>
</tr>
<tr>
<td>DROP</td>
<td>If the object to be dropped exists in the DB2 catalog, the products create a DROP entry in the CD tables.</td>
</tr>
<tr>
<td>ALTER</td>
<td>If the object to be altered exists in the DB2 catalog, the products match the corresponding old and new values for the attributes to be changed, and create an ALTER entry in the CD tables for those attributes that change. An ALTER statement might also refer to a previous CREATE statement in the DDL.</td>
</tr>
</tbody>
</table>

In addition, if force-type change rules are specified in an inbound migrate profile, CHANGE MANAGER suppresses the changes for that specific attribute.

Validation of object text dependencies

The products validate text for the following objects:

- views
- view dependents
- triggers
- check constraints
- indexes on an expression
- materialized query tables (MQTs)

This feature performs the following functions:

- parses the object for object and attribute dependencies
- applies change rules for name changes to objects and columns
Reviewing the results of the import

In addition to creating the change requests for a work ID, the products also return a completion code and build a diagnostic output log. The diagnostic output log, ALUPRINT, lists the messages and errors that the job generates. The log contains the following information:

- ALUIN parameters
- DB2 subsystem connection message
- summary of objects parsed and processed
- final return code

Table 34 lists some of the return codes that might appear in the diagnostic log.

Table 34 Import return codes

<table>
<thead>
<tr>
<th>Return Code (RC)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no errors or warnings were issued</td>
</tr>
<tr>
<td>4</td>
<td>warnings were issued</td>
</tr>
<tr>
<td>8 or 12</td>
<td>errors were issued</td>
</tr>
</tbody>
</table>

When you import a DML file, the diagnostic output includes a DML report. For more information about DML reports, see “DML report” on page 637.

Correcting errors

If the import was unsuccessful, the diagnostic output log contains messages that indicate the errors. The diagnostic output log might be assigned to TERM (in the foreground), assigned to SYSOUT, or sent to a data set. You cannot assign diagnostic output to a member of a PDS. The messages indicate the type of error, the sequence number of the command that contains the error, and the token where the error was detected. For additional information about messages, see the BMC Documentation Center.
If errors occur while data structures are imported, you can correct the errors by using one of the following methods:

- Delete the work ID that was used, edit the input file to correct the errors, and reimport the file under a new work ID. You can use the same work ID name for both the old and the new work IDs.

- Use Specification to enter the changes that were omitted or modify the changes that were incorrect.

The method that you choose depends on the quantity and severity of errors. For example, assume that every imported table space definition from a CDL or DDL file failed because of naming conflicts. In this case, you can create an inbound migrate profile to modify the table space names, and then reimport the file and use the inbound migrate profile. If only a few errors exist in a very large input file, it might be easier and faster to use Specification to enter the changes that were omitted, rather than to reimport the entire file.

**Estimating space**

Space estimation enables you to determine the amount of space that a table space or index will require based on the data structure definitions and their estimated usages. You can specify estimates of the number of rows in the table, the average length of the rows, and the average length of the index key. Using the CM/PILOT component of CHANGE MANAGER, you can specify space estimation parameters by using DML statements, user-defined tables, and other work IDs.

You can use space estimation for the following purposes:

- to project the number of tracks, cylinders, or blocks required to allocate the data set for that object

- to estimate the space requirements when you create a new structure or when you copy and modify an existing structure

- to estimate the space requirements for existing table spaces and indexes when the work load changes

**NOTE**

Note the following items when you estimate space:

- Ensure that you make any changes to your structures before you estimate space.
- You cannot resize the DB2 catalog or directory tables.
- Space estimation is not supported for auxiliary table spaces or indexes.
ALTER and CHANGE MANAGER can perform table and index space estimation by using statistics from the tables in the DASD MANAGER PLUS database (with the BMCSTATS utility), if the following conditions exist:

- you have the DASD MANAGER PLUS product installed (the DASDMAN installation option is set to Y)
- the installer selected to interface ALTER or CHANGE MANAGER with DASD MANAGER PLUS

If DASD MANAGER PLUS is not installed, ALTER and CHANGE MANAGER use the DB2 catalog statistics (from the IBM RUNSTATS utility) to perform space estimation. If the DB2 catalog statistics are not available, ALTER and CHANGE MANAGER use projected values for estimating table and index space. The products can also estimate space based on values that you provide.

**Estimating space for a table space**

The space estimate for a table space is based on the number of pages, which is determined by the following attributes:

- percent of free space on each page
- free page frequency
- number of rows expected in the table or partition
- average row length
- size for each partition
- device type (for example, 3380)
- percent compressed (reduces the average row length)
- maximum number of partitions (for partition-by-growth table spaces)

You can estimate the space for a table space and modify the data structure definitions in the Tablespace Estimation panel, which is displayed in Figure 80.
Estimating space for a table space

To estimate the space for a table space, you can modify the values for the fields that are listed in Table 35. When you modify one or more of these values, a space estimation marker ($) is added adjacent to the object on the Mixed List panel. The marker denotes that the object is stored in the CD tables by Specification solely because of space estimation parameters, and that no changes are requested which would affect analysis.

### Table 35 Attributes for estimating the table space (part 1 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Type</td>
<td>the type of disk device on which the DB2 data set is to be allocated</td>
</tr>
<tr>
<td>Percent Compressed</td>
<td>the percentage used to reduce the average row length</td>
</tr>
</tbody>
</table>

### TIP

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.
To modify the data structure definitions, you can edit the values for the fields that are listed in Table 36. When you modify the definition, an “A” is added adjacent to the object on the Mixed List panel. This marker indicates that a change request has been made for the object.

### Table 35 Attributes for estimating the table space (part 2 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Row Length</td>
<td>the average row length for the table</td>
</tr>
<tr>
<td></td>
<td>The <strong>Avg Orig</strong> field indicates the origination or source of the <strong>Avg Row Length</strong> field. To derive the value of the <strong>Avg Row Length</strong>, the products gather information from one of the following sources (shown in order):</td>
</tr>
<tr>
<td></td>
<td>1. if the DASDMAN installation option is Y, statistics that the BMCSTATS utility gathers in the DASD MANAGER PLUS tables (Stat)</td>
</tr>
<tr>
<td></td>
<td>2. statistics that the IBM RUNSTATS utility gathers in the DB2 system catalog (Rnst)</td>
</tr>
<tr>
<td>Avg Row Length</td>
<td>3. a projected value calculated by the products (Proj)</td>
</tr>
<tr>
<td>NbrRows</td>
<td>the number of rows in the table</td>
</tr>
</tbody>
</table>

Table 35 Attributes for estimating the table space (part 2 of 2)

To modify the data structure definitions, you can edit the values for the fields that are listed in Table 36. When you modify the definition, an “A” is added adjacent to the object on the Mixed List panel. This marker indicates that a change request has been made for the object.

### Table 36 Attributes for modifying the table space (part 1 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segsize</td>
<td>the segment size for the segmented table space</td>
</tr>
<tr>
<td>Priqty</td>
<td>the primary space (expressed in <strong>Allocation Unit</strong>) that is allocated for the DB2-defined data set</td>
</tr>
<tr>
<td></td>
<td>This value must be at least 1. Valid values are 1 through 67,108,864. The default value comes from the catalog or data set. If the Allocation Unit is K, the value will be rounded to a multiple of 4 KB. For example, if the Priqty is zero (0), the value is rounded to 4 KB.</td>
</tr>
<tr>
<td></td>
<td>Entering the PE action code (or line command) for a partition propagates the estimated space to the Priqty field. Entering the PEALL command-line command propagates the estimated space to the Priqty field for all of the partitions in the table space.</td>
</tr>
</tbody>
</table>
### Table 36  Attributes for modifying the table space  (part 2 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secqty</td>
<td>the secondary space (expressed in Allocation Unit) allocated for the DB2-defined data set</td>
</tr>
<tr>
<td></td>
<td>Valid values are 0 through 4,194,304. The default value comes from the catalog or data set. If the Allocation Unit is K, the value will be rounded to a multiple of 4 KB. For example, if the SecQty is seven (7), the value is rounded to 8 KB.</td>
</tr>
<tr>
<td>Allocation Unit</td>
<td>the unit to use for space estimation calculations, as well as primary and secondary quantities</td>
</tr>
<tr>
<td></td>
<td>The default value comes from the installation standards. Possible values are K (kilobytes), T (tracks), or C (cylinders).</td>
</tr>
<tr>
<td>Freepage</td>
<td>how often to leave a free page when the index, table space, or partition is loaded or reorganized</td>
</tr>
<tr>
<td></td>
<td>Valid values are 0 through 255. The default value is 0, leaving no free pages.</td>
</tr>
<tr>
<td>Pctfree</td>
<td>the percentage of each page to leave free when the table space or partition is loaded or reorganized</td>
</tr>
<tr>
<td></td>
<td>Valid values are 0 through 99. The default value is 5.</td>
</tr>
<tr>
<td>Compress</td>
<td>specifies whether compression is enabled for data within the table space or partition</td>
</tr>
<tr>
<td></td>
<td>Changing the value of Compress from N to Y does not affect “what if” estimates using Percent Compressed.</td>
</tr>
<tr>
<td>Maxrows</td>
<td>the maximum number of rows on each data page</td>
</tr>
<tr>
<td>Dssize</td>
<td>in a partitioned table space, the maximum size for each partition in each data set</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> When you import a CDL or DDL file that does not include the DSSIZE for an auxiliary table space, the Import component assigns a value of 4,194,304 bytes to the DSSIZE.</td>
</tr>
<tr>
<td>Maxpartitions</td>
<td>for partition-by-growth table spaces, the maximum number of partitions</td>
</tr>
</tbody>
</table>

**TIP**

To estimate the space requirements for a table space, see page 453.
Estimating space for an index

The space estimate for an index is based on pages and levels, which are determined by the following attributes:

- percent of free space on each page
- free page frequency
- the maximum length of the index key (for padded indexes) or the average length of the index key (for nonpadded indexes)
- number of rows expected in the table
- average number of rows for each nonunique key
- piecesize for nonpartitioned indexes
- the unique rule: whether the index key is unique (U), defined as UNIQUE WHERE NOT NULL (W), or allows duplicates (D)
- page size (for compressed indexes)

You can estimate the space for an index and modify the data structure definitions in the Index Space Estimation panel, which is displayed in Figure 81.

Figure 81   Index Space Estimation panel (partitioned)
To estimate the space for an index, you can modify the values for the fields that are listed in Table 37. When you modify one or more of these values, a space estimation marker ($) is added adjacent to the object on the Mixed List panel. The marker denotes that the object is put away by Specification solely because of space estimation parameters, and that no changes are requested which would affect Analysis.

Table 37  Attributes for estimating index space

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NbrRows</td>
<td>the estimated number of rows in the partition</td>
</tr>
<tr>
<td>Rows/Key</td>
<td>the estimated number of rows per key value</td>
</tr>
<tr>
<td>Avg Len (Average Key Length)</td>
<td>for padded indexes, the maximum key length; for nonpadded indexes, the average length of the index key</td>
</tr>
</tbody>
</table>

The Key Orig field indicates the origination or source of the Avg Len field. To derive the value of the Avg Len field, the products gather information from one of the following sources (shown in order):

1. if the DASDMAN installation option is Y, statistics that the BMCSTATS utility gathers in the DASD MANAGER PLUS tables (Stat)
2. statistics that the IBM RUNSTATS utility gathers in the DB2 system catalog (Rnst)
3. a projected value calculated by the products (Proj)

If a value for the average key length was saved in the product CD tables, space estimation uses that value, unless the value was projected. In that case, the products recalculate the value.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devt</td>
<td>the type of disk device on which the DB2 data set is to be allocated</td>
</tr>
<tr>
<td></td>
<td>Valid values are 3330, 3340, 3350, 3375, 3380, 3390.</td>
</tr>
<tr>
<td>Pct Compressed</td>
<td>the percentage used to reduce the average key length</td>
</tr>
<tr>
<td></td>
<td>User-supplied percentages are stored on the CD tables.</td>
</tr>
<tr>
<td></td>
<td>Valid values are 00 through 99. The default is 0.</td>
</tr>
</tbody>
</table>

To modify the data structure definitions, you can edit the values for the fields that are listed in Table 38. When you modify the definition, an “A” is added adjacent to the object on the Mixed List panel. This marker indicates that a change request has been made for the object.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.
### Table 38 Attributes for modifying the index space (part 1 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique</td>
<td>whether the index key is unique or allows duplicates&lt;br&gt;Valid values are U (Unique), D (Duplicates), and W (defined as UNIQUE WHERE NOT NULL).</td>
</tr>
</tbody>
</table>
| Piecesize | maximum piecesize for nonpartitioned indexes<br>Avoid wasting space by ensuring that the sum of the PriQty and the SecQty values evenly divide into the Piecesize value.  
**Note:** When you import a CDL or DDL file that does not include the PIECESIZE for an auxiliary index, the Import component performs the following functions:
- If a value is specified for the DSSIZE for the auxiliary table space, Import uses that value for the PIECESIZE.
- If a value is not specified for the DSSIZE for the auxiliary table space, Import assigns a value of 4,194,304 bytes for the PIECESIZE. |
| Compress | whether the index uses compression |
| PriQty | the primary space (expressed in Aloc Unit) that is allocated for the DB2-defined data set<br>This value must be at least 1. Valid values are -1 and 1 through 67,108,864. The default value comes from the catalog or data set. If the Allocation Unit is K, the value will be rounded to a multiple of 4 KB. For example, if the PriQty is one (1), the value is rounded to 4 KB. If a value of -1 is specified for the PriQty, space is estimated with a PriQty of 4 and a SecQty of 0.<br>Entering the PE action code (or line command) for a partition propagates the estimated space to the PriQty field. Entering the PE command-line command propagates the estimated space to the PriQty field for all of the partitions in the index.<br>To ensure that you do not waste space, specify a value for the PriQty that does not exceed the value of Piecesize. |
| SecQty | the secondary space (expressed in Aloc Unit) allocated for the DB2-defined data set<br>Valid values are -1 and 0 through 4,194,304. The default value comes from the catalog or data set. If the Allocation Unit is K, the value will be rounded to a multiple of 4 KB. For example, if the SecQty is seven (7), the value is rounded to 8 KB. If a value of -1 is specified for the SecQty, space is estimated with a SecQty of 0. |
Estimating space requirements based on user-specified values

Standard space-estimation tools use IBM default DDL values for object attributes when calculating estimates. In contrast, the Simple Space Estimation (SSE) feature allows you to replace those defaults with values that are specific to your objects. DASD MANAGER PLUS, CATALOG MANAGER, and CHANGE MANAGER support this feature.

**NOTE**
You do not need to run BMCSTATS before using SSE.

For example, for a non-partitioned table space estimate, you can change the fields highlighted in Figure 82. SSE then estimates how much space will be required if you reorganize the table space. SSE displays estimates for the entire table space in the **Estimated** box on the right, and table-level estimates at the bottom of the panel.

---

**Table 38 Attributes for modifying the index space (part 2 of 2)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU (Allocation Unit)</td>
<td>the unit to use for space estimation calculations, as well as primary and secondary quantities</td>
</tr>
<tr>
<td></td>
<td>The default value comes from the installation standards. Possible values are K (kilobytes), T (tracks), or C (cylinders).</td>
</tr>
<tr>
<td>Free Page</td>
<td>how often to leave a free page when the index or partition is loaded or reorganized</td>
</tr>
<tr>
<td></td>
<td>Valid values are 0 through 255. The default value is 0, leaving no free pages.</td>
</tr>
<tr>
<td>Pct Free</td>
<td>the percentage of each page to leave free when the partition is loaded or reorganized</td>
</tr>
<tr>
<td></td>
<td>Valid values are 0 through 99. The default value is 10.</td>
</tr>
<tr>
<td>NonLf KeyLn</td>
<td>the average key length of nonleaf pages</td>
</tr>
<tr>
<td></td>
<td>Valid values are 0 through 99999. The default value is 0.</td>
</tr>
</tbody>
</table>

---

**TIP**
To estimate the space requirements for an index, see page 460.
Analyzing changes

For a description of each field that accepts user-specified values, see the online Help.

**Figure 82  Tablespace Estimation panel**

Analyzing changes

After you create the change requests in an alter-type work ID, Analysis checks the requests for validity with the DB2 catalog, develops an optimal implementation strategy, and generates a worklist. The worklist contains the utility commands, AMS commands, DB2 commands, and SQL statements that are necessary for implementing the changes. In addition, Analysis propagates changes into dependent structures.

When you analyze an alter-type work ID, you have the option to generate a worklist that establishes a full-recovery baseline or a worklist that falls back to a full-recovery baseline. For information about creating full-recovery baselines, see Chapter 2, “Taking a snapshot of a database environment.” For information about falling back to full-recovery baselines, see Chapter 5, “Maintaining database environments.”
Overriding the default processing options

Analysis uses the processing options that you set through Options in the Main Menu to generate analysis JCL and to generate keywords in the ALUIN parameter input data stream. How you run Analysis determines how the options are processed:

- If you run Analysis in the foreground, Analysis uses the product options and the installation options to generate the worklist.

- If you run Analysis in batch and use foreground to generate JCL, Analysis uses the product options and the installation options to generate the worklist.

- If you run Analysis in batch and apply user-generated JCL, Analysis uses any manually-inserted keywords that are specified in the ALUIN input stream and the installation options to generate the worklist. In this case, Analysis does not use the product options.

You can temporarily override some of the installation options through the Analysis Options panels. Remember that the values you set through these panels are not stored in your ISPF profile. They only remain in effect for that invocation of analysis, until you return to the WORKID Action Menu or to the WORKID Selection List.

The following panels provide a way to override the default settings for utilities and statistics:

- Analysis Utility Options
- Analysis Reorg Options
- Analysis Image Copy Options
- Analysis Utility Dataset Options
- Analysis Statistics Options
- Analysis Unload / Copy Options

For more information about using utilities with CHANGE MANAGER, see Chapter B, “Using utilities with ALTER and CHANGE MANAGER.”
Validity and performance checks

The Analysis component performs much the same validity and performance checks when it analyzes the change requests as it does for requests for migration. Analysis checks for objects that are affected by the requested changes. For information about the other validity and performance checks that are performed, see “Validity and performance checks” on page 73.

Reviewing the results of Analysis

When you alter data structures, Analysis produces two types of output: worklists and diagnostic output.

Reviewing the alter-type worklist

When you change data structures, Analysis generates a special type of worklist, called an alter-type worklist. The alter-type worklist is a sequential, 80-column data set (or PDS member) that contains worklist commands. The worklist also includes a list of the keywords in the ALUIN input stream. When you run Analysis in batch mode, the worklist data set is identified by the ddname WORKL001.

NOTE

Analysis generates the worklist commands in a specific order. You can edit the worklist, but changing the order of commands might yield incorrect results.

Each worklist command (such as -TIME, -SSID, or -SQL) consists of a dash followed by a three- or four-character command identifier, and a blank in column 6. The command identifier is followed by a six-digit sequence number. The sequence number is used when worklists are restarted. The contents of the command line following the sequence number vary according to the command. For example, the -AUTH command requires a user ID to follow the sequence number. The last line of each command, in columns 73–80, contains a hash verification number. CHANGE MANAGER uses the hash number for diagnostics on the worklist file to detect changed commands and inserted commands. Do not insert or modify this number.
For more information about worklist commands, see the \textit{ALTER and CHANGE MANAGER for DB2 Reference Manual}.

When worklist parallelism is enabled for an alter-type worklist, Analysis uses the same parallelism commands as it does in a migrate-type worklist. A worklist can be executed in parallel only in the Database Administration solution.

\section*{Editing the alter-type worklist}

Certain types of tasks involve modifying worklists and data input streams. The ability to modify a worklist, or to edit the input stream for Analysis or Execution, enables you to customize tasks. For more information, see "Editing the migrate-type worklist" on page 79.

\section*{Reviewing the diagnostic output}

Analysis also produces diagnostic output, ALUPRINT, which consists of status messages, error messages, and warnings that are generated during the analysis process. The diagnostic output might be assigned to TERM (in foreground), assigned to SYSOUT, or sent to a sequential file. It might not be assigned to a member of a partitioned data set.

Several other conditions can cause Analysis to issue warnings or to halt processing. For a description of some of the checks that can generate warnings or errors, see "Validity and performance checks" on page 73. For information about error and warning messages, see the \textit{Administrative Products for DB2 Messages Manual}.

\section*{Generating JCL}

When you generate JCL for Execution, the Front End component uses symbolic variables to resolve all data set names that appear on the panels of the components. Parameters (from the Execution panels) are passed to JCL Generation using the AJXIN input stream. These parameters include the names of input files, JCL files, and the diagnostic output files. After the data set names have been resolved, the JCL Generation component performs several functions. These functions are the same as those that are performed when you generate JCL for a migrate-type work ID. For more information about generating JCL, see "Generating JCL" on page 83.
Executing changes

The Execution component performs the commands in a worklist that are generated by ALTER and CHANGE MANAGER. Execution is the only component of ALTER and CHANGE MANAGER that makes physical changes to DB2 data structures or data. Specification and Import are used to build a set of change requests, while Analysis generates a worklist from those requests. Execution must execute SQL, unload DB2 tables, switch DB2 authorization IDs, run DB2 utilities, establish synchronization (sync) points, and perform restarts.

---

**TIP**

To execute a worklist, see page 562.

---

**NOTE**

For more information, view the Quick Course Executing a Worklist. You must have a BMC Support ID to view the Quick Course.

---

### Specifying options for Execution

The Execution Interface panels provide you with a way to specify processing options and direct the flow of the Execution job. In addition to specifying options for processing a worklist, you can specify JCL generation options.

For information about specifying your options for Execution and overriding the installation options, see “Specifying options for Execution” on page 82.

### Creating a multi-step job

In CHANGE MANAGER, when you generate Execution JCL from the Front End, you can optionally create a multi-step job. The job can include such steps as creating a baseline and performing a comparison before or after the actual execution run. Use this feature to consolidate jobs that are commonly performed at the same time into one job. For example, you can combine updating data structures, establishing recovery points, and generating change records into one job. You can create a multi-step job by using the Execution Pre- and Post-Processing Interface panel, as shown in Figure 83.
NOTE
Because Compare and Baseline are not components in the ALTER product, you cannot create a multi-step job in ALTER.

Figure 83  Execution Pre- and Post-Processing Interface panel

Example

A typical scenario would be to establish a baseline on a set of structures, modify the structures, and then compare the old structures (in the baseline) to the new structures, and generate CDL. If the preceding tasks are performed as a single JCL job stream, the name of the baseline that is established is not known at the time that the job is submitted. Instead, the pre-Execution baseline establishes the baseline and generates a new baseline name. Execution runs the worklist and changes the database. Finally, a post-Execution comparison passes the BLWORKID keyword and looks up the name of the baseline that was established before the worklist was executed. Execution uses that baseline as a primary input. The scope rules from the baseline profile that are used to establish that baseline select the catalog objects to be compared.

Use of name templates

The Compare component, as with the Import, Analysis, and Execution components, supports the use of name templates for alter-type work IDs in batch mode. You build the JCL data set to specify the template in the ALUIN input stream. The most current alter-type work ID that matches the name template is used. The name template characters are #####, @@@@@@, and @@@@@@@@, and are defined in Table 39.
Table 39  Name template characters

<table>
<thead>
<tr>
<th>Characters</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>####</td>
<td>a sequence of four numbers; a counter that is incremented by 1 by template</td>
</tr>
<tr>
<td>@@@@@@@@</td>
<td>a date in the format of YYMMDD</td>
</tr>
<tr>
<td>@@@@@@@@@@</td>
<td>a date in the format of YYYYMMMDD</td>
</tr>
</tbody>
</table>

Figure 84 shows the use of a name template for a baseline to catalog comparison that uses an alter-type work ID as the primary input.

Figure 84  Example of name template

SSID DEBA
COMMIT 500
SOURCETYPE CATALOG
BLPROFILE RDACRJ.BLPROF1
BLNAME DEMO####
WORKID RDACRJ.ALT03

You can use name templates in a multi-step batch job if you periodically receive CDL files to import and process. As the last step in the job, you might want to compare a pre-Execution baseline with the catalog to record the changes that were made. For example, you can create the job with the steps shown in Table 40.

Table 40  Comparing a baseline to a catalog

<table>
<thead>
<tr>
<th>Step</th>
<th>Component</th>
<th>ALUIN keyword used</th>
<th>Work ID used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Import</td>
<td>NEWWORKID</td>
<td>MJF.CDL####</td>
</tr>
<tr>
<td>2</td>
<td>Analysis</td>
<td>WORKID</td>
<td>MJF.CDL####</td>
</tr>
<tr>
<td>3</td>
<td>Baseline</td>
<td>WORKID</td>
<td>MJF.CDL####</td>
</tr>
<tr>
<td>4</td>
<td>Execution</td>
<td>WORKID</td>
<td>MJF.CDL####</td>
</tr>
<tr>
<td>5</td>
<td>Compare</td>
<td>BLWORKID</td>
<td>MJF.CDL####</td>
</tr>
</tbody>
</table>

Considerations

Take the following items into consideration when you create a multi-step job:

- Do not generate JCL for a pre-Execution baseline in jobs that establish a full-recovery baseline, or in any execution of a worklist that contains the -BASE command. Because the work ID that creates these baselines is stored along with the baseline information, you would be attempting to establish two baselines with the same work ID, and the second one would terminate with an error.
When the Front End creates the multi-step job with Execution, the job terminates if any of the parts returns an error. If the job terminates and you want to restart it, you should inspect the diagnostic output to determine the job step that caused the error. For example, if a pre-Execution baseline caused the error, you should not use the RESTART keyword with Execution after fixing the error because the worklist Execution run has not actually started yet. Likewise, if a post-Execution job step failed, you should not have to rerun or restart Execution because it completed successfully. In that instance, you should edit the JCL and remove the successful job steps before attempting to rerun the job step that failed.

**NOTE**

Because Compare and Baseline are not components in the ALTER product, you cannot create a multi-step job in ALTER.

**TIP**

To generate a multi-step job, see page 564.

## Controlling authorizations

The Execution component performs an authorization switching function to ensure that DB2 structures are created with the proper owner and creator. To secure this function, restrict the EXECUTE authority on the Execution Main plan, use the Execution security exit, or do both. For more information about switching and controlling authorizations, see “Controlling authorizations” on page 94.

## Processing a worklist

The Execution worklist data set contains all of the worklist commands, SQL statements, and utility commands that are needed to implement the requested changes, including

- unloading data
- dropping and re-creating or altering all affected objects, privileges, and referential constraints
- reorganizing table spaces and indexes, and rebuilding indexes
- reloading data
Reviewing the results of your changes

- collecting statistics
- making image copies
- checking the data
- rebinding application plans and packages

**NOTE**
No changes are actually made to the DB2 catalog until the worklist is executed.

An alter-type worklist is processed in much the same way as a migrate-type worklist is processed. An alter-type worklist, however, is not separated into two phases.

**NOTE**
If you executed an alter-type work ID and need to restart the worklist after SQL DROP statements have processed, generate the restart JCL by selecting the **Build Restart JCL from previous execution JCL** option on the Execution JCL Build Interface panel. This method avoids problems with the substitution of symbolics in the data set names.

For more information about processing a worklist, including viewing the status of the execution, using sync points, restarting a worklist, or starting a worklist over, see “Processing a worklist” on page 98.

**Reviewing the results of your changes**

The primary output from Execution is the modified data structures that are stored in the DB2 catalog and any cataloged output from the utilities such as image copy data sets. However, the exact nature of these modifications depends on the content of the worklist that is being executed. In addition to the DB2 modifications, Execution produces a diagnostic output file.

The diagnostic output file, also called a worklog or SYSOUT, is a log or audit trail of all input statements shown in their entirety before execution. The worklog also contains the responses that are caused by those actions. It is the primary tool for tracking events during execution. The worklog ddname is AEXPRINT.

When Execution runs component programs, it captures the SYSPRINT outputs from those programs in the worklog. Lines beginning with a colon (:) echo the input worklist commands to the worklog. All other lines are generated by Execution.
Accomplishing your goals

When a worklist is executed in parallel in the Database Administration solution, Execution produces the output files listed in Table 10 on page 103.

Accomplishing your goals

ALTER and CHANGE MANAGER enable you to create, modify, and drop data structures within a DB2 subsystem. The products also enable you to import changes to data structures from DDL files and CDL files. You can also use scripts in the CM/PILOT component of CHANGE MANAGER to change your data structures. (For more information about scripts, see Appendix A, “Using scripting tools to automate change management.”) Table 41 lists the tasks that you can perform to alter data structures.

Table 41  Alter tasks  (part 1 of 4)

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating and maintaining ALTER and CHANGE MANAGER objects</td>
<td></td>
</tr>
<tr>
<td>“Creating an alter-type work ID”</td>
<td>page 357</td>
</tr>
<tr>
<td>“Creating an alter-type work ID in batch mode”</td>
<td>page 358</td>
</tr>
<tr>
<td>“Sorting work IDs”</td>
<td>page 359</td>
</tr>
<tr>
<td>“Deleting a work ID”</td>
<td>page 360</td>
</tr>
<tr>
<td>“Creating an inbound migrate profile in CHANGE MANAGER”</td>
<td>page 361</td>
</tr>
<tr>
<td>“Replicating multiple work IDs”</td>
<td>page 362</td>
</tr>
<tr>
<td>“Modifying a work ID with values in a user-defined table”</td>
<td>page 366</td>
</tr>
<tr>
<td>“Changing column definitions for imported DDL”</td>
<td>page 370</td>
</tr>
<tr>
<td>“Deleting multiple work IDs”</td>
<td>page 373</td>
</tr>
<tr>
<td>“Deleting sync table entries for multiple work IDs”</td>
<td>page 376</td>
</tr>
<tr>
<td>Importing files</td>
<td></td>
</tr>
<tr>
<td>“Importing a CDL, DDL, or DML file in CHANGE MANAGER”</td>
<td>page 379</td>
</tr>
<tr>
<td>“Importing a DDL file in ALTER”</td>
<td>page 381</td>
</tr>
<tr>
<td>Specifying changes to objects</td>
<td></td>
</tr>
<tr>
<td>“Specifying changes to data structures”</td>
<td>page 383</td>
</tr>
<tr>
<td>“Creating a primary constraint for a table”</td>
<td>page 384</td>
</tr>
<tr>
<td>“Creating a unique constraint for a table”</td>
<td>page 386</td>
</tr>
<tr>
<td>“Editing the columns for a unique constraint”</td>
<td>page 390</td>
</tr>
<tr>
<td>“Editing a primary key”</td>
<td>page 392</td>
</tr>
<tr>
<td>“Naming a primary key”</td>
<td>page 395</td>
</tr>
<tr>
<td>“Changing the attributes of an index”</td>
<td>page 397</td>
</tr>
<tr>
<td>“Creating indexes for ROWID GENERATED ALWAYS columns by importing DML”</td>
<td>page 400</td>
</tr>
</tbody>
</table>
### Table 41  Alter tasks (part 2 of 4)

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Adding a column to a table”</td>
<td>page 402</td>
</tr>
<tr>
<td>“Dropping a column from a table”</td>
<td>page 404</td>
</tr>
<tr>
<td>“Changing the data type and length of a column”</td>
<td>page 405</td>
</tr>
<tr>
<td>“Moving multiple tables using Quick Edit”</td>
<td>page 406</td>
</tr>
<tr>
<td>“Changing data structures by using an existing work ID”</td>
<td>page 408</td>
</tr>
<tr>
<td>“Receiving DDL to create data structures”</td>
<td>page 411</td>
</tr>
<tr>
<td>“Changing data structures by using DML”</td>
<td>page 415</td>
</tr>
<tr>
<td>“Adding columns to the DB2 PLAN_TABLE table by importing DML”</td>
<td>page 420</td>
</tr>
<tr>
<td>“Deleting aliases by importing DML”</td>
<td>page 422</td>
</tr>
<tr>
<td>“Copying a column by importing DML”</td>
<td>page 423</td>
</tr>
<tr>
<td>“Updating index partitions for a database by importing DML”</td>
<td>page 424</td>
</tr>
<tr>
<td>“Adding WITH Restrict ON DROP to tables by importing DML”</td>
<td>page 426</td>
</tr>
<tr>
<td>“Updating table spaces or indexes with existing data sets by importing DML”</td>
<td>page 427</td>
</tr>
<tr>
<td>“Creating synonyms for selected tables by importing DML”</td>
<td>page 428</td>
</tr>
<tr>
<td>“Converting VCAT-defined partitions to STOGROUP-defined partitions by importing DML”</td>
<td>page 429</td>
</tr>
<tr>
<td>“Creating a materialized query table (MQT)”</td>
<td>page 431</td>
</tr>
<tr>
<td>“Creating an application-period temporal table”</td>
<td>page 433</td>
</tr>
<tr>
<td>“Creating a system-period temporal table”</td>
<td>page 435</td>
</tr>
<tr>
<td>“Creating a history table for a system-period temporal table”</td>
<td>page 437</td>
</tr>
<tr>
<td>“Creating a history table from an existing history table”</td>
<td>page 440</td>
</tr>
<tr>
<td>“Creating a history table from a base table”</td>
<td>page 443</td>
</tr>
<tr>
<td>“Creating an external stored procedure”</td>
<td>page 446</td>
</tr>
<tr>
<td>“Creating a native SQL stored procedure”</td>
<td>page 447</td>
</tr>
<tr>
<td>“Creating a new version of a native SQL stored procedure”</td>
<td>page 449</td>
</tr>
<tr>
<td>“Updating options for a native SQL stored procedure by importing DML”</td>
<td>page 451</td>
</tr>
<tr>
<td>“Updating parameters for a native SQL stored procedure by importing DML”</td>
<td>page 452</td>
</tr>
<tr>
<td><strong>Changing space for an object</strong></td>
<td></td>
</tr>
<tr>
<td>“Estimating the space requirements for a table space”</td>
<td>page 453</td>
</tr>
<tr>
<td>“Estimating the space requirements for an index”</td>
<td>page 460</td>
</tr>
<tr>
<td>“Estimating space requirements for a table space based on user-specified values”</td>
<td>page 464</td>
</tr>
<tr>
<td>“Estimating space requirements for an index based on user-specified values”</td>
<td>page 466</td>
</tr>
<tr>
<td>“Setting space estimation parameters by importing DML”</td>
<td>page 467</td>
</tr>
</tbody>
</table>
### Table 41  Alter tasks (part 3 of 4)

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Estimating the primary and secondary quantities in table space and</td>
<td>page 470</td>
</tr>
<tr>
<td>index partitions by importing DML”</td>
<td></td>
</tr>
<tr>
<td>“Setting a secondary quantity in table space and index</td>
<td>page 471</td>
</tr>
<tr>
<td>partitions by importing DML”</td>
<td></td>
</tr>
<tr>
<td><strong>Working with table spaces</strong></td>
<td></td>
</tr>
<tr>
<td>“Creating objects for the SHRLEVEL CHANGE HSSC process”</td>
<td>page 473</td>
</tr>
<tr>
<td>“Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC</td>
<td>page 476</td>
</tr>
<tr>
<td>process”</td>
<td></td>
</tr>
<tr>
<td>“Converting table spaces to partition-by-growth table spaces by</td>
<td>page 479</td>
</tr>
<tr>
<td>importing DML”</td>
<td></td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to an index-controlled</td>
<td>page 480</td>
</tr>
<tr>
<td>partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to a table-controlled</td>
<td>page 486</td>
</tr>
<tr>
<td>partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to a range-partitioned</td>
<td>page 490</td>
</tr>
<tr>
<td>table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to a partition-by-growth</td>
<td>page 491</td>
</tr>
<tr>
<td>table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting a partitioned table space to a nonpartitioned table</td>
<td>page 493</td>
</tr>
<tr>
<td>space”</td>
<td></td>
</tr>
<tr>
<td>“Changing a partitioned table space to a range-partitioned table</td>
<td>page 495</td>
</tr>
<tr>
<td>space by importing DML”</td>
<td></td>
</tr>
<tr>
<td>“Converting a table-controlled partitioned table space to an index-</td>
<td>page 497</td>
</tr>
<tr>
<td>controlled partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting table-controlled partitioned table spaces to range-</td>
<td>page 498</td>
</tr>
<tr>
<td>partitioned table spaces by importing DML”</td>
<td></td>
</tr>
<tr>
<td>“Converting a range-partitioned table space to an index-controlled</td>
<td>page 500</td>
</tr>
<tr>
<td>partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting an index-controlled partitioned table space to a</td>
<td>page 500</td>
</tr>
<tr>
<td>table-controlled partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting index-controlled partitioned table spaces to table-</td>
<td>page 504</td>
</tr>
<tr>
<td>controlled partitioned table spaces by importing DML”</td>
<td></td>
</tr>
<tr>
<td>“Converting an index-controlled partitioned table space to a range-</td>
<td>page 505</td>
</tr>
<tr>
<td>partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting index-controlled partitioned table spaces to range-</td>
<td>page 506</td>
</tr>
<tr>
<td>partitioned table spaces by importing DML”</td>
<td></td>
</tr>
<tr>
<td>“Converting a partitioned table space to a partition-by-growth</td>
<td>page 507</td>
</tr>
<tr>
<td>table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting a partition-by-growth table space to an index-</td>
<td>page 510</td>
</tr>
<tr>
<td>controlled partitioned table space”</td>
<td></td>
</tr>
<tr>
<td>“Converting a partition-by-growth table space to a table-controlled</td>
<td>page 512</td>
</tr>
<tr>
<td>partitioned table space”</td>
<td></td>
</tr>
</tbody>
</table>
### Alter tasks (part 4 of 4)

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Converting a partition-by-growth table space to a range-partitioned table space”</td>
<td>page 513</td>
</tr>
<tr>
<td>“Changing a single-table table space to a partition-by-growth table space by importing DML”</td>
<td>page 514</td>
</tr>
<tr>
<td>“Changing tables in an explicit database to an implicit database by importing DML”</td>
<td>page 515</td>
</tr>
<tr>
<td>“Converting explicit databases and implicit table spaces to implicit databases and table spaces by importing DML”</td>
<td>page 517</td>
</tr>
<tr>
<td>“Changing the encoding scheme for a table space by importing DML”</td>
<td>page 518</td>
</tr>
</tbody>
</table>

### Working with LOBs

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Adding a ROWID and a LOB column to create a base table”</td>
<td>page 519</td>
</tr>
<tr>
<td>“Creating auxiliary objects for a partitioned table space”</td>
<td>page 525</td>
</tr>
<tr>
<td>“Creating auxiliary objects for a single partition in a partitioned table space”</td>
<td>page 531</td>
</tr>
<tr>
<td>“Liking a base table and creating auxiliary objects for a table space”</td>
<td>page 536</td>
</tr>
<tr>
<td>“Creating auxiliary objects for a nonpartitioned table space”</td>
<td>page 538</td>
</tr>
<tr>
<td>“Converting a VARCHAR data type to a LOB data type”</td>
<td>page 543</td>
</tr>
<tr>
<td>“Creating a new database and new table spaces for tables by importing DML”</td>
<td>page 548</td>
</tr>
<tr>
<td>“Changing multi-table table spaces to single-table table spaces by importing DML”</td>
<td>page 550</td>
</tr>
<tr>
<td>“Moving all empty tables in a database to one table space by importing DML”</td>
<td>page 552</td>
</tr>
<tr>
<td>“Setting table spaces and indexes to DEFINE NO by importing DML”</td>
<td>page 554</td>
</tr>
<tr>
<td>“Updating table spaces or indexes defined with DEFINE NO by importing DML”</td>
<td>page 557</td>
</tr>
<tr>
<td>“Updating the length of an inline LOB column by importing DML”</td>
<td>page 558</td>
</tr>
</tbody>
</table>

### Analyzing and executing changes

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER”</td>
<td>page 560</td>
</tr>
<tr>
<td>“Executing an alter-type worklist”</td>
<td>page 562</td>
</tr>
<tr>
<td>“Generating a multi-step job”</td>
<td>page 564</td>
</tr>
</tbody>
</table>

---

**Creating an alter-type work ID**

You can create an alter-type work ID to alter data structures.
1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of the WORKID and select Create A NEW WORKID. Then, press Enter.

   The Create WORKID panel is displayed.

3 Specify the information for the work ID.

   A Select Alter for the Type.

   B (optional) Specify a Comment to describe the work ID.

   C Press Enter.

4 Press END.

   The WORKID Action Menu is displayed. A message displayed in the upper-right corner indicates that the work ID has been added.

**Where to go from here**

After you create an alter-type work ID, perform the tasks in the following table to specify changes to data structures, analyze a work ID, and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Specify DB2 definitions.</td>
<td>“Specifying changes to data structures” on page 383</td>
</tr>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Creating an alter-type work ID in batch mode**

ALTER and CHANGE MANAGER enable you to automate your batch job processes by providing the ability to create new work IDs in batch. This capability allows you to create work IDs by editing the JCL that ALTER or CHANGE MANAGER created outside of the products.
1 Import the source file using the batch run type, and create the JCL (page 379 or page 381).

2 In ISPF, edit the Import JCL. In the ALUIN input stream, change the WORKID keyword to NEWWORKID and specify the name of the new work ID that you want to create.

3 To import the file, submit the JCL (page 379 or page 381).

**Sorting work IDs**

You can sort the work IDs in a selection list by one or more of the following columns: owner, name, type, status, status date, and comments.

1 On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.

   The WORKID Action Menu is displayed.

2 Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **List** and press **Enter**.

   The WORKID Selection List is displayed.

3 On the **Command** line, type **SORT**.

4 On the Sort Specifications panel, specify the columns on which to sort a list of work IDs:

   A In the **Seq** field, specify sequential digits (starting with 1).

      Valid values for the field are 1 through 6 (if the **Stat Date** column is specified) or 1 through 8 (if the **Stat Date** column is not specified).

      **NOTE**

      If you specify a sequence for the **Stat Date** column, you cannot specify a sequence for the **Stat YYYY**, **Stat MM**, or **Stat DD** column.

   B In the **Asc/Desc** field, specify whether the data in the columns should be in ascending (A) order or descending (D) order.

      By default, the data is in ascending order.
Deleting a work ID

Press END.

The data is displayed in the WORKID Selection List in the specified sort sequence and order.

To sort by a single column

To sort the list of work IDs by a single column in ascending order, position your cursor on a column name (for example, Type), and press Enter.

Where to go from here

After you sort the work IDs, perform the actions in the following table to specify changes to data structures, analyze a work ID, and execute a worklist.

<table>
<thead>
<tr>
<th>Action</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the Act column, type S.</td>
<td>“Specifying changes to data structures” on page 383</td>
</tr>
<tr>
<td>In the Act column, type A.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>In the Act column, type X.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Deleting a work ID

To help control the size of the CM_WORKID and CM_SYNC tables, as well as other product CD tables, you can delete any unwanted work IDs. For information about controlling the size of sync tables, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1. You can also delete work IDs by listing the work IDs from the WORKID Selection List. To delete multiple work IDs, you can use a CM/PILOT script.

TIP

To delete multiple work IDs, see page 373.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Delete a WORKID and press Enter.

The WORKID Delete Confirmation panel is displayed.
Creating an inbound migrate profile in CHANGE MANAGER

You can use an inbound migrate profile in CHANGE MANAGER to modify the input that is imported from another system to match the receiving subsystem’s version of the application. An inbound migrate profile can contain change rules that modify imported data structure definitions.

1. On the CHANGE MANAGER Main Menu, select **Migrate Profiles**, and press **Enter**.

   The Migrate Profile Action Menu is displayed.

2. Type the name for the new **Migrate Profile** and select **Create a new Migrate Profile**. Then, press **Enter**.

   The Create Migrate Profile panel is displayed.

3. Specify the information for the migrate profile.

   A. Select **Inbound** for the **Type**.

   B. To specify change rules for the migrate profile, select **Change Rules**.

   **TIP**

   To create change rules for a migrate profile, see page 151.

   C. Press **Enter**.

4. Press **END**.

   The Migrate Profile Action Menu is displayed. A message displayed in the upper-right corner indicates that the profile has been added.

---

**TIP**

If you are deleting several work IDs, you can avoid the display of the WORKID Delete Confirmation panel for each work ID by selecting **Delete WORKID and Set Confirmation Off**. The WORKID Delete Confirmation panel will be displayed only for the first work ID.
Where to go from here

After you create an inbound migrate profile, perform the tasks in the following table to specify changes to data structures, analyze a work ID, and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Specify DB2 definitions.</td>
<td>“Specifying changes to data structures” on page 383</td>
</tr>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Replicating multiple work IDs

The CM/PILOT REPPLICATE_WORKID script contains the steps to replicate work IDs.

In this task, you will

1. create a task ID
2. create a DML statement
3. execute the task ID

To create a task ID

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3. Type the name of a new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

4. Select Replicate a WORKID, and press Enter.

   The Create TASKID panel is displayed.
5 Specify the information for a task ID.

A *(optional)* Specify a *Comment* to describe the task ID.

B Type S to select Create or Edit DML.

C Press Enter.

The WORKID Detail for Replicate DML panel is displayed.

**To create the DML statement**

1 For the REPLICATE WORKID DML statement, type a value for an attribute for the SET clause. Then, type an operator and a value for the WHERE clause. Press END.

---

**TIP**

To view all of the fields that can be used in the WHERE clause, type NEXT on the *Command* line after you have specified a value for the SET and the WHERE clauses.

---

The DML statement is displayed in the ISPF Editor.

2 Edit your DML statement to match that shown in Figure 85 and provide the information that is required for your site. For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**Figure 85  DML for replicating a work ID**

<table>
<thead>
<tr>
<th>REPLICATE WORKID</th>
<th>SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKOWNER</td>
<td>= 'RDACRJ'</td>
</tr>
<tr>
<td>WHERE</td>
<td></td>
</tr>
<tr>
<td>WKOWNER</td>
<td>= 'DEM'</td>
</tr>
</tbody>
</table>

3 When you have finished editing the statement, press END.

The List DML Statements panel is displayed.

4 Specify the information for the CHANGE DATABASE DML statement.

A Type A in the Act column adjacent to the REPLICATE WORKID statement and press Enter.

The Object Selection List for Replication Changes panel is displayed.
Replicating multiple work IDs

B Select **Database** for the **Object** and press **Enter**.

The Database Detail for Replicate DML panel is displayed.

C Type a value for an attribute for the SET clause. Then, type an operator and a value for the WHERE clause.

D Press **END**.

The DML statement is displayed in the ISPF Editor, as shown in **Figure 86**.

**Figure 86   DML for replicating a database**

```
CHANGE DATABASE
    SET CDNAME = 'RDAMKE',
    CDOWNER = 'RDACRJ',
    DBNAME = 'DEMOCJ',
    NEW_DBNAME = 'DEMOME'
WHERE
    CDOWNER = 'DEM'
AND CDNAME LIKE 'RDAE%'
AND DBNAME = 'AAX100'
AND NEW_DBNAME LIKE 'AAMX%'
AND NEW_BPOOL = 'BP1'
AND CHANGELVL = 'L'
AND DBOWNER LIKE 'DEMOC%'
;
```

5 When you have finished modifying the DML statements, press **END** until the TASKID Action Menu is displayed.

The task ID containing the DML statements for the script is created.

**To execute the task ID**

1 Select **Execute a TASKID** and press **Enter**.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT **Worklist**.

B Type the data set name for the CM/PILOT **JCL**.

C Type the data set name for **Diagnostics** or type **SYSOUT**.
3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

--- WARNING ---

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select Step 1 Replicate WORKID and press Enter.

The TASKID Replicate WORKID Step Options panel is displayed.

C Select whether to replace any existing work IDs. Then, press Enter.

--- WARNING ---

Use caution before selecting the option to replace existing work IDs.

The Step Settings panel is displayed.

D Press END.

The TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.
Modifying a work ID with values in a user-defined table

C To review or modify the JCL, type S to select **Edit JCL**.

D To submit the JCL to run the CM/PILOT worklist, type S to select **Submit JCL**.

E Press **Enter**.

CM/PILOT replicates the work IDs in the database.

---

**NOTE**

An Analysis worklist is not created and is not required.

---

**Modifying a work ID with values in a user-defined table**

You can use the CM/PILOT REPLICATE_WORKID script to create a task ID that replicates a work ID which applies changes that are specified from a user-defined table.

---

**TIP**

To replicate multiple work IDs, see page 362.

---

In this task, you will

1. create a task ID
2. create a DML statement
3. execute the task ID

**To create a task ID**

1. On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.
   
   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press **Enter**.
   
   The TASKID Action Menu is displayed.
3 Type the name of a new **TASKID** and select **Create a TASKID**. Then, press **Enter**.

The Script Selection List is displayed.

4 Select **Replicate a WORKID**, and press **Enter**.

The Create TASKID panel is displayed.

5 Specify the information for a task ID.

   A *(optional)* Specify a **Comment** to describe the task ID.

   B Type **S** to select **Create or Edit DML**.

   C Press **Enter**.

The WORKID Detail for Replicate DML panel is displayed.

**To create the DML statement**

1 Type a value for an attribute for the SET clause. Then, type an operator and a value for the WHERE clause. Press **END**.

The DML is displayed in the ISPF Editor.

2 Edit your DML to match that shown in **Figure 87** and provide the information that is unique to your site.

For information about DML syntax, see the **ALTER and CHANGE MANAGER for DB2 Reference Manual**.

**Figure 87  DML for applying changes from a user-defined table (part 1 of 2)**

<table>
<thead>
<tr>
<th>DML for applying changes from a user-defined table (part 1 of 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REPLICATE  WORKID</strong></td>
</tr>
<tr>
<td><strong>SET</strong> <strong>WKOWNER</strong> = 'RDADEM'</td>
</tr>
<tr>
<td><strong>WHERE</strong> <strong>WKOWNER</strong> = 'DEM'</td>
</tr>
<tr>
<td><strong>AND</strong> <strong>WKNAME</strong> = 'WRK1620'</td>
</tr>
<tr>
<td><strong>;</strong></td>
</tr>
<tr>
<td><strong>CHANGE TABLEPART</strong></td>
</tr>
<tr>
<td><strong>JOIN</strong> <strong>DEM.JOINPARTS</strong></td>
</tr>
<tr>
<td><strong>SET</strong> <strong>NEW_PQTY</strong> = <strong>PRI_QTY</strong></td>
</tr>
<tr>
<td><strong>NEW_SQTY</strong> = <strong>SEC_QTY</strong></td>
</tr>
<tr>
<td><strong>WHERE</strong> <strong>(DBNAME</strong> = <strong>QUALIFIER</strong> AND <strong>TSNAME</strong> = <strong>NAME</strong>)**</td>
</tr>
</tbody>
</table>
Modifying a work ID with values in a user-defined table

Figure 87  DML for applying changes from a user-defined table (part 2 of 2)

<table>
<thead>
<tr>
<th>IXNAME = NAME</th>
</tr>
</thead>
</table>

The DML in Figure 87 replicates the old work ID. Any table partitions that match the user-defined table are modified. In this example, DEMJOINPARTS is a table that is used to set the primary and secondary quantities of specific table and index partitions. Setting these quantities can be useful in migrating from a development to a production environment.

NOTE

Information about the index parts is stored in the table partition CD table. You should make changes to the index partitions and table partitions with one DML statement.

3 When you have finished editing the statements, press END until the TASKID Action Menu is displayed.

The task ID containing the DML statements for the script is created.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

   A Type the data set name for the CM/PILOT Worklist.

   B Type the data set name for the CM/PILOT JCL.

   C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

   For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.
Modifying a work ID with values in a user-defined table

---

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

---

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

   The Step Settings panel is displayed.

B Type S to select Step 1 Replicate WORKID and press Enter.

   The TASKID Replicate WORKID Step Options panel is displayed.

C Select whether to replace any existing work IDs. Then, press Enter.

---

**WARNING**

Use caution before selecting the option to replace existing work IDs.

---

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

   The Step Settings panel is displayed.

B Type S to select Step 1 Replicate WORKID and press Enter.

   The TASKID Replicate WORKID Step Options panel is displayed.

C Select whether to replace any existing work IDs. Then, press Enter.

---

**WARNING**

Use caution before selecting the option to replace existing work IDs.

---

The Step Settings panel is displayed.

D Press END.

   The TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

   CM/PILOT applies the changes.
Changing column definitions for imported DDL

You can use the CM/PILOT REPLICATE_WORKID script to create a task ID that sets column definitions for imported DDL to NOT NULL. This task ID is useful when you use imported DDL that is created by a data modeling tool that automatically sets column definitions to NULL.

To replicate multiple work IDs, see page 362.

In this task, you will

1. create a task ID
2. create a DML statement
3. execute the task ID

To create a task ID

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.
2 Select TASKIDs and press Enter.
   The TASKID Action Menu is displayed.
3 Type the name of a new TASKID and select Create a TASKID. Then, press Enter.
   The Script Selection List is displayed.
4 Select Replicate a WORKID, and press Enter.
   The Create TASKID panel is displayed.
5 Specify the information for a task ID.

A *(optional)* Specify a Comment to describe the task ID.

B Type S to select Create or Edit DML.

C Press Enter.

The WORKID Detail for Replicate DML panel is displayed.

**To create the DML statement**

1 Type a value for an attribute for the SET clause. Then, type an operator and a value for the WHERE clause. Press END.

The DML statement is displayed in the ISPF Editor.

2 Edit your DML to match that shown in Figure 88 and provide the information that is unique to your site.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**Figure 88  DML for changing column definitions**

```
REPPLICATE WORKID
  SET WKNAME = 'NEWDDL2'
  WHERE
    WKNAME = 'NEWDDL1'
;
CHANGE COLUMN
  SET NEW_NULLI = 'N'
  WHERE
    CD_NAME = 'NEWDDL1'
;
```

The DML in Figure 88 replicates the old work ID. All of the columns for the work ID are changed to NOT NULL with the CHANGE COLUMN statement. You can write the DML statement to select specific columns to change to NOT NULL.

For a list of the CD table column attributes that can be modified, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

3 When you have finished editing the statements, press END until the TASKID action Menu is displayed.

The task ID containing the DML statements for the script is created.
To execute the task ID

1. Select **Execute a TASKID** and press **Enter**.

   The TASKID Interface panel is displayed.

2. On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.
   
   **A** Type the data set name for the CM/PILOT **Worklist**.
   
   **B** Type the data set name for the CM/PILOT **JCL**.
   
   **C** Type the data set name for **Diagnostics** or type SYSOUT.

3. Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4. **(optional)** Override the options that are set in CHANGE MANAGER.

   For more information about the CHANGE MANAGER options, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*.

   **WARNING**

   BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

5. On the TASKID Interface panel, create a CM/PILOT worklist.

   **A** Type **S** to select **Create Worklist**. Then, press **Enter**.

   The Step Settings panel is displayed.

   **B** Type **S** to select **Step 1 Replicate WORKID** and press **Enter**.

   The TASKID Replicate WORKID Step Options panel is displayed.

   **C** Select whether to replace any existing work IDs. Then, press **Enter**.
Deleting multiple work IDs

WARNING
Use caution before selecting the option to replace existing work IDs.

The Step Settings panel is displayed.

D Press END.

The TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT sets the column definitions for DDL to NOT NULL.

NOTE
An Analysis worklist is not created and is not required.

Deleting multiple work IDs

The CM/PILOT DELETE_WORKID script contains the steps to delete multiple work IDs. To help control the size of the CM_WORKID and CM_SYNC tables, as well as other product CD tables, you can delete any unwanted work IDs. For information about controlling the size of sync tables, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

In this task, you will

1. create a task ID

2. create a DML statement
3. execute the task ID

**To create a task ID**

1. On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.
   
   The CM/PILOT Main Menu is displayed.

2. Select **TASKIDs** and press **Enter**.
   
   The TASKID Action Menu is displayed.

3. Type the name of a new **TASKID** and select **Create a TASKID**. Then, press **Enter**.
   
   The Script Selection List is displayed.

4. Select **Delete one or more WORKIDs** and press **Enter**.
   
   The Create TASKID panel is displayed.

5. Specify the information for a task ID.

   A. *(optional)* Specify a **Comment** to describe the task ID.

   B. Type **S** to select **Create or Edit DML**.

   C. Press **Enter**.

   The WORKID Detail for Delete WORKID DML panel is displayed.

**To create the DML statement**

1. Type an operator and a value for the WHERE clause. Then, press **END**.

   **TIP**
   
   To view all of the fields that can be used in the WHERE clause, type **NEXT** on the **Command** line after you have specified a value for the WHERE clause.

   The DML statement is displayed in the ISPF Editor.

2. Edit your DML statement in the ISPF Editor as shown in **Figure 89**, if necessary.

   For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*. 
Deleting multiple work IDs

Chapter 4 Altering data structures in a database environment

3 When you have finished editing the statement, press END until the TASKID Action Menu is displayed.

The task ID containing the DML statement for the script is created.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT Worklist.

B Type the data set name for the CM/PILOT JCL.

C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 On the TASKID Interface panel, create a CM/PILOT worklist and select your options for processing the worklist.

A Type S to select Create Worklist. Then, press Enter.

B To edit the CM/PILOT worklist, type S to select Edit Worklist.

C To create the JCL to run the worklist, type S to select Create JCL.

NOTE
You can copy the DML statement from member ACMDMLD2 in the HLQ.DBCNTL data set.

Figure 89 DML for deleting work IDs

DELETE WORKID
WHERE
  (WKOWNER LIKE 'ABC%' OR
   WKOWNER LIKE 'DEF%')
AND    DATE(MODDATE) < DATE('2009-11-01')
;

3 When you have finished editing the statement, press END until the TASKID Action Menu is displayed.

The task ID containing the DML statement for the script is created.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT Worklist.

B Type the data set name for the CM/PILOT JCL.

C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 On the TASKID Interface panel, create a CM/PILOT worklist and select your options for processing the worklist.

A Type S to select Create Worklist. Then, press Enter.

B To edit the CM/PILOT worklist, type S to select Edit Worklist.

C To create the JCL to run the worklist, type S to select Create JCL.
Deleting sync table entries for multiple work IDs

The CM/PILOT DELETE_WORKID script can be modified to contain the steps to delete the sync table entries for one or more work IDs. Only the sync table entries are deleted; all of the other entries in the CHANGE MANAGER CD tables that are associated with the original work ID are not deleted. For information about controlling the size of sync tables, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.*

In this task, you will

1. create a task ID
2. create a DML statement
3. execute the task ID

**To create a task ID**

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select TASKIDS and press Enter.

   The TASKID Action Menu is displayed.

3. Type the name of a new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

To review or modify the JCL, type S to select Edit JCL.

To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

Press Enter.

CM/PILOT deletes the work IDs.

**NOTE**

An Analysis worklist is not created and is not required.
Deleting sync table entries for multiple work IDs

4 Select **Delete one or more WORKIDs** and press Enter.

The Create TASKID panel is displayed.

5 Specify the information for a task ID.

A *(optional)* Specify a **Comment** to describe the task ID.

B Type S to select **Create or Edit DML**.

C Press Enter.

The WORKID Detail for Delete WORKID DML panel is displayed.

**To create the DML statement**

1 Type an operator and a value for the WHERE clause. Then, press END.

---

**TIP**

To view all of the fields that can be used in the WHERE clause, type NEXT on the Command line after you have specified a value for the WHERE clause.

---

The DML statement is displayed in the ISPF Editor.

2 Edit your DML statement in the ISPF Editor, as shown in Figure 90, if necessary.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

---

**NOTE**

You can copy the DML statement from member ACMDMLD3 in the HLQ.DBCNTL data set.

---

**Figure 90  DML for deleting sync table entries**

```
DELETE SYNTABLE
WHERE
    STATUS = 'C'             -- LIMITS TO STATUS COMPLETE
    AND (WKOWNER LIKE 'ABC%' OR
        WKOWNER LIKE 'DEF%')
    AND DATE(MODDATE) < DATE('2009-11-01')
;
```
Delet ing sync table entries for multiple work IDs

**NOTE**
Ensure that you change the DML statement from DELETE WORKID to DELETE SYNCTABLE.
The WHERE clause includes CD table column attributes from the CM_WORKID table. These attributes are used to delete the rows from the CM_SYNC table.

3 When you have finished editing the statement, press END until the TASKID Action Menu is displayed.

The task ID containing the DML statement for the script is created.

**To execute the task ID**

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

   A Type the data set name for the CM/PILOT Worklist.

   B Type the data set name for the CM/PILOT JCL.

   C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 On the TASKID Interface panel, create a CM/PILOT worklist and select your options for processing the worklist.

   A Type S to select Create Worklist. Then, press Enter.

   B To edit the CM/PILOT worklist, type S to select Edit Worklist.

   C To create the JCL to run the worklist, type S to select Create JCL.

   D To review or modify the JCL, type S to select Edit JCL.

   E To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

   F Press Enter.

   CM/PILOT deletes the sync table entries.
You can import a CDL, DDL, or DML file in the CHANGE MANAGER product. For DDL and CDL files, you can use an inbound migrate profile to specify change rules for the import.

**TIP**
To create an alter-type work ID, see page 357. To create an inbound migrate profile, see page 361.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID and select Import CDL, DDL or DML to a WORKID. Then, press Enter.

   The Import Interface panel is displayed.

3. Specify the information to import the file.
   
   A. *(optional for CDL and DDL files)* Type the name of an inbound migrate profile, or type a wildcard pattern to display a list of migrate profiles.

   B. Select the Source Type.

   C. Select the Run Type.

   D. Press Enter.

   The Import JCL Processing Interface panel is displayed.

4. Specify the data set names to be used by Import.

   A. *(batch run type)* Type the data set name for the JCL that is generated by this process.

   B. Type the data set name for the CDL, DDL, or DML file that you want to import.

---

**NOTE**
An Analysis worklist is not created and is not required.
C Type one of the following options for Diagnostics:

- (foreground run type) To display the diagnostics on the terminal, type TERM.
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type SYSOUT.

D (foreground run type) If you specified SYSOUT in step 4C, specify the Sysout Class.

5 Select the options for importing the source file.

A To review or modify the source file before it is stored as entries in the CD tables, type S to select Edit CDL, Edit DDL, or Edit DML.

B To create the input, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

C To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

D To import the file, select one of the following options:

- (foreground run type) Type S to select Run.
- (batch run type) Type S to select Submit JCL.

E Press Enter.

6 Press END.

The WORKID Action Menu is displayed.

**Where to go from here**

After you import a CDL, DDL, or DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Importing a DDL file in ALTER

You can import a DDL file in the ALTER product for an existing alter-type work ID.

_TIP_ To create an alter-type work ID, see page 357.

1. On the ALTER Main Menu, select WORKID, and press Enter.
   The WORKID Action Menu is displayed.

2. Type the name of the WORKID and select Import DDL to a WORKID. Then, press Enter.
   The Import Interface panel is displayed.

3. Select the Run Type and press Enter.
   The Import JCL Processing Interface panel is displayed.

4. Specify the data set names to be used by Import.

   A (batch run type) Type the data set name for the JCL that is generated by this process.

   B Type the data set name for the DDL file that you want to import.

   C Type one of the following options for Diagnostics:
     - (foreground run type) To display the diagnostics on the terminal, type TERM.
     - To write the diagnostics to a sequential file, type the name of the data set.
     - To write the diagnostics to a print data set, type SYSOUT.

   D (foreground run type) If you specified SYSOUT in step 4C, specify the Sysout Class.
5 Select your options for importing the DDL file.

A To review or modify the DDL before it is stored as entries in the CD tables, type S to select Edit DDL.

B To create the input, select one of the following options:

- (foreground run type) Type S to select Create Input.
- (batch run type) Type S to select Create JCL.

C To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Input.
- (batch run type) Type S to select Edit JCL.

D To import the file, select one of the following options:

- (foreground run type) Type S to select Run.
- (batch run type) Type S to select Submit JCL.

E Press Enter.

6 Press END.

The WORKID Action Menu is displayed.

Where to go from here

After you import a DDL file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Specifying changes to data structures

You can specify changes to data structures in an existing alter-type work ID.

--- Tip ---
To create an alter-type work ID, see page 357.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name (or use wildcard characters) of any data structures to be included in a list. For more information about specifying data structures, see “Specifying the data structures for a migration” on page 41.

4 Press Enter.

The Mixed List panel is displayed.

5 Type the letter of an action in the Act column to create, edit, or drop a data structure. For a description of the action codes, see “Specifying the data structures for a migration” on page 41 or press F1.

--- Tip ---
You can perform several tasks from the Mixed List panel, including adding a column to a table. For an example of adding a column to a table, see “Adding a column to a table” on page 402.

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

6 Press END.

The Object Specification panel is displayed.

7 To view a list of the data structures that have changed, type S to select Changed Objects List.

The Changed Objects List is displayed.
Creating a primary constraint for a table

8 From the Changed Objects List panel, you can type the letter of an action in the Act column to change a data structure. Then, press END.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

The Object Specification panel is displayed.

9 Press END.

The WORKID Action Menu is displayed.

**Where to go from here**

After you import specify changes to the data structures, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Creating a primary constraint for a table**

You can use an existing index to create a new primary constraint for a table that does not have a primary key. A table can have a primary constraint and none, one, or more unique (nonprimary) constraints.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the existing WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name of a database, table space, table, and index (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.
4 Type CUC in the Act column adjacent to the index (IX) that will enforce the primary constraint.

The Create Constraint panel is displayed.

5 Specify the information about the constraint.

A *(optional)* Type the name of the constraint.

B Type P for the **Constraint Type** to create a primary constraint.

C Press Enter.

D Press END.

The Mixed List panel is displayed, as shown in Figure 91.

**Figure 91  Mixed List panel**
Creating a unique constraint for a table

The new primary constraint, designated by the UCP object type, is displayed below the table object.

**NOTE**

After you analyze the work ID and execute the worklist, the UCP object type will be displayed below the index that enforces the constraint you created. The primary constraint will contain the same columns as the index.

**6** Press END until the WORKID Action Menu is displayed.

**Where to go from here**

After you specify to create a primary constraint, perform the tasks in the following table to analyze the work ID and execute the worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Creating a unique constraint for a table**

You can create a new unique (nonprimary) constraint for a table. A table might already have a primary key and none, one, or more unique (nonprimary) constraints.

**1** On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

**2** Type the name of the existing WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

**3** Specify the name of a database, table space, and table (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

**4** Type CUC in the Act column adjacent to the table (TB) for which you want to create a unique (nonprimary) constraint.

The Create Constraint panel is displayed.
5 Specify the information about the constraint.

A (optional) Type the name of the constraint.

**TIP**
Some object names might be too long to be displayed on a panel. To edit a long object name, position the cursor on the object name and press the ZOOM (F4) key.

B Type U for the Constraint Type to create a unique constraint.

C To specify the columns in the constraint, type S to select Constraint Key Columns.

D Press Enter.

The Constraint Key Mixed List panel is displayed, as shown in Figure 92.

**Figure 92  Constraint Key Mixed List panel**

<table>
<thead>
<tr>
<th>ACT</th>
<th>COLNAME</th>
<th>COLUMN TYPE</th>
<th>UDT LENGTH</th>
<th>NL</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>CONSTRAINT KEY .:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARENT TABLE .: DEMOCJ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0110_REFERRAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>REFERRAL_NUM</td>
<td>DECIMAL</td>
<td>10</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>AREA_NUM</td>
<td>SMALLINT</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CASE_NUM</td>
<td>DECIMAL</td>
<td>10</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>CREATION_DT</td>
<td>DATE</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DISCOVERY_DT</td>
<td>DATE</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MAX_HIST_SEQ_NUM</td>
<td>SMALLINT</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CASE_POOL_NUM</td>
<td>SMALLINT</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>
6 Specify the columns in the constraint.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Type C in the Act column adjacent to the column name that you want to use.

**NOTE**

To select more than one name, type CC in the Act column adjacent to the first column name and CC in the Act column adjacent to the last column name.

B Type A in the Act column adjacent to the Constraint Key.

C Press Enter.

The Constraint Key Mixed List panel is displayed as shown in Figure 93. The column specified for the constraint is displayed below the constraint key.

**Figure 93  New constraint key column**

```
ALUSUC2 ER  Constraint Key Mixed list
Command  =>
WORKID . . . : RDACRJ.UC01
Table . . . : DEMOCJ.T0110_REFERRAL
Const Name . : UKEY2
(C)opy column from the Table, indicate position in Key Columns using (A)fter/(B)efore, and press ENTER.
LINES 1 TO 7 OF 15
More: +
Act Colname          Column Type      UDT Length N1 FP
TOP
Constraint Key .:
L 1 REFERRAL_NUM
PARENT TABLE . . : DEMOCJ T0110_REFERRAL
  REFERRAL_NUM    DECIMAL           10 N N
  AREA_NUM        SMALLINT          N N
  CASE_NUM        DECIMAL           10 N N
  CREATION_DT     DATE              N N
  DISCOVERY_DT    DATE              N N
  MAX_HIST_SEQ_NUM SMALLINT         N N
```

7 Press END twice.

The Mixed List panel is displayed, as shown in Figure 94.
Creating a unique constraint for a table

### Figure 94  Mixed List panel

![Image of Mixed List panel]

The new unique constraint, designated by the UCU object type, is displayed below the table object.

8 Press END until the WORKID Action Menu is displayed.

9 Create or update a unique index for this unique constraint.

**Where to go from here**

After you specify to create a unique constraint, perform the tasks in the following table to analyze the work ID and execute the worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Editing the columns for a unique constraint

You can add or delete the columns for an existing primary or unique constraint.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the existing WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a database, table space, table, and constraint (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Type CO in the Act column adjacent to the unique constraint (UCU) or primary constraint (UCP) that you want to modify.

   The Constraint Key Mixed List panel is displayed, as shown in Figure 95.

Figure 95  Constraint Key Mixed List panel

<table>
<thead>
<tr>
<th>Act</th>
<th>Colname</th>
<th>Column Type</th>
<th>UDT Length</th>
<th>Nl</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>CONSTRAINT KEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>COLUMN_15</td>
<td>INTEGER</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>PARENT TABLE</td>
<td>ACMX01</td>
<td>T_X01PS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 Specify the modifications to the columns in the constraint.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A To add a column, type C in the Act column adjacent to the column name or names that you want to use. Then, type A in the Act column adjacent to the Constraint Key or to the column after which the new column should appear.

**NOTE**

To select more than one name, type CC in the Act column adjacent to the first column name and CC in the Act column adjacent to the last column name.

B To delete a column, type D in the Act column adjacent to the column name below the Constraint Key.

C Press Enter.

The Constraint Key Mixed List panel is displayed as shown in Figure 96. The column added for the constraint is displayed below the constraint key.

**Figure 96   Edited constraint key columns**

<p>| ALUSUC2 ER | ----------------------------------- Constraint Key Mixed list ----------------------------------- | Command ===&gt; Scroll.. PAGE |
| WORKID . . : RDACTRL.UC01 Commands: CANCEL |
| Table . . : ACMX01.T_X01PS |
| Const Name . : COLUMN_15 |
| (C)opy column from the Table, indicate position in Key Columns using (A)fter/(B)efore, and press ENTER. Lines 1 to 7 of 25 More: + |</p>
<table>
<thead>
<tr>
<th>Act</th>
<th>Colname</th>
<th>Column Type</th>
<th>UDT Length</th>
<th>NL</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTRAINT KEY .:</td>
<td>TQP</td>
<td>**********************************************</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>COLUMN_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>COLUMN_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>COLUMN_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>COLUMN_15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARENT TABLE . : ACMX01.T_X01PS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN_1</td>
<td>INTEGER</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN_2</td>
<td>SMALLINT</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN_3</td>
<td>CHAR</td>
<td>12</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>
6 Press END until the WORKID Action Menu is displayed.

7 Update the associated index columns.

8 Update the associated foreign key columns of the child table.

**NOTE**
The PKEYPROP keyword in the ALUIN input stream will propagate changes to the column data type or length.

**Where to go from here**

After you specify to modify a unique constraint, perform the tasks in the following table to analyze the work ID and execute the worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Editing a primary key**

You can add or delete columns in an existing primary key. You can also drop a primary key from a table that existed before the subsystem was upgraded with the IBM CATMAINT utility. (A UCP object type will not be displayed in the Mixed List for this type of table.)

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of the existing WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3 Specify the name of a database, table space, and table (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.
4 Type **PK** in the **Act** column adjacent to the table (TB) that you want to modify. A primary key is created if one does not exist.

The Constraint Key Mixed List panel is displayed, as shown in Figure 97.

**Figure 97  Constraint Key Mixed List panel**

---

5 Specify the modifications to the columns in the constraint.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM** (F4) key.

---

A To add a column, type **C** in the **Act** column adjacent to the column name that you want to use. Then, type **A** in the **Act** column adjacent to the Constraint Key.

---

**NOTE**

To select more than one name, type **CC** in the **Act** column adjacent to the first column name and **CC** in the **Act** column adjacent to the last column name.
Editing a primary key

To delete a column, type D in the Act column adjacent to the column name below the Constraint Key.

NOTE
To drop the primary key from a table that has been created by using the IBM CATMAINT utility to migrate to a later version of DB2, type D in the Act column adjacent to every column.

Press Enter.

The Constraint Key Mixed List panel is displayed as shown in Figure 98. The column added for the constraint is displayed below the constraint key.

Figure 98  Constraint Key Mixed List panel

6 Press END until the WORKID Action Menu is displayed.

7 Update the associated index columns.

8 Update the associated foreign key columns of the child table.

NOTE
The PKEYPROP keyword in the ALUIN input stream will propagate changes to the column data type or length.
Naming a primary key

You can name an existing primary key in a table that existed before the subsystem was upgraded with the IBM CATMAINT utility.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the existing WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a database, table space, and table (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Type PK in the Act column adjacent to the table (TB) that you want to modify. Then, press Enter.

   The Constraint Key Mixed List panel is displayed.

5. Press END.

   The Mixed List panel is displayed, as shown in Figure 99.

Where to go from here

After you specify to modify a primary key, perform the tasks in the following table to analyze the work ID and execute the worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Naming a primary key
An "*A" is displayed next to the primary constraint, which is designated by the UCP object type. The name of the constraint is "<DEFLT>".

6 Type **E** in the Act column adjacent to the primary constraint (UCP) that you defined in the previous step. Then, press **Enter**.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.

---

The Constraint Detail panel is displayed.

7 Type the new value for the Constraint Name. Then, press **END**.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.

---

The Mixed List panel is displayed.
Changing the attributes of an index

You can change several of the attributes of an index. This task provides instructions for changing the uniqueness, clustering, padding, and key sequence attributes of an index and the key columns in the index.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name (or use wildcard characters) of any data structures to be included in a list. Then, press Enter.

   The Mixed List panel is displayed.

### NOTE

The name of the constraint will always be displayed as “<DEFLT>” for this work ID.

8. Press END until the WORKID Action Menu is displayed.

### Where to go from here

After you specify to name a primary key, perform the tasks in the following table to analyze the work ID and execute the worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

### Chapter 4  Altering data structures in a database environment  397
4 Type E in the Act column adjacent to the index that you want to modify.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

The Index Detail panel is displayed.

5 Specify the uniqueness and the clustering of the index.

A For **Unique**, type U (unique), D (duplicates), or W (where not null).

B For **Cluster**, type Y or N.

C For **Padded**, type Y or N.

D Type S to select **Keys and TB Cols**.

E Press **Enter**.

The Index Key Mixed List panel is displayed, as shown in **Figure 100**.

**Figure 100  Index Key Mixed List panel**

```
ALUSIXKXR --------- Index Key Mixed List ---------
Command ===> Scroll. . PAGE
WORKID . . . : RDACRJ.CHGCOL Commands: CANCEL END
(C)opy columns from the list under TABLE COLUMNS up to make a list under INDEX KEY. Use (A)fter/(B)efore to position columns under INDEX KEY.  
Objects 1 to 10 of 23 More:
Act      Colname      Column Type UDT Length N1 FP
******* TOP ***************
A INDEX KEY . . . : DEMOCJ IXS23A
TABLE COLUMNS . : DEMOCJ TBS23A
C C1 CHAR 3 N N
TS23A_C4_1 CHAR 4 N N
TS23A_C4_2 CHAR 4 N N
TS23A_C6_3 CHAR 6 N N
TS23A_C4_4 CHAR 4 N N
TS23A_VC10_1 VARCHAR 10 N N
TS23A_SI_1 SMALLINT N N
TS23A_I_1 INTEGER N N
TS23A_FS_1 FLOAT 21 N N
TS23A_FD_1 FLOAT 53 N N
```
6 Specify the changes to the index key columns.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

- To add a column, type C in the Act column adjacent to the column name that you want to use. Then, type A in the Act column adjacent to the Index Key. Press Enter.

**NOTE**
To select more than one name, type CC in the Act column adjacent to the first column name and CC in the Act column adjacent to the last column name.

- To delete a column, type D in the Act column adjacent to the column name below the Index Key. Press Enter.

The Index Key Mixed List panel, which is shown in Figure 101, displays the index key column that has been added.

**Figure 101  Index Key Mixed List panel—column added**

7 Press END.

The Index Detail panel is displayed.
8 Type S to select Keys.

The Index Keys List panel is displayed.

9 To change the key sequence, type A (ascending), D (descending), I (include) or R (random) in the Asc/Desc/Incl/Rand column. Then, press Enter.

10 Press END.

The Index Detail panel is displayed.

11 Press END.

The Mixed List panel is displayed. An asterisk (*) and a change-level indicator are displayed adjacent to the object that you changed.

12 Press END until the WORKID Action Menu is displayed.

Where to go from here

After you modify the attributes of an index, perform the tasks in the following table to analyze the work ID and execute the worklist. The worklist contains the commands for unloading, dropping, and creating the table, and loading the remaining columns into their adjusted positions.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating indexes for ROWID GENERATED ALWAYS columns by importing DML

Columns that are defined as ROWID GENERATED ALWAYS might not have a unique index defined on the column. For example, if you add a LOB column to a table but do not add a ROWID column, DB2 creates the ROWID column on the table as GENERATED ALWAYS. DB2 does not create an index. If you want to use the BMC LOADPLUS utility to load data from the table, you must create a unique index for the column. You can import DML statements to create these unique indexes.

Before you import DML to create unique indexes, you must create a view that identifies all of the columns that have data type of ROWID GENERATED ALWAYS and do not have a unique index.
In this task, you will

1. create a view by executing DDL
2. create indexes and key columns by importing a DML file

---

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

---

**To create a view**

1. In a TSO session, open the SPUFI (SQL processor using file input) facility.
2. Type the DDL statement to match the one shown in Figure 102, and provide TBCREATOR information.

---

**NOTE**
You can copy the DDL statement from member ACMDMLU8 in the HLQ.DBCNTL data set.

---

**Figure 102**  DDL to create a view

```sql
CREATE  VIEW DEM.NOINDEX_FOR_ROWID (CREATOR, NAME, COLUMN) AS
SELECT ALL
    COLS.TBCREATOR,  
    COLS.TBNAME,  
    COLS.NAME
FROM SYSIBM.SYSCOLUMNS COLS
WHERE COLS.DEFAULT = 'A'
    AND TBCREATOR LIKE 'DEM922%'  -- change or remove
    AND NOT EXISTS (SELECT * FROM
    SYSIBM.SYSINDEXES IXS,
    SYSIBM.SYSKEYS KYS
    WHERE
        COLS.TBCREATOR = IXS.TBCREATOR
        AND COLS.TBNAME = IXS.TBNAME
        AND COLS.NAME = KYS.COLNAME
        AND KYS.IXCREATOR = IXS.CREATOR
        AND KYS.IXNAME = IXS.NAME
        AND IXS.UNIQUERULE = 'U'
        AND IXS.COLCOUNT = 1 )
    AND EXISTS (SELECT * FROM SYSIBM.SYSAUXRELS AUX
    WHERE  AUX.TBOWNER = COLS.TBCREATOR AND
    AUX.TBNAME = COLS.TBNAME)
;
```
3 Execute the DDL statement.

**To create the indexes and key columns**

Import the DML statements in member ACMDMLU8 in the `HLQ.DBCNTL` data set. The statements reference the view that was created and a unique index in which the only key column is a ROWID column. These statements create the indexes and the key columns.

For information about importing DML statements, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

---

**Adding a column to a table**

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3 Specify the name (or use wildcard characters) of any data structures to be included in a list. Then, press Enter.

   The Mixed List panel is displayed.

4 Type E in the Act column adjacent to the table that you want to modify.

   The Table Detail panel is displayed.

5 Type S to select Table Column List.

   The Table Columns List is displayed.

6 Type I in the Act column adjacent to the column name where you want to insert a column and press Enter.

   If you want to insert a line at the start of the column list, type the I in the Act column on the TOP line or the line that contains three asterisks (**`). A blank line is inserted into the list of columns.
Adding a column to a table

7 Type the **Column Name**, **Column Type**, **Length**, and attribute values. You can scroll to the right to enter additional column information or you can type E in the **Act** column to view a more detailed panel that displays the old and new values. Then, press END.

8 Press END.

The Mixed List panel is displayed. An asterisk (*) and a change-level indicator are displayed adjacent to the object that you changed.

9 Press END.

The Object Specification panel is displayed. You can make additional changes, if necessary.

10 Press END.

The WORKID Action Menu is displayed.

**Where to go from here**

After you add a column to a table, perform the tasks in the following table to analyze a work ID and execute a worklist. The worklist contains the commands for unloading, dropping, and creating the table, and loading the remaining columns into their adjusted positions.

---

**NOTE**
You can also type L in the **Act** column to duplicate a column definition. The Like action code creates an identical column listing immediately below the original. You can then edit the name of the column to create a new column.

---

**WARNING**
When you duplicate a column, ensure that you edit the column below the one that contains the L prefix notation. If you edit the original column name, Analysis creates a -RNAM command for the original, renames the original column, and creates a new column with the original name in the CREATE TABLE SQL statement. This action could result in unwanted data, data structure changes, or both.

---

**TIP**
Some object names might be too long to be displayed on a panel. To enter a long object name, position the cursor on the object name and press the ZOOM (F4) key.
Dropping a column from a table

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name (or use wildcard characters) of any data structures to be included in a list. Then, press Enter.

   The Mixed List panel is displayed.

4. Type CO in the Act column adjacent to the table that you want to modify.

   The Table Columns List is displayed.

5. Type D in the Act column adjacent to the name of the column that you want to remove.

6. Press END.

   The Mixed List panel is displayed. An asterisk (*) and a change-level indicator are displayed adjacent to the object that you changed.

7. Press END until the WORKID Action Menu is displayed.

Where to go from here

After you remove a column from a table, perform the tasks in the following table to analyze a work ID and execute a worklist. The worklist contains the commands for unloading, dropping, and creating the table, and loading the remaining columns into their adjusted positions.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Changing the data type and length of a column

You can change the data type of a column and the length of a column. You can also change any of the column attributes for existing data, provided that the data can be converted to the new definition.

To change the data type and length

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name (or use wildcard characters) of any data structures to be included in a list. Then, press Enter.

   The Mixed List panel is displayed.

4. Type CO in the Act column adjacent to the table that you want to modify.

   The Table Columns List is displayed.

5. Modify the value of the Column Type column for the column that you want to change.

   **TIP**

   Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

6. Modify the value of the Length column for the column that you want to change.
Moving multiple tables using Quick Edit

7 Press END.

The Mixed List panel is displayed. An asterisk (*) and a change-level indicator are displayed adjacent to the object that you changed.

8 Press END until the WORKID Action Menu is displayed.

Where to go from here

After you change the data type of a column, perform the tasks in the following table to analyze a work ID and execute a worklist. The worklist contains the commands for unloading, dropping, and creating the table, and loading the remaining columns into their adjusted positions.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Moving multiple tables using Quick Edit

You can use the Quick Edit feature in the Mixed List panel to move multiple tables quickly and easily into a new table space. This feature is useful for database administrators (DBAs) who implement and manage ERP applications.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name (or use wildcard characters) of any data structures to be included in a list. Then, press Enter.

The Mixed List panel is displayed.

4 Type QE in the Act column adjacent to the tables that you want to move. Then, type USE in the Act column adjacent to the nonpartitioned table space to which the tables are moving and press Enter.
The Table Quick Edit Confirmation dialog is displayed, as shown in Figure 103.

**Figure 103 Table Quick Edit Confirmation dialog**

```
<table>
<thead>
<tr>
<th>ALUSMXLP</th>
<th>Table Quick Edit Confirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command ===&gt;</td>
<td>Scroll..CSR</td>
</tr>
<tr>
<td>WORKID . .</td>
<td>ALUSMXLP</td>
</tr>
<tr>
<td>Type action</td>
<td>New Tablespace and Database name will be set for</td>
</tr>
<tr>
<td>E=Edit</td>
<td>selected Tables. Specify the TSNAME and DBNAME below.</td>
</tr>
<tr>
<td>Act Obj</td>
<td>Note: TS must exist or be created by this workid.</td>
</tr>
<tr>
<td>DB</td>
<td>Database name . . . . DEMOCJ</td>
</tr>
<tr>
<td>T</td>
<td>Tablespace name . . ASCIN3</td>
</tr>
<tr>
<td>T</td>
<td>END or ENTER to Process Quick Edit.</td>
</tr>
<tr>
<td>T</td>
<td>CANCEL to terminate Quick Edit.</td>
</tr>
<tr>
<td>QE</td>
<td>IXU. . . RDACRJ CLAIM</td>
</tr>
<tr>
<td></td>
<td>UCU. . DEMOCJ T0027_CLAIM CLAIM_NUM</td>
</tr>
<tr>
<td></td>
<td>TB . . . DEMOCJ T0110_REFERRAL</td>
</tr>
<tr>
<td></td>
<td>TB . . . DEMOCJ T0572_PROG_REF</td>
</tr>
<tr>
<td></td>
<td>TS . . . DEMOCJ CUST01</td>
</tr>
<tr>
<td></td>
<td>TS . . . DEMOCJ CUST02</td>
</tr>
<tr>
<td></td>
<td>TB . . . DEMOCJ EXAM_REPRT_CLASS</td>
</tr>
</tbody>
</table>
```

**5** Press Enter.

**NOTE**

You can specify only one table space. If you do not select a table space, the Table Quick Edit Confirmation dialog does not display a name for the database and the table space. If the table space specified does not exist, you must create the specified object by using the same work ID. New table spaces are not created dynamically.

The Mixed List panel is displayed with an indication of which objects were changed and the type of change that was made (such as add [A], like [L], migrate [M], or delete [D]). In addition, an asterisk (*) and a change-level indicator are displayed adjacent to the object that you changed since you built the mixed list.

**6** Press END.

The Object Specification panel is displayed. You can make additional changes, if necessary.

**7** Press END.

The WORKID Action Menu is displayed.
Changing data structures by using an existing work ID

**Where to go from here**

After you move multiple tables, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Changing data structures by using an existing work ID**

The CM/PILOT WKID_STRUCTURE_CHG script contains the steps to create an Analysis worklist to create, change, and delete data structures on the local DB2 subsystem using a work ID.

In this task, you will create and execute a task ID.

**Before you begin**

Using CHANGE MANAGER, create an alter-type Work ID and define the data structure changes with the Specification component.

**TIP**

To create an alter-type work ID, see page 357. To define the data structure changes, see page 383.

**To create a task ID**

1. On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.

   The CM/PILOT Main Menu is displayed.

2. Select **TASKIDs** and press **Enter**.

   The TASKID Action Menu is displayed.

3. Type the name of a new **TASKID** and select **Create a TASKID**. Then, press **Enter**.

   The Script Selection List is displayed.
4 Select Change data structures using an existing WORKID and press Enter.

The Create TASKID panel is displayed.

5 Specify the information for a task ID.

A Type the name of an existing alter-type WORKID or type the name of a work ID name template.

B (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

C (optional) Specify a Comment to describe the task ID.

D Press END.

The task ID is created and the TASKID Action Menu is displayed.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT Worklist.

B Type the data set name for the CM/PILOT JCL.

C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

A Type S to select Override CHANGE MANAGER options and then press Enter.

The Override CHANGE MANAGER Options panel is displayed.

B Select Override CHANGE MANAGER Options.
C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

---

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select the steps in the script that you want to modify, and press Enter.

C To display subsequent Step Settings panels, press Enter.

After the Step Settings panels are displayed, the CHANGE MANAGER Datasets panel is displayed.

D Specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press END until the TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.
To change the data structures, execute the Analysis worklist.

--- NOTE ---
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.

--- TIP ---
To execute a worklist, see page 562.

### Receiving DDL to create data structures

The CM/PILOT REC_DDL script contains the steps to import data structures for a new application and generate an Analysis worklist to create the structures on a local DB2 subsystem. The data structures cannot yet exist at the local subsystem and they must be defined in a DDL file or a worklist that was generated by CHANGE MANAGER.

This script also contains a DDL baseline step to create a baseline of the data structures received. You can use this baseline in comparisons with DDL files or worklists that were generated by CHANGE MANAGER that change the application.

You can use change rules that are defined in an inbound migrate profile when you create the CM/PILOT worklist.

In this task, you will create and execute a task ID.

#### Before you begin

Ensure that a DDL file or worklist that was generated by CHANGE MANAGER exists. Using CHANGE MANAGER, create a DDL baseline profile. For information, see Chapter 2, “Taking a snapshot of a database environment.” If you want to use change rules, use CHANGE MANAGER to create an inbound migrate profile.

--- TIP ---
To create an inbound migrate profile, see page 361.
To create a task ID

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3 Type a name for a new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

4 Select Receive DDL to create data structures, and press Enter.

   The Create TASKID panel is displayed.

5 Specify the information for a task ID.

   A CM/PILOT uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for WORKID or type the name of a work ID name template.

   B (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C (optional) Specify a Comment to describe the task ID.

   D Press END.

   The task ID is created and the TASKID Action Menu is displayed.

To execute the task ID

1 Select Execute a TASKID and press Enter.

   The TASKID Interface panel is displayed.
2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT Worklist.

B Type the data set name for the CM/PILOT JCL.

C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

A Type S to select Override CHANGE MANAGER options and then press Enter.

The Override CHANGE MANAGER Options panel is displayed.

B Select Override CHANGE MANAGER Options and press Enter.

C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

WARNING

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select the steps in the script that you want to modify, and press Enter.
C To display subsequent Step Settings panels, press Enter.

After the Step Settings panels are displayed, the TASKID Migrate Profiles panel is displayed.

D (optional) On the TASKID Migrate Profiles panel, type the name of the inbound Migrate Profile to be used in this task ID or type a wildcard pattern and select a profile from the list. Then, press Enter.

The TASKID Baseline Profiles panel is displayed.

E On the TASKID Baseline Profiles panel, type the name of the Baseline Profile to be used in this task ID or type a wildcard pattern and select a profile from the list. Then, press Enter.

The TASKID New Baselines panel is displayed.

F On the TASKID New Baselines panel, specify the information for the new baseline.

---

NOTE

If the baseline profile that you specified in step 5E on page 414 uses a baseline profile template, the name of the template is displayed. To use the baseline profile template, do not specify the name of the baseline. To override the baseline profile template, type a name or a template for Baseline Name.

If you are using baseline name templates to create names for baselines, the latest baseline that is created from the baseline profile that is specified for this script is used in the comparison step.

1. (optional) Type the name of the baseline.

2. To protect the baseline from deletion, type Y to select Protect.

3. (optional) Specify Comments to describe the baseline.

4. Press Enter.

The CHANGE MANAGER Datasets panel is displayed.

G On the CHANGE MANAGER Datasets panel, specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press END until the TASKID Interface panel is displayed.
6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

7 To import the data structures for a new application, execute the Analysis worklist.

---

**NOTE**
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.

---

**TIP**
To execute a worklist, see page 562.

---

**Changing data structures by using DML**

The CM/PILOT DML_STRUCTURE_CHG script contains the steps to create an Analysis worklist to update and delete data structures on a local DB2 subsystem. You can use DML to specify the changes to the data structures. The script uses a DML trigger to determine whether data structures exist. The DML trigger is the SEARCH condition of a DML WHERE clause. If no data structures are found, the CM/PILOT worklist processing stops, and an Analysis worklist is not created. If data structures are found, an Analysis worklist is created to apply the changes.
In this task, you will

1. create a task ID

2. create a DML statement

3. execute the task ID

**To create a task ID**

1 On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.

   The CM/PILOT Main Menu is displayed.

2 Select **TASKIDs** and press **Enter**.

   The TASKID Action Menu is displayed.

3 Type the name of a new **TASKID** and select **Create a TASKID**. Then, press **Enter**.

   The Script Selection List is displayed.

4 Select **Change data structures using a DML trigger** and press **Enter**.

   The Create TASKID panel is displayed.

5 Specify the information for a task ID.

   A **CM/PILOT** uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for **WORKID** or type the name of a work ID name template.

   B *(optional)* Type the name of a CM/PILOT **Application**. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C *(optional)* Specify a **Comment** to describe the task ID.

   D Type **S** to select **Create or Edit DML**.

   E Press **Enter**.

   The Select Object and Action for Change DML panel is displayed.
To create the DML statement

1  Specify the information to create a DML statement.

   A  Select an **Object** type to be included in a DML statement.

   B  Select an **Action**.

   **NOTE**
   
   If you select **Select Editor**, you cannot specify an **Object**. The **Select Editor** action enables you to enter the DML in an ISPF editor without first typing values for the SET and WHERE clause in a **Action Object** DML panel.

   C  Press **Enter**.

   The panel that is displayed is based on the object and the action that you select.

   D  Type a value for an attribute for the SET clause. Then, type an operator and a value for the WHERE clause and press **END**.

   The DML statement is displayed in the ISPF Editor, as shown in the example in Figure 104.

   **Figure 104  DML for updating database**

   ```
   UPDATE DATABASE SET 
   NAME               =    'DEMOCJ'
   WHERE 
   NAME               =    'DEMOME'
   : 
   ```

2  Edit your DML statement in the ISPF Editor, if necessary. For information about DML syntax, see the **ALTER and CHANGE MANAGER for DB2 Reference Manual**.

3  When you have finished editing the statement, press **END**.

   The List DML Statements panel is displayed.

4  You can perform the following actions by typing the action code adjacent to the DML statement:

   - **A** (add a new statement) displays the Select Object and Action for Change DML panel, so that you can create another DML statement. The new statement is placed at the end of the list.

   - **B** (browse) displays the ISPF Editor and places you in browse mode. After you browse the statement, press **END**.
Changing data structures by using DML

- **C (copy)** copies the DML statement and displays the ISPF Editor. After you modify the statement, press END. The copied and modified statement is located at the end of the list.

- **D (delete)** deletes a DML statement. The statement is immediately deleted; no delete confirmation panel is displayed.

- **E (edit)** displays the ISPF Editor.

**NOTE**

Syntax checking is performed on the values that you enter in the input fields of a DML panel. No syntax checking is performed when you edit DML statements.

5 When you have finished modifying the DML statements, press END until the TASKID Action Menu is displayed.

The task ID containing the DML statement for the script is created.

**To execute the task ID**

1 Select **Execute a TASKID** and press **Enter**.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

   A Type the data set name for the CM/PILOT Worklist.

   B Type the data set name for the CM/PILOT JCL.

   C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

   A Type **S** to select **Override CHANGE MANAGER options** and then press **Enter**.

   The Override CHANGE MANAGER Options panel is displayed.

   B Select **Override CHANGE MANAGER Options**.
C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

For more information about overriding the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

---

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

---

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select the steps in the script that you want to modify, and press Enter.

C To display subsequent Step Settings panels, press Enter.

After the Step Settings panels are displayed, the CHANGE MANAGER Datasets panel is displayed.

D Specify the data set names to be used by the Analysis and Execution components of CHANGE MANAGER. Then, press END until the TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.
To change your data structures, execute the Analysis worklist.

**NOTE**
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for **JCL Dataset Name** in the CHANGE MANAGER Datasets panel.

**TIP**
To execute a worklist, see page 562.

### Adding columns to the DB2 PLAN_TABLE table by importing DML

You can import DML to upgrade DB2 PLAN_TABLE tables with columns that were added by a later version of DB2.

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

#### Before you begin

Create a temporary table, as shown in Figure 105, that contains the new columns that were added by subsequent versions of DB2 for the PLAN_TABLE table, using the BMC CATALOG MANAGER product or SPUFI.

**Figure 105  SQL to create temporary table**

```sql
CREATE DATABASE DEMPLAN;
CREATE TABLESPACE TS1 IN DEMPLAN;
CREATE TABLE DEM.TESTPLAN
  (WHEN_OPTIMIZE CHAR(1) NOT NULL,
   QBLOCK_TYPE CHAR(6) NOT NULL,
   BIND_TIME TIMESTAMP NOT NULL,
   OPTHINT CHAR(8) NOT NULL WITH DEFAULT,
   HINT_USED CHAR(8) NOT NULL WITH DEFAULT,
   PRIMARY_ACCESS CHAR(1) NOT NULL WITH DEFAULT)
IN DEMPLAN.TS1
;```

420  ALTER and CHANGE MANAGER for DB2 User Guide, Volume 2
Adding columns to the DB2 PLAN_TABLE table by importing DML

To create and import the DML statement

1 In an ISPF editor, type the DML statement to match the one shown in Figure 106 and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

**TIP**
You can copy the DML statement from member ACMDMLU6 in the HLQ.DBCNTL data set.

---

**Figure 106  DML to add columns to the PLAN_TABLE**

```sql
-- EXAMPLE DML TO ADD PLAN TABLE COLUMNS
LIKE COLUMNS
JOIN SYIBM.SYSCOLUMNS DEM
SET NAME     = DEM.NAME,
            COLNO    = 200 + DEM.COLNO,
            COLTYPE  = DEM.COLTYPE,
            LENGTH   = DEM.LENGTH,
            NULLS    = DEM.NULLS,
            DEFAULT = DEM.DEFAULT
WHERE SYSIBM.SYSCOLUMNS.TBNAME    = 'PLAN_TABLE'
   --EXCLUDE THE SYSTEM PLAN TABLE FOR DB2
   AND SYSIBM.SYSCOLUMNS.TBCREATOR <> 'SYSIBM'
   AND SYSIBM.SYSCOLUMNS.COLNO     = 1
   --CHANGE TO WHATEVER YOU CALLED THE TEMP TABLE ABOVE
   AND DEM.TBNAME                  = 'PLAN_TABLE'
   --CHANGE TO WHATEVER YOU USED TO CREATE TEMP TABLE
   AND DEM.TBCREATOR               = 'DEM'
   --FIND PLAN TABLES THAT DO NOT CONTAIN NEW COLUMNS
   AND NOT EXISTS
   (SELECT * FROM SYIBM.SYSCOLUMNS X
    WHERE X.TBNAME     = SYSIBM.SYSCOLUMNS.TBNAME
    AND X.TBCREATOR   = SYSIBM.SYSCOLUMNS.TBCREATOR
    AND X.NAME        = DEM.NAME)
;
```

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.
Deleting aliases by importing DML

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**NOTE**

The worklist for this example should contain the following types of commands:

- data unload commands for DB2 PLAN_TABLE tables
- drop commands for the DB2 PLAN_TABLE tables
- alter table commands for the DB2 PLAN_TABLE tables that add the new NOT NULL WITH DEFAULT columns
- create table commands for the DB2 PLAN_TABLE tables that add the new NOT NULL columns, as well as the three new NOT NULL WITH DEFAULT columns
- load commands to load the data that was previously unloaded
- statistics and image copy commands
- rebind commands

**Deleting aliases by importing DML**

You can import DML to delete all aliases that refer to tables and views that are no longer in the DB2 catalog.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 107 and provide the information that is unique to your site.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*. 
Copying a column by importing DML

TIP
You can copy the DML statement to delete aliases from member ACMDM1D1 in the HLQ.DBCNTL data set.

Figure 107  DML for deleting aliases

```
DELETE ALIASES
WHERE NOT EXISTS
  (SELECT * FROM SYSIBM.SYSTABLES B
   WHERE ALIAS.TBNAME = B.NAME
   AND ALIAS.TBCREATOR = B.CREATOR)
```

The WHERE NOT EXISTS portion of the DML statement includes only those aliases that do not have a match in the SYSIBM.SYSTABLES. Because alias definitions are stored in the SYSIBM.SYSTABLES, the correlation B is used in the subselect to differentiate the alias row.

2  Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Copying a column by importing DML

You can import DML to add a new column to every table that contains a specific existing column.

NOTE
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.
1 In an ISPF editor, type the DML statement to match the one shown in Figure 108 and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 108  DML for copying a column

```
LIKE COLUMNS
SET
  NAME = 'ZIPCODE_2'
WHERE
  NAME = 'ZIPCODE' AND
  TBNAME LIKE '%' AND
  TBCREATOR LIKE 'RDACRJ%'
```

The example creates a column named ZIPCODE_2 in all tables that were created by RDACRJ and have a column named ZIPCODE. Each of the new ZIPCODE_2 columns will have the same attributes as those of ZIPCODE except for NAME and COLNO. Because a column number was not specified for ZIPCODE_2, the column number will be the next sequential number after the ZIPCODE column.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Updating index partitions for a database by importing DML

You can import DML to update all of the index partitions that are associated with a particular database. The DML uses a CASE statement to switch to different storage groups, depending on the index creator.
NOTE
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 109 and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 109  DML for updating index parts

```sql
UPDATE INDEXPARTS
JOIN SYSIBM.SYSINDEXES SYSIX
SYSIBM.SYSTABLES SYSTB
SET STORNAME =
(CASE SYSIBM.SYSINDEXPART.IXCREATOR
WHEN 'ALU075'
THEN 'AAMX1001'
ELSE 'AAVT9999'
END)
WHERE
SYSTB.DBNAME = 'ALUQA75' AND
SYSIX.TBCREATOR = SYSTB.CREATOR AND
SYSIX.TBNAME = SYSTB.NAME AND
INDEXPART.IXNAME = SYSIX.NAME AND
INDEXPART.IXCREATOR = SYSIX.CREATOR
```

In the example shown in Figure 109, the DML updates the index parts for database ALUQA75. The CASE statement assigns all index parts, except those created by ALU075, to storage group AAVT9999. Index parts that are created by ALU075 are assigned by the CASE statement to storage group AAMX1001.

NOTE
The product stores information about the index parts in the table parts CD table. You should make changes to the index parts and table parts with one DML statement.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.
You can import DML to add a WITH RESTRICT ON DROP clause to the table specification for all tables in a database.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. In an ISPF editor, type the DML statement to match the one shown in Figure 110 and provide the information that is unique to your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

   **Figure 110  DML to add WITH RESTRICT ON DROP**

   ```sql
   UPDATE TABLES
   SET
       CLUSTERTYPE = 'Y'
   WHERE
       DBNAME = 'DEMOCRJ'
   ;
   ```

2. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.
Updating table spaces or indexes with existing data sets by importing DML

You can import DML to update table spaces or indexes for which a physical data set exists.

---

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

---

1. In an ISPF editor, type the DML statement to match the one shown in Figure 111 and provide the information that is unique to your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

---

**TIP**

You can copy the DML statements to perform this task from member ACMDMMLUE in the HLQ.DBCNTL data set.

---

**Figure 111  DML to locate indexes with existing data sets**

```sql
UPDATE INDEXES
SET
  DEFINE = 'N'
WHERE
  -- The following subSELECT statement finds the indexes
  -- for which a physical data set exists.
  NOT EXISTS (SELECT * FROM SYSIBM.SYSINDEXPART IP
              WHERE IP.SPACE = -1 AND
              SYSIBM.SYSINDEXES.CREATOR = IP.IXCREATOR AND
              SYSIBM.SYSINDEXES.NAME   = IP.IXNAME )
;
```

2. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
Creating synonyms for selected tables by importing DML

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating synonyms for selected tables by importing DML

You can import DML to create a task ID that creates synonyms for selected tables and derives synonym names from the literal SYN_.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 112 and provide the information that is unique to your site.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**TIP**

You can copy the DML statement to create synonyms for selected tables from member ACMDMLU3 in the HLQ.DBCNTL data set.

Figure 112 DML to create synonyms (part 1 of 2)

```
LIKE SYNONYM
JOIN SYSIBM.SYSTABLES
SET
  NAME = 'SYN_'
  CONCAT SUBSTR(SYSIBM.SYSTABLES.NAME,1,12),
    CREATOR = 'NEWSYN',
    TBCREATOR = SYSIBM.SYSTABLES.CREATOR,
    TBNAME = SYSIBM.SYSTABLES.NAME
WHERE
  SYNONYMS.CREATOR = 'RDACRJ' AND
  SYNONYMS.NAME = 'SY_T_PRODUCT01' AND
```
In Figure 112, the DML has been edited to join to the SYSIBM.SYSTABLES. The SET clause uses the SUBSTR function to use only the first 12 characters of the TABLE name. The WHERE clause has a LENGTH specification. The LENGTH check ensures that only tables whose names have an additional four characters added to the beginning of the name for the synonym are utilized.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting VCAT-defined partitions to STOGROUP-defined partitions by importing DML

You can import DML to create a task ID that converts table space and index partitions from VCAT to STOGROUP. The script sets the storage allocation to implicit and removes invalid field values. If you specify a new primary or secondary space allocation in the DML statement, CM/PILOT uses those values. Otherwise, CM/PILOT calculates the new values from the VCAT fields for each partition.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.
In an ISPF editor, type the DML statement to match the one shown in Figure 113 and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

**TIP**
You can copy the DML statements to convert the partitions from member ACMDMLU7 in the HLQ.DBCNTL data set.

---

**Figure 113  DML to convert VCAT-defined partitions to STOGROUP-defined partitions**

```sql
UPDATE TABLEPARTS
SET STORTYPE = 'I',
    STORNAME = 'ACMX',
    VCATNAME = '
WHERE
    VCATNAME = 'DEBACAT'
;
UPDATE INDEXPARTS
SET STORTYPE = 'I',
    STORNAME = 'ACMX',
    VCATNAME = '
WHERE
    VCATNAME = 'DEBACAT'
;
```

In Figure 113, the DML sets the type of storage allocation to implicit (STOGROUP) for the specified VCAT.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Creating a materialized query table (MQT)

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.
   
   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.
   
   The Object Specification panel is displayed.

3. Specify the name of a view (or use wildcard characters) and press Enter.
   
   The Mixed List panel is displayed.

4. Type E in the Act column to edit a view (VW) object, and press Enter.
   
   The View Detail panel is displayed.

5. Type S to select Create Materialized Query Table based on this view, and press END.
   
   The Create MQT panel is displayed.

6. Specify the information about the materialized query table (MQT).
   
   A. Type the owner of the table.
   
   B. Type the name of the table.
   
   C. Type the name of an existing database and table space.
   
   D. Specify the attributes for the table.
   
   E. Specify whether the parent table space already exists in the DB2 catalog.
   
   F. To modify the columns in the table, type S to select Table Column List.
   
   G. To specify a comment for the table, type S to select Table Comment.

TIP

Some object names might be too long to be displayed on a panel. To enter a long object name, position the cursor on the object name and press the ZOOM (F4) key.
Type S to select MQT Parameters.

The Create MQT Parameters panel is displayed.

Specify the parameters for the MQT.

A  In the Maintained By field, type SYSTEM or USER.

B  In the Query Optimize field, type ENABLE or DISABLE.
   - ENABLE specifies that the table can be used to optimize queries.
   - DISABLE specifies that the table cannot be used to optimize queries.

C  In the MQT has Explicit Column List field, type Y or N.

D  To edit the text for the MQT, type S to select MQT Text.

E  To specify columns for the MQT, type S to select MQT Column List.

F  To edit the schema path, type S to select Schema Path.

G  To view the old text for the MQT, type S to select Old MQT Text.

H  Press END.

Where to go from here

After you create the MQT, perform the tasks in the following table to analyze a work ID and execute a worklist.

NOTE

The products do not guarantee the correct ordering of a worklist if the MQTs contain a large number of interdependencies.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Creating an application-period temporal table

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.
   
The WORKID Action Menu is displayed.

2. Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.
   
The Object Specification panel is displayed.

3. Specify the name of a database and table space (or use wildcard characters). Then, press **Enter**.
   
The Mixed List panel is displayed.

4. Type **CTB** in the **Act** column adjacent to the table space (TS) for which you want to create an application-period temporal table.

5. Specify the information about the application-period temporal table.

   A. Type the owner of the table.

   B. Type the name of the table.

   C. Specify the attributes for the table.

   D. Type **S** to select **Table Column List**.
      
The Table Columns List is displayed.

6. Specify the columns for the application-period temporal table.

   A. Specify the begin column with the following attributes, and then press **Enter**:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td><em>userDefinedApplicationBeginName</em></td>
</tr>
<tr>
<td>Column Type</td>
<td>DATE or TIMESTAMP</td>
</tr>
<tr>
<td>Nl</td>
<td>N</td>
</tr>
<tr>
<td>Df</td>
<td>Y</td>
</tr>
<tr>
<td>Pd</td>
<td>B</td>
</tr>
<tr>
<td>Default Value</td>
<td><em>userDefinedValue</em></td>
</tr>
</tbody>
</table>
NOTE

Until you specify the end column for the table, the product displays the following message:

BMC396627E - Too many BUSINESS TIME start or end columns or mismatch

You can disregard the message and specify the next column.

B Type L in the Act column adjacent to the begin column name. Then, press Enter.

C Specify the end column with the following attributes, and then press Enter:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>userDefinedApplicationEndTimeName</td>
</tr>
<tr>
<td>Column Type</td>
<td>DATE or TIMESTAMP</td>
</tr>
<tr>
<td>Nl</td>
<td>N</td>
</tr>
<tr>
<td>Df</td>
<td>Y</td>
</tr>
<tr>
<td>Pd</td>
<td>C</td>
</tr>
</tbody>
</table>

The product automatically sets the default value to SYSTEM ASSIGNED.

D Press END.

7 Press END.

Where to go from here

After you create the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Creating a system-period temporal table

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a database and table space (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Type CTB in the Act column adjacent to the table space (TS) for which you want to create a system-period temporal table.

5. Specify the information about the system-period temporal table.

   A. Type the owner of the table.

   B. Type the name of the table.

   C. Specify the attributes for the table.

   D. In the Link History TB field, type Y.

   E. Type S to select Table Column List.

      The Table Columns List is displayed.

6. Specify the columns for the system-period temporal table.

   A. Specify the begin column with the following attributes, and then press Enter:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>userDefinedSystemBeginName</td>
</tr>
<tr>
<td>Column Type</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
</tr>
<tr>
<td>Nl</td>
<td>N</td>
</tr>
<tr>
<td>Df</td>
<td>A</td>
</tr>
<tr>
<td>Pd</td>
<td>S</td>
</tr>
</tbody>
</table>
Creating a system-period temporal table

The product automatically sets the default value to AS ROW BEGIN.

**NOTE**

Until you specify the end column and the transaction ID column for the table, the product displays the following message:

BMC396684E - Too many SYSTEM TIME start, end or transaction id columns or mismatch

You can disregard the message and specify the next column.

**B** Type L in the Act column adjacent to the begin column name. Then, press Enter.

**C** Specify the end column with the following attributes, and then press Enter:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>userDefinedSystemEndName</td>
</tr>
<tr>
<td>Column Type</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
</tr>
<tr>
<td>Nl</td>
<td>N</td>
</tr>
<tr>
<td>Df</td>
<td>A</td>
</tr>
<tr>
<td>Pd</td>
<td>T</td>
</tr>
</tbody>
</table>

The product automatically sets the default value to AS ROW END.

**NOTE**

Until you specify the transaction ID column for the table, the product displays the following message:

BMC396684E - Too many SYSTEM TIME start, end or transaction id columns or mismatch

You can disregard the message and specify the next column.

**D** Type L in the Act column adjacent to the begin column name. Then, press Enter.

**E** Specify the transaction ID column with the following attributes, in the order specified in the table:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>userDefinedSystemTransactionIDName</td>
</tr>
<tr>
<td>Column Type</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
</tr>
</tbody>
</table>
Creating a history table for a system-period temporal table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pd</td>
<td>blank</td>
</tr>
<tr>
<td>NI</td>
<td>Y</td>
</tr>
<tr>
<td>Df</td>
<td>A</td>
</tr>
</tbody>
</table>

The product automatically sets the default value to AS TRANSACTION START ID.

**F** Press Enter.

**G** Press END.

7 On the Create Table panel, type S to select **History Table Information**.

8 On the History Table Information panel, specify the table owner and name.

9 Press END.

**Where to go from here**

After you create the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Creating a history table for a system-period temporal table**

1 On the CHANGE MANAGER Main Menu, select **WORKID**, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press Enter.

The Object Specification panel is displayed.

3 Specify the name of a database and table space (or use wildcard characters). Then, press Enter.
The Mixed List panel is displayed.

4 Type CTB in the Act column adjacent to the table space (TS) for which you want to create a history table for an associated system-period temporal table.

**NOTE**
You must create the history table in a table space other than the one in which the system-period temporal table resides.

5 Specify the information about the history table.

A Type the owner of the table.

B Type the name of the table.

C Specify the attributes for the table.

D Type S to select Table Column List.

The Table Columns List is displayed.

6 Specify the columns for the history table.

**NOTE**
The values for the attributes for the history table columns must be the same as the values for the system-period temporal table columns.

A Specify the begin column with the following attributes, and then press Enter.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td><code>userDefinedSystemBeginName</code></td>
</tr>
<tr>
<td>Column Type</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
</tr>
<tr>
<td>Nl</td>
<td>N</td>
</tr>
<tr>
<td>Df</td>
<td>N</td>
</tr>
<tr>
<td>Pd</td>
<td>blank</td>
</tr>
<tr>
<td>Default Value</td>
<td>blank</td>
</tr>
</tbody>
</table>

B Type L in the Act column adjacent to the begin column name. Then, press Enter.

C Specify the end column with the following attributes, and then press Enter.
Type L in the Act column adjacent to the begin column name. Then, press Enter.

Specify the transaction ID column with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>userDefinedSystemTransactionIDName</td>
</tr>
<tr>
<td>Column Type</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
</tr>
<tr>
<td>Nl</td>
<td>Y</td>
</tr>
<tr>
<td>Df</td>
<td>N</td>
</tr>
<tr>
<td>Pd</td>
<td>blank</td>
</tr>
<tr>
<td>Default Value</td>
<td>blank</td>
</tr>
</tbody>
</table>

Press Enter.

Press END.

Modify the base system-period temporal table.

On the Mixed List panel, type E in the Act column adjacent to the system-period temporal table.

On the Table Detail panel, at the Link History TB field, type Y.

Type S to select History Table Information.

On the History Table Information panel, specify the name of the history table.

Press Enter.

Press END.

Press END.
Creating a history table from an existing history table

Where to go from here

After you create the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating a history table from an existing history table

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a database and table space (or use wildcard characters). Then, press Enter.

4. On the Mixed List panel, type L in the Act column adjacent to the table space (TS) that contains an existing history table.

   For example, as shown in Figure 114, you would type L adjacent to the DEMO124B.TS2 table space.

   Figure 114 LIKE the table space with an existing history table
5 On the Create Tablespace panel, specify the information about the table space.
   A Specify a new name for the table space.
   B Press END.

6 On the Mixed List panel, type L in the Act column adjacent to the existing history table (#TBH).
   For example, as shown in Figure 115, you would type L adjacent to the DEMO124B.HIST_POLICY_INFO table.

**Figure 115** LIKE the existing history table

<table>
<thead>
<tr>
<th>Act</th>
<th>Object-Type</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>DEMO124B</td>
<td>TS1</td>
</tr>
<tr>
<td>TS</td>
<td>DEMO124B</td>
<td>HIST_POLICY_INFO</td>
</tr>
<tr>
<td>TB</td>
<td>DEMO124B</td>
<td>POLICY_INFO</td>
</tr>
<tr>
<td>TS</td>
<td>DEMO124B</td>
<td>TS2</td>
</tr>
<tr>
<td>L</td>
<td>#TBH.</td>
<td>DEMO124B HIST_POLICY_INFO</td>
</tr>
<tr>
<td>*L</td>
<td>TS</td>
<td>DEMO124B TS2H</td>
</tr>
</tbody>
</table>

7 On the Create Table panel, specify the information about the new history table.
   A Specify a new name for the table.
   B Specify the table space name that you specified in step 5A.
   C Select Parent Tablespace is being created within this WORKID.
   D Press END.

8 On the Mixed List panel, type E in the Act column adjacent to the base table.
   For example, as shown in Figure 116, you would type E adjacent to the DEMO124B.POLICY_INFO table.
9 Specify a new table name.

A On the Table Detail panel, type S to select History Table Information.

B Press END.

C On the History Table Information dialog, specify the history table name that you specified in step 7A.

D Press END.

Where to go from here

After you create the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

NOTE

The worklist will contain the following -SQL commands:

- CREATE TABLESPACE
- ALTER TABLE DROP VERSIONING
- CREATE TABLE
- ALTER TABLE ADD VERSIONING USE HISTORY TABLE

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Creating a history table from a base table

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.

   The WORKID Action Menu is displayed.

2. Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.

   The Object Specification panel is displayed.

3. Specify the name of a database and table space (or use wildcard characters). Then, press **Enter**.

4. On the Mixed List panel, type **L** in the **Act** column adjacent to the table space (TS) that contains an existing history table.

   For example, as shown in **Figure 117**, you would type **L** adjacent to the DEMO124B.TS1 table space.

   **Figure 117** LIKE the table space with an existing history table

<table>
<thead>
<tr>
<th>Act</th>
<th>Object-Type</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>. . . . . .</td>
<td>DEMO124B</td>
</tr>
<tr>
<td>L</td>
<td>TS . . . .</td>
<td>DEMO124B TS1</td>
</tr>
<tr>
<td></td>
<td>#TBH. . . .</td>
<td>DEMO124B HIST_POLICY_INFO</td>
</tr>
<tr>
<td></td>
<td>TB . . . .</td>
<td>DEMO124B POLICY_INFO</td>
</tr>
<tr>
<td></td>
<td>TS . . . .</td>
<td>DEMO124B TS2</td>
</tr>
<tr>
<td></td>
<td>#TBH. . . .</td>
<td>DEMO124B HIST_POLICY_INFO</td>
</tr>
</tbody>
</table>

5. On the Create Tablespace panel, specify the information about the table space.

   A Specify a new name for the table space.

   B Press END.

6. On the Mixed List panel, type **L** in the **Act** column adjacent to the base table.

   For example, as shown in **Figure 118**, you would type **L** adjacent to the DEMO124B.HIST_POLICY_INFO table.
7 On the Create Table panel, specify the information about the new history table.

A Specify a new name for the table.

B Specify the table space name that you specified in step 5A.

C Select Parent Tablespace is being created within this WORKID.

D At the Link History TB field, type N.

E Press END.

F On the Create Table panel, type S to select Table Column List.

G On the Table Columns List panel, specify N for the Df (Default) field for the starting, ending, and transaction ID columns.

H Scroll to the right, and type a blank for the Pd (Period) field for the starting and ending columns.

I Press END.

J Press END.

8 On the Mixed List panel, type E in the Act column adjacent to the base table.

For example, as shown in Figure 119, you would type E adjacent to the DEMO124B.POLICY_INFO table.
Creating a history table from a base table

Figure 119  Edit the base table

<table>
<thead>
<tr>
<th>Act</th>
<th>Object-Type</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>. . . . . .  DEM0124B</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>. . . . . DEM0124B  TS1</td>
<td></td>
</tr>
<tr>
<td>@TBH</td>
<td>. . . . . DEM0124B  HIST_POLICY_INFO</td>
<td></td>
</tr>
<tr>
<td>TB</td>
<td>. . . . . DEM0124B  POLICY_INFO</td>
<td></td>
</tr>
<tr>
<td>*L</td>
<td>TS . . . . . DEM0124B  TS1H</td>
<td></td>
</tr>
<tr>
<td>*L</td>
<td>TB . . . . . DEM0124B  POLICY_INFO_H</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>. . . . . DEM0124B  TS2</td>
<td></td>
</tr>
<tr>
<td>#TBH</td>
<td>. . . . . DEM0124B  HIST_POLICY_INFO</td>
<td></td>
</tr>
</tbody>
</table>

9 Specify information about the history table.

A On the Table Detail panel, type S to select History Table Information.

B On the History Table Information dialog, specify the history table name that you specified in step 7A.

C Press END.

D Press END.

Where to go from here

After you create the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

**NOTE**
The worklist will contain the following -SQL commands:

- CREATE TABLESPACE
- ALTER TABLE DROP VERSIONING
- CREATE TABLE
- ALTER TABLE ADD VERSIONING USE HISTORY TABLE

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist</strong>.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis</strong>.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Creating an external stored procedure

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a stored procedure (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. On the Command line, type CSP.

   The Create Procedure panel is displayed.

5. Specify the options for the external stored procedure.

   A. At Origin, type E (external).
   
   B. At Language, type ? to view a list of languages. Then, type S to select a language and press END.
   
   C. Specify any other options for the procedure.

6. To specify additional options for the external stored procedure, type S to select Proc Options.

   The Create External Procedure Options panel is displayed. Modify the options, and then press END to return to the Create Procedure panel.

7. To specify the parameters for the external stored procedure, type S to select Parameters.

   The Procedure Parameter List panel is displayed. Specify the parameters, and then press END to return to the Create Procedure panel.

8. To specify the comment for the external stored procedure, type S to select Comment.

   The Procedure Comment panel is displayed. Modify the comment, and then press END to return to the Create Procedure panel.
9 If the external stored procedure uses JAVA, type S to select JAVA Options.

The Create JAVA Options panel is displayed. Modify the options, and then press END to return to the Create Procedure panel.

10 To specify the path for the external stored procedure, type S to select Path.

The Procedure Package Path panel is displayed. Modify the path, and then press END to return to the Create Procedure panel.

11 Press END.

Where to go from here

After you create the stored procedure, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating a native SQL stored procedure

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name of a stored procedure (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

4 On the Command line, type CSP.
Creating a native SQL stored procedure

5 Specify the options for the native SQL stored procedure.

A At Procedure Version, type a value for the version or leave the value blank. The version must begin with an alphabetic character. DB2 defaults the value to V1.

B At Origin, type N (native).

C At Language, type SQL.

D Specify any other options for the procedure.

E Press Enter.

6 To specify additional options for the native SQL stored procedure, type S to select Proc Options.

The Create Native Procedure Options panel is displayed. Modify the options, and then press END to return to the Create Procedure panel.

7 To specify the parameters for the native SQL stored procedure, type S to select Parameters.

The Procedure Parameter List panel is displayed. Specify the parameters, and then press END to return to the Create Procedure panel.

8 To specify the comment for the native SQL stored procedure, type S to select Comment.

The Procedure Comment panel is displayed. Modify the comment, and then press END to return to the Create Procedure panel.

9 Specify the text for the native SQL stored procedure.

A Type S to select Text.

The product invokes the ISPF editor.

B Type the text for the body of the procedure.

If the body exceeds the width of the screen, you can scroll the panel to the right to provide additional text.

C Press END.

10 To specify the path for the native SQL stored procedure, type S to select Path.

The Procedure Package Path panel is displayed. Modify the path, and then press END to return to the Create Procedure panel.
Creating a new version of a native SQL stored procedure

11 Press END.

Where to go from here

After you create the stored procedure, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating a new version of a native SQL stored procedure

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name of a stored procedure (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

4 Type L in the Act column adjacent to the native SQL stored procedure (SPN) or the active native SQL stored procedure (SPNA).

**NOTE**

To create a new version of a native SQL stored procedure, you must copy an existing native SQL stored procedure. The procedure can exist in the DB2 structures or in a change request in a work ID.

5 Specify the options for the new version of the native SQL stored procedure.

**NOTE**

All versions of a native SQL stored procedure must have the same procedure signature. The procedure signature includes the Procedure Owner, Procedure Schema, Procedure Name, Parameter CCSID, number of parameters, and attributes for the parameters.
Creating a new version of a native SQL stored procedure

A At Procedure Version, type a new value for the version. The version must begin with an alphabetic character.

B At Add Version, type Y.

**NOTE**
If you type N at Add Version, the product attempts to create a new native SQL stored procedure.

C Specify any other options for the procedure.

6 Specify additional options for the native SQL stored procedure.

A Type S to select Proc Options.

B At Active, type Y or N to indicate whether the version of the procedure is the active version.

C Specify any other additional options for the procedure.

D Press END.

7 To specify the parameters for the native SQL stored procedure, type S to select Parameters.

The Procedure Parameter List panel is displayed. Specify the parameters, and then press END to return to the Create Procedure panel.

8 To specify the comment for the native SQL stored procedure, type S to select Comment.

The Procedure Comment panel is displayed. Modify the comment, and then press END to return to the Create Procedure panel.

9 To modify the text for the native SQL stored procedure, type S to select Text.

The product invokes the ISPF editor. Modify the text. If the body exceeds the width of the screen, you can scroll the panel to the right to provide additional text. Then, press END to return to the Create Procedure panel.

10 To view the previous text for the native SQL stored procedure, type S to select View Old Text.

The product invokes the ISPF browse utility, from which you can review the text for the native SQL stored procedure that existed before you made changes. Press END to return to the Create Procedure panel.
11  To specify the path for the native SQL stored procedure, type S to select Path.

The Procedure Package Path panel is displayed. Modify the path, and then press END to return to the Create Procedure panel.

12  Press END.

Where to go from here

After you create the new version, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Updating options for a native SQL stored procedure by importing DML**

You can import DML to create a task ID that updates an option for a native SQL stored procedure.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1  In an ISPF editor, type the DML statements to match the one shown in Figure 120. Customize the database name to match your local requirements.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**TIP**

You can copy the DML statements to estimate the space for all table spaces and indexes in a database from the member ACMDMLP2 in the HLQ.DBCNTL data set.
The DML specifies to update the DETERMINISTIC option in the SYSIBM.SYSROUTINES table and the IMMEDWRITE option in the SYSIBM.SYSPACKAGES table.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Updating parameters for a native SQL stored procedure by importing DML**

You can import DML to create a task ID that updates a parameter for all versions of a native SQL stored procedure.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statements to match the one shown in Figure 120. Customize the database name to match your local requirements.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*. 

---

**Figure 120  DML to update an option**

```
UPDATE ROUTINES SET
    DETERMINISTIC = 'N',
    IMMEDWRITE = 'N'
WHERE NAME = 'SPSQNG1' AND VERSION = 'V1'
    AND SCHEMA = 'RDACMC';
```
Estimating the space requirements for a table space

**TIP**

You can copy the DML statements to estimate the space for all table spaces and indexes in a database from the member ACMDMLP3 in the HLQ.DBCNTL data set.

---

**Figure 121  DML to update a parameter**

<table>
<thead>
<tr>
<th>UPDATE PARMS SET</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LENGTH = 5</td>
<td></td>
</tr>
<tr>
<td>WHERE NAME = 'SPOP0035' AND SCHEMA = 'IDML1ALL' AND PARMNAME = 'ICHAR3';</td>
<td></td>
</tr>
</tbody>
</table>

The DML specifies to update the LENGTH attribute for the ICHAR3 parameter in the SYSIBM.SYSPARMS table.

**2** Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

---

**Estimating the space requirements for a table space**

1 On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.

The WORKID Action Menu is displayed.

2 Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.

The Object Specification panel is displayed.

3 Specify the name of a table space (or use wildcard characters). Then, press **Enter**.

The Mixed List panel is displayed.
4 Type E in the Act column to edit a table space (TS) object. Then, press Enter.

The Tablespace Detail panel is displayed.

5 Type S to select Space Estimation. Then, press Enter.

The Tablespace Estimation panel is displayed.

**NOTE**

The Tablespace Estimation panel that is displayed might vary, depending on whether the table space is partitioned and whether DASD MANAGER PLUS is installed.

6 On the Tablespace Estimation panel, you can modify the values for the fields.

- To estimate the space for a nonpartitioned table space (Figure 122), you can complete one of the following steps:

*Figure 122  Tablespace Estimation panel (nonpartitioned)*
— Modify the values for the Avg Row Length (average row length) or the Nbr Rows (number of rows) field in the lower portion of the panel for a table in the table space.

— Type CO in the Ac column. In the Average Row Length panel that is displayed (Figure 123), type over the values for Average to Use in Estimate. Then press END.

Figure 123 Average Row Length panel

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Type</th>
<th>Length</th>
<th>Null</th>
<th>Projected Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSNO</td>
<td>CHAR</td>
<td>9</td>
<td>Y</td>
<td>10</td>
</tr>
<tr>
<td>EMPNO</td>
<td>CHAR</td>
<td>9</td>
<td>N</td>
<td>9</td>
</tr>
<tr>
<td>EMPNAME</td>
<td>CHAR</td>
<td>40</td>
<td>Y</td>
<td>41</td>
</tr>
<tr>
<td>EMPDEPT</td>
<td>CHAR</td>
<td>4</td>
<td>Y</td>
<td>5</td>
</tr>
<tr>
<td>EMPDEPTNAME</td>
<td>CHAR</td>
<td>36</td>
<td>Y</td>
<td>37</td>
</tr>
<tr>
<td>EMPHIREDATE</td>
<td>DATE</td>
<td>N</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>EMPDOB</td>
<td>DATE</td>
<td>Y</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>EMPAGE</td>
<td>SMALLINT</td>
<td></td>
<td>Y</td>
<td>3</td>
</tr>
</tbody>
</table>
To estimate the space for a partitioned table space (Figure 124), you can complete one of the following steps:

**Figure 124 Tablespace Estimation panel (partitioned)**

- Modify the values for the NbrRows (number of rows), PriQty (primary quantity), SecQty (secondary quantity), Aloc Unit (allocation unit), or Avg Row Length (average row length) field.

- To change the values for the NbrRows, PriQty, SecQty, and Aloc Unit fields for all of the partitions in the table space, type PA in the Ac column adjacent to the partition that you want to propagate. In the Space Estimation Attribute Propagation panel that is displayed (Figure 125), type over the values that you want to change. Then press END.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.
Figure 125  Space Estimation Attribute Propagation panel

TIP
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

To propagate the space estimates to the PriQty field for a specified partition, type PE in the Ac column.

NOTE
To propagate the space estimates for all of the partitions in the table space, type PEALL on the Command line.

— Type CO in the Ac column. In the Average Row Length panel that is displayed (see Figure 123), type over the values for Average to Use in Estimate. Then press END.

- To estimate the space for a partition-by-growth table space (Figure 126), you can modify the value of the Maxpartitions, Avg Row Len, or Nbr of Rows fields in the upper portion of the panel. The fields in the lower portion of the panel are read only.
Estimating the space requirements for a table space

Figure 126  Tablespace Estimation panel (partition-by-growth table space)

<table>
<thead>
<tr>
<th>ALUSTGSE R</th>
<th>-------------------</th>
<th>Tablespace Estimation (Partition By Growth) -------------------</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>---&gt;</td>
<td>Scroll. . PAGE</td>
</tr>
<tr>
<td>Tablespace Name:</td>
<td>J9QTALT.TRT10AIM</td>
<td></td>
</tr>
<tr>
<td>TB Name:</td>
<td>J9QTALT.T_T10AIM</td>
<td></td>
</tr>
<tr>
<td>Pagesize:</td>
<td>8</td>
<td>Max Row Len : 1255</td>
</tr>
<tr>
<td>Segsize:</td>
<td>4 (1-64)</td>
<td>Avg Row Len : 843 Proj</td>
</tr>
<tr>
<td>Priqty:</td>
<td>-1</td>
<td>Comp Row Len : 843</td>
</tr>
<tr>
<td>Secqty:</td>
<td>-1</td>
<td>Nbr of Rows : 1</td>
</tr>
<tr>
<td>Allocation Unit:</td>
<td>K (K,C,T)</td>
<td>Estimated</td>
</tr>
<tr>
<td>Device Type:</td>
<td>3390</td>
<td>Pages : 6</td>
</tr>
<tr>
<td>Freepage:</td>
<td>0 (0-255)</td>
<td>Space : 48 K</td>
</tr>
<tr>
<td>Pctfree:</td>
<td>5 (0-99)</td>
<td>Parts Needed : 1</td>
</tr>
<tr>
<td>Compress:</td>
<td>N (Y,N)</td>
<td>Rows/part : 1</td>
</tr>
<tr>
<td>Maxrows:</td>
<td>255</td>
<td>in last part : 0</td>
</tr>
<tr>
<td>Dssize:</td>
<td>4 G</td>
<td>NbrSec/part : 0</td>
</tr>
<tr>
<td>Pct Compressed:</td>
<td>0 (0-99)</td>
<td>in last part : 0</td>
</tr>
<tr>
<td>Maxpartitions:</td>
<td>256</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Avg</th>
<th>Part</th>
<th>NbrRows</th>
<th>Orig</th>
<th>Space</th>
<th>Rows/Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>1</td>
<td>0</td>
<td>843 Proj</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

More:

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

7 Press Enter.

The new space estimates for the number of pages, amount of space, or number of secondary allocations are displayed.

8 When you finish reviewing estimates, modify the values for the nonpartitioned table space in the upper left-hand portion of the panel, and press Enter. The values that you can modify are described in Table 36 on page 339.

9 When you have finished updating the fields, press END.

The Tablespace Detail panel is displayed.

10 Press END.

The Mixed List panel is displayed, as shown in Figure 127.
Estimating the space requirements for a table space

Chapter 4 Altering data structures in a database environment 459

Figure 127 Mixed List panel

11 Press END.

The Object Specification panel is displayed.

12 Press END.

The WORKID Action Menu is displayed.

Where to go from here

After you estimate the space requirements for a table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**NOTE**

If you do not make any changes which would affect Analysis, a space estimation marker is added. The $ marker denotes that an object is put away by Specification solely because of space estimation parameters.
**Estimating the space requirements for an index**

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a partitioned index (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Type E in the Act column to edit an index (IX) object. Then, press Enter.

   The Index Detail panel is displayed.

5. Type S to select **Space Estimation**. Then, press Enter.

   The Index Space Estimation panel is displayed, as shown in Figure 128.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.
6 (optional) Change the uniqueness of the index key.

7 Modify the following values in the lower portion of the panel for an index.

- **NbrRows**—the number of rows in this partition
- **Avg Len**—the average length of the key
- **Rows/Key**—the number of rows per key value
- **PriQty**—the primary quantity for the partition
- **SecQty**—the secondary quantity for the partition
- **AU (Aloc Unit)**—the allocation unit for the partition
- **Free Page**—the page increment for inserting a free page
- **Pct Free**—the percentage of free space to leave in each nonleaf page
- **NonLf KeyLn**—the average key length for a nonleaf page

**NOTE**

You can also type an action code in the Ac column to modify partitioned indexes:

- To propagate the space estimates to the **PriQty** field for a specified partition, type **PE**.
- To change the values for **NbrRows**, **PriQty**, **SecQty**, **AU (Aloc Unit)**, **Rows/Key**, and **Avg Len** fields for all of the partitions in the index, type **PA** adjacent to the partition that you want to propagate. In the Space Estimation Attribute Propagation panel that is displayed (Figure 129), type over the values that you want to change. Then press END.

To propagate the space estimates for all of the partitions in the index, type **PEALL** on the **Command** line.
8 Press Enter.

The new space estimates for the number of pages, the total amount of space, and the number of index levels that are required are displayed.

9 When you have finished reviewing the space estimates, modify the values for the index in the upper left-hand portion of the panel. The values that you can modify are described in Table 38 on page 343. Then, press Enter.

10 When you have finished updating the fields, press END.

The Index Detail panel is displayed.

11 Press END.

The Mixed List panel is displayed, as shown in Figure 130.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.
Figure 130  Mixed List panel

---

NOTE

If you do not make any changes which would affect Analysis, a space estimation marker is added. The $ marker denotes that an object is put away by Specification solely because of space estimation parameters.

12 Press END.

The Object Specification panel is displayed.

13 Press END.

The WORKID Action Menu is displayed.

Where to go from here

After you estimate the space requirements for an index, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Estimating space requirements for a table space based on user-specified values

You can use SSE to estimate space requirements for non-partitioned, partitioned, and partition-by-growth table spaces:

- “To estimate space requirements for a non-partitioned table space” on page 464
- “To estimate space requirements for a partitioned table space” on page 465
- “To estimate space requirements for a partition-by-growth table space” on page 466

NOTE
Space estimation is not available for XML objects, LOB objects, or objects that have partial statistics.

To estimate space requirements for a non-partitioned table space

1. From the command prompt, type SSE and press Enter.

2. On the Space Estimation panel, enter TS in the Object Type field and press Enter.

3. Estimate space requirements for the entire table space or for individual tables as follows:

   A. Change any of the modifiable values (highlighted in Figure 82 on page 345) as needed to correspond to your table space.

   NOTE
   For more information about a specific field, press F1.

   B. Press Enter to update the Estimated fields:

      - The Estimated box on the right side of the panel displays estimates for the entire table space.
      - The Estimated list in the bottom right corner shows estimates per table.

   NOTE
   The bottom of the panel shows more information about the table. In the Cmd column, you can enter I to insert, R to repeat, or D to delete lines.
Estimating space requirements for a table space based on user-specified values

4  (optional) If you want to see additional estimates based on different values, repeat step 3.

5  When finished, press END to exit.

To estimate space requirements for a partitioned table space

1  From the command prompt, type SSE and press Enter.

2  On the Space Estimation panel, enter TS in the Object Type field, and P in the Tablespace Type field. Then, press Enter.

3  Estimate space requirements for the entire table space or for individual partitions as follows:

   A  Change any of the modifiable values (highlighted in Figure 82 on page 345) as needed to correspond to your table space.

   **NOTE**

   For more information about a specific field, press F1.

   B  Press Enter to update the Estimated fields:

      ■ The Estimated box on the right side of the panel displays estimates for the entire table space.

      ■ The Estimated list in the bottom right corner shows estimates per partition.

   **NOTE**

   The bottom of the panel shows more information about the table. In the Cmd column, you can enter I to insert, R to repeat, or D to delete lines.

4  (optional) If you want to see additional estimates based on different values, repeat step 3.

   **NOTE**

   This space estimation function does not support the Average Length feature for table columns.

5  When finished, press END to exit.
To estimate space requirements for a partition-by-growth table space

1 From the command prompt, type `SSE` and press Enter.

2 On the Space Estimation panel, enter TS in the Object Type field, and G in the Tablespace Type field. Then, press Enter.

3 Estimate space requirements for the entire table space or for individual partitions are as follows:

   A Change any of the modifiable values (highlighted in Figure 82 on page 345) as needed to correspond to your table space.

   ---
   **NOTE**
   For more information about a specific field, press F1.
   ---

   B Press Enter to update the Estimated fields:

   - The Estimated box on the right side of the panel displays estimates for the entire table space.
   - The bottom of the panel shows more information about the table itself.

4 (optional) If you want to see additional estimates based on different values, repeat step 3.

5 When finished, press END to exit.

Estimating space requirements for an index based on user-specified values

You can use SSE to estimate space requirements for an index.

---
**NOTE**
Space estimation is not available for XML objects, LOB objects, or objects that have partial statistics.
---
To estimate space requirements for an index

1. From the command prompt, type `SSE` and press Enter.

2. On the Space Estimation panel, enter `IX` in the Object Type field, and press Enter.

3. Estimate space requirements for the index as follows:
   - Change any of the modifiable values (highlighted in Figure 82 on page 345) as needed to correspond to your index.
   - Press Enter to update the Estimated fields:
     - The Estimated box on the right side of the panel displays estimates for the index.
     - The Estimated list in the bottom right corner shows estimates for the index.

4. (optional) If you want to see additional estimates based on different values, repeat step 3.

5. When finished, press END to exit.

Setting space estimation parameters by importing DML

You can import DML to specify space estimation parameters by using DML statements, user-defined tables, and existing work IDs. For more information about space estimation, see “Estimating space” on page 336.

NOTE
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.
1 In an ISPF editor, type the DML statement to match the statement shown in one of the following examples and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 131 shows an example of setting the cardinality of a table to 50 and setting the average row length to 2000.

**Figure 131  DML to update a table**

```sql
UPDATE TABLES SET
  CARDF = 50,
  ROWAVG = 2000
WHERE
  SYSIBM.SYSTABLES.DBNAME = 'ALUQA75'
  AND SYSIBM.SYSTABLES.CREATOR = 'ALUO750'
  AND SYSIBM.SYSTABLES.NAME = 'TBS1A'
;
```

Figure 132 shows an example of setting the cardinality of a table partition to 153.

**Figure 132  DML to update a table partition**

```sql
UPDATE TABLEPART SET
  CARDF = 153
WHERE
  PARTITION > 0
  AND TSNAME = 'P1'
  AND DBNAME = 'ALUQA75'
;
```

Figure 133 shows an example of setting the cardinality of an index partition to 50 and setting the ROWSPKEY to 500.

**Figure 133  DML to update an index partition**

```sql
UPDATE INDEXPARTS SET
  CARDF = 50,
  ROWSPKEYF = 500
WHERE
  PARTITION > 0
  AND IXNAME = 'IXP1A__C'
  AND IXCREATOR = 'ALUO750'
;
```

Figure 134 shows an example of using the JOIN command to copy the space estimation entries for all tables that were previously specified in a work ID.
Figure 134  DML to update a table by using the JOIN command

```
UPDATE TABLES
JOIN ACM83.CD_TABLE CD
SET CARDF = CD.CARDF,
       ROWAVG = CD.ROWAVGSTATS
WHERE
    SYSIBM.SYSTABLES.DBNAME = CD.DBNAME
    AND SYSIBM.SYSTABLES.CREATOR = CD.TBOWNER
    AND SYSIBM.SYSTABLES.NAME = CD.TBNAME
    AND CDNAME = 'SE5310A'
    AND CDOWNER = 'AUC'
;
```

Figure 135 shows an example of using the JOIN command to copy the space estimation entries for all table partitions that were previously specified in a work ID and to change all of the primary quantities to 100.

Figure 135  DML to update a table partition by using the JOIN command

```
UPDATE TABLEPART
JOIN ACM83.CD_TABLEPART CD
SET CARDF = CD.CARDF,
        PQTY = 100
WHERE
    TABLEPART.TSNAME = CD.TSNAME
    AND TABLEPART.DBNAME = CD.DBNAME
    AND TABLEPART.PARTITION = CD.PARTID
    AND CD.PARTTYPE = 'T'
    AND CDNAME = 'SE5310A'
    AND CDOWNER = 'AUC'
;
```

Figure 136 shows an example of using the JOIN command to copy the space estimation entries for all index partitions that were previously specified in a work ID and to change all of the primary quantities to 100.

**NOTE**

The index partition information is stored in the table partition CD tables.

Figure 136  DML to update an index partition by using the JOIN command (part 1 of 2)

```
UPDATE INDEXPART
JOIN ACM83.CD_TABLEPART CD
SET CARDF = CD.CARDF,
       ROWSPKEYF = CD.ROWSPKEYF,
       PQTY = 100
WHERE
    TABLEPART.TSNAME = CD.TSNAME
    AND TABLEPART.DBNAME = CD.DBNAME
    AND TABLEPART.PARTITION = CD.PARTID
    AND CD.PARTTYPE = 'T'
    AND CDNAME = 'SE5310A'
    AND CDOWNER = 'AUC'
;
```
Estimating the primary and secondary quantities in table space and index partitions by importing DML

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Figure 136  DML to update an index partition by using the JOIN command (part 2 of 2)

```
INDEXPARTS.IXNAME = CD.IXNAME
AND INDEXPARTS.IXCREATOR = CD.IXOWNER
AND INDEXPARTS.PARTITION = CD.PARTID
AND CD.PARTTYPE = 'I'
AND CDNAME = 'SE5310A'
AND CDOWNER = 'AUC'
;
```

1 In an ISPF editor, type the DML statements to match the one shown in Figure 137. Customize the database name to match your local requirements.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Estimating the primary and secondary quantities in table space and index partitions by importing DML

You can import DML to create a task ID that performs space estimation for all table space partitions and index partitions in a database.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.
Setting a secondary quantity in table space and index partitions by importing DML

**TIP**

You can copy the DML statements to estimate the space for all table spaces and indexes in a database from the member ACMDMLU5 in the HLQ.DBCNTL data set.

---

**Figure 137  DML to estimate the primary and secondary quantities**

```
SET SPACE ESTIMATION IMPLICIT -- turn space estimation on
-- PQTY = BMCSPACE -- default options
-- ,SQTY = BMCSPACE/10
UPDATE DATABASE SET
WHERE
  NAME = 'yourDB' -- select database
```

The DML specifies to implicitly estimate the space for all table space partitions and index partitions in a database. The primary quantity is set to the value that is returned from space estimation. The secondary quantity is set to one tenth of the value that is returned from space estimation.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

---

**Setting a secondary quantity in table space and index partitions by importing DML**

You can import DML to create a task ID that sets a secondary quantity to a factor of a primary quantity for all STOGROUP-defined table space and index partitions in a database.
1 In an ISPF editor, type *one* of the DML statements shown in Figure 138 and provide the information that is unique to your site.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

**NOTE**

PRIQTY and SECQTY are expressed in 4K pages.

---

**Figure 138  Example DML for setting SQTY**

```
UPDATE TABLEPART
SET SQTY = PQTY / 2  -- set secondary quantity to half of primary quantity
WHERE
   DBNAME LIKE 'CRJ%'
   AND (PQTY BETWEEN 4 AND 60) -- pages, not K
   AND STORTYPE = 'I'  -- STOGROUP-defined
;

UPDATE TABLEPART
SET SQTY = 2 * PQTY  -- set secondary quantity to twice primary quantity
WHERE
   DBNAME LIKE 'CRJ%'
   AND PQTY < 4  -- pages, not K
   AND STORTYPE = 'I'  -- STOGROUP-defined
;

UPDATE INDEXPART
JOIN SYSIBM.SYSINDEXES IX
SET SQTY = PQTY * 1.5 --set secondary quantity to 1 1/2 of primary quantity
WHERE
   STORTYPE = 'I'
   AND IXNAME = IX.NAME
   AND IXCREATOR = IX.CREATOR
   AND IX.DBNAME LIKE 'CRJ%'
;
```

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
Creating objects for the SHRLEVEL CHANGE HSSC process

You can use CHANGE MANAGER with the BMC Recovery Management for DB2 High Speed Structure Change (HSSC) feature. This feature reduces the downtime and CPU consumption required to convert one type of table space to another, or to modify table space attributes.

The SHRLEVEL CHANGE HSSC process enables other applications to have read-write access during transformation. This feature uses the BMC Log Master for DB2 product and the BMC High-Speed Apply Engine product to capture and apply changes during the transformation.

You can import DML to create and modify a target set of objects for the SHRLEVEL CHANGE HSSC process. The DML performs the following tasks:

- edits the source table and indexes, and renames them
- modifies the dependent alias, synonym, and trigger objects on the source table and refers the objects to the target table
- edits the target table and indexes, and renames them to the source table and indexes

To create objects for the SHRLEVEL CHANGE HSSC process

1. Copy the DML statements from member ACMDMLT2 in the HLQ.DBCNTL data set.

2. Edit the DML statements (Figure 139) in an ISPF editor. Provide the information that is required for your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Figure 139  DML statements to create and modify the target set of objects (part 1 of 3)

```sql
-- CHANGE <TBOWNER> OWNER ALL
-- CHANGE <TBNAME> NAME ALL
-- NO OTHER MODIFICATIONS SHOULD BE NEEDED
--
-- If your source tablespace is a multi-table tablespace and you have multiple
-- target tablespaces you can replicate these statements for each source
-- table and process it all in the same worklist UNLESS there is referential
-- defined amongst two of your source tables. In that case you must have a
-- separate worklist for each source table.
--
-- RENAME NEW TABLE TO ORIGINAL
UPDATE TABLES
SET NAME = SUBSTR(NAME, 4, 100)
WHERE
CREATOR = '<TBOWNER>'
AND NAME = 'NEW<TBNAME>'
;
-- RENAME ORIGINAL TABLE TO BACKUP
UPDATE TABLES
SET NAME = SUBSTR('BKUP' || NAME, 1, 100)
WHERE
CREATOR = '<TBOWNER>'
AND NAME = '<TBNAME>'
;
-- RENAME NEW INDEX TO ORIGINAL
UPDATE INDEXES
SET NAME = SUBSTR(NAME, 4, 100)
WHERE
TBCREATOR = '<TBOWNER>'
AND TBNAME = 'NEW<TBNAME>'
;
-- RENAME ORIGINAL INDEX TO BACKUP
UPDATE INDEXES
SET NAME = SUBSTR('BKUP' || NAME, 1, 100)
WHERE
TBCREATOR = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;
-- LIKE ORIGINAL ALIASES AND REFERENCE NEW TABLE
LIKE ALIAS
SET TBNAME = SUBSTR('NEW' || TBNAME, 1, 100)
WHERE
TBCREATOR = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;
```
-- DROP ALIASES ON ORIGINAL TABLE
DROP ALIAS
WHERE
TBCREATOR  = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;

-- LIKE ORIGINAL SYNONYMS AND REFERENCE NEW TABLE
LIKE SYNONYM
SET TBNAME =  SUBSTR('NEW' || TBNAME, 1, 100)
WHERE
TBCREATOR  = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;

-- DROP SYNONYMS ON ORIGINAL TABLE
DROP SYNONYM
WHERE
TBCREATOR  = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;

-- LIKE ORIGINAL TRIGGERS AND REFERENCE NEW TABLE
LIKE TRIGGER
SET TBNAME =  SUBSTR('NEW' || TBNAME, 1, 100)
WHERE
TBOWNER    = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;

-- DROP TRIGGERS ON ORIGINAL TABLE
DROP TRIGGER
WHERE
TBOWNER    = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;

-- LIKE ORIGINAL FK  AND REFERENCE NEW TABLE - AS CHILD/RECURSIVE
LIKE RELS
SET TBNAME =  SUBSTR('NEW' || TBNAME, 1, 100)
WHERE
CREATOR    = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;

-- DROP FK ON ORIGINAL TABLE - AS CHILD/RECURSIVE
DELETE RELS
WHERE
CREATOR    = '<TBOWNER>'
AND TBNAME = '<TBNAME>'
;
You can use CHANGE MANAGER with the BMC Recovery Management for DB2 High Speed Structure Change (HSSC) feature. This feature reduces the downtime and CPU consumption required to convert one type of table space to another, or to modify table space attributes.
The SHRLEVEL REFERENCE HSSC process uses a VSAM image copy (such as an Online Consistent Copy or Instant Snapshot) as input, reads each page, and transforms the page to fit the new definition. The process then moves the page to the new object. During this transformation, applications can have read-only access or concurrent (update) access to the objects. Using concurrent access reduces the length of the outage.

You can import DML to modify and migrate objects for the SHRLEVEL REFERENCE HSSC process. The DML performs the following tasks:

- edits the source table space and renames it
- makes changes to the SEGSIZE or DSSIZE table space attributes
- includes tables, indexes, unique constraints, check constraints, and authorizations in the migration
- renames the tables and indexes

1 Copy the DML statements from member ACMDMLT1 in the HLQ.DBCNTL data set.

2 Edit the DML statements (Figure 140) in an ISPF editor. Provide the information that is required for your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 140  DML statements to modify and migrate objects (part 1 of 2)

```sql
-- After you edit the DML statements, you can use CHANGE MANAGER
-- to import them.
-- Reminder: Generate your worklist without DATA. The Recovery
-- Management for DB2 High Speed Structure Change (HSSC) feature
-- will move the data into the new structures.

-- CHANGE <TBOWNER> OWNER ALL
-- CHANGE <TBNAME>  NAME  ALL
-- NAME OF TS NEEDS TO BE MODIFIED '<TS_FROM>' '<TS_TO>'
-- NO OTHER MODIFICATIONS SHOULD BE NEEDED
--
-- If your source tablespace is a multi-table tablespace and you
-- have multiple target tablespaces, you can replicate these
-- statements for each target tablespace and run them in a
-- single worklist.
--
-- IMPORT THIS DML INTO AN ALTER WORKID. IT WILL CONVERT THE
-- WORKID TO A MIGRATE WORKID
```
Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC process

Figure 140  DML statements to modify and migrate objects (part 2 of 2)

```sql
-- AFTER DML IS IMPORTED USE SPECIFICATION TO MODIFY NEW TS WITH
-- DESIRED CHARACTERISTIC
-- LIKE TS FOR ORIGINAL TABLE.
LIKE TABLESPACE
JOIN SYSIBM.SYSTABLES TABLS
INCLUDE TABLES, AUTHORIZATIONS, CHECKS,
INDEXES, CONSTRAINTS
SET NAME = REPLACE(
SYSIBM.SYSTABLESPACE.NAME,'<TS_FROM>','<TS_TO>')
WHERE
  SYSIBM.SYSTABLESPACE.DBNAME = SYSIBM.SYSTABLES.DBNAME
  AND SYSIBM.SYSTABLESPACE.NAME = SYSIBM.SYSTABLES.TSNAME
  AND SYSIBM.SYSTABLES.CREATOR = '<TBOWNER>'
  AND SYSIBM.SYSTABLES.NAME = '<TBNAME>'
;
-- MIGRATE ORIGINAL TABLE
-- ADDING NEW PREFIX TO ITS NAME
MIGRATE TABLES
INCLUDE AUTHORIZATIONS, CHECKS,
  CONSTRAINTS
SET NAME = SUBSTR('NEW' || NAME, 1, 100)
  ,TSNAME = '<TS_TO>'
WHERE
  CREATOR = '<TBOWNER>'
  AND NAME = '<TBNAME>'
;
-- LIKE INDEXES ON ORIGINAL TABLE
-- ADDING NEW PREFIX TO THEIR NAMES
LIKE INDEXES
SET NAME = SUBSTR('NEW' || NAME, 1, 100)
  ,TBNAME = SUBSTR('NEW' || TBNAME, 1, 100)
WHERE
  TBCREATOR = '<TBOWNER>'
  AND TBNAME = '<TBNAME>'
;
```

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.
Converting table spaces to partition-by-growth table spaces by importing DML

You can import DML to convert one of the following types of table spaces to partition-by-growth table spaces:

- table-controlled partitioned table spaces
- index-controlled partitioned table spaces
- nonpartitioned simple table spaces
- segmented table spaces

The DML statement automatically deletes all but the first partition of a partitioned table space and any dependent indexes.

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. Copy the DML statement from member ACMDMLUC in the HLQ.DBCNTL data set.

2. Edit the DML statement (Figure 141) in an ISPF editor. Provide the information that is required for your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

**Figure 141** DML to convert to partition-by-growth table spaces (part 1 of 2)
Converting a nonpartitioned table space to an index-controlled partitioned table space

Figure 141 DML to convert to partition-by-growth table spaces (part 2 of 2)

```sql
MAXPARTITIONS = 10
WHERE
  DBNAME = 'MG025'
  AND NTABLES = 1
;
```

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting a nonpartitioned table space to an index-controlled partitioned table space

You can convert a nonpartitioned table space into an index-controlled partitioned table space. You can also add a clustering, partitioning index and limit keys for the additional partitions. In this example, a base table exists in the nonpartitioned table space. This task also provides instructions for creating a clustering index for an existing base table and creating a partitioning index.

**TIP**

To create the auxiliary objects for the table space, see page 525.

In this task, you will

1. display the list of table space partitions
2. specify the additional partitions
3. create the clustering, partitioning index for the base table
Converting a nonpartitioned table space to an index-controlled partitioned table space

To display the list of table space partitions

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a nonpartitioned table space (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Type PT in the Act column adjacent to the nonpartitioned table space that you want to modify.

   The Tablespace Parts List panel is displayed.

To specify the additional partitions

1. From the Tablespace Parts List panel, specify whether the table space will be a regular partitioned table space or a large partitioned table space.

2. In the Dssize field, specify the maximum size for each partition or data set.

3. For VCAT-defined table spaces, specify whether the table space should use a variable control interval size (CISIZE):
   - Type Y to use a CISIZE that is equal to the page size of the table space (4, 8, 16, or 32 KB).
   - Type N to use a CISIZE of 4 KB, regardless of the page size.
   - Leave the Variable CISIZE field blank to have the product use the value of the DB2 subsystem parameter DSNZPARM to create the VSAM data sets.

4. Type I in the Act column adjacent to the existing partition. Then, press Enter.

   A blank line is inserted after the line on which you typed I.
Converting a nonpartitioned table space to an index-controlled partitioned table space

**NOTE**
You can also type L in the Act column to duplicate an existing partition definition. You can then edit the attributes of the partition by typing over them or by typing E in the ACT column adjacent to the partition that you want to modify.

5 Type the name of the Stogroup or VCAT and specify values for the following fields:

- **Volume**—the first volume upon which a VCAT partition resides
- **PriQty**—the primary quantity for the partition
- **SecQty**—the secondary quantity for the partition
- **Aloc Unit**—the allocation unit for the partition
- **Free Page**—the page increment for inserting a free page
- **Pct Free**—the percentage of free space to leave in each nonleaf page
- **Gbp Cach**—the group buffer pool cache setting for the partition
- **Comp**—the compression attribute for the partition
- **SMS Storclass**—for VCAT partitions, the SMS storage class for the partition
- **SMS Dataclass**—for VCAT partitions, the SMS data class for the partition
- **SMS Mgmtclass**—for VCAT partitions, the SMS management class for the partition
- **Track Mod**—the indicator for tracking page modifications in the space map pages

6 Repeat step 4 and step 5 for each partition that you want to create.

7 Press END.

The Mixed List panel is displayed.

**To create the clustering, partitioning index for the base table**

**NOTE**
Before you create the clustering, partitioning index for the base table, determine if such an index already exists for the table. You can have only one clustering, partitioning index for the table. If a clustering, partitioning index exists, perform one of the following actions:

- Determine whether you want to use the existing index or create a new index. If you choose to create a new index by following the steps in this section, you must delete the existing index.

- Create additional partitions for the existing index on the Index Parts List panel. Do not follow the steps in this section to create the index.
1 From the Mixed List panel, type CIX in the Act column adjacent to the base table. Then, press Enter.

The Create Index panel is displayed.

2 Specify the attributes for the index.

--- TIP ---

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Specify the owner of the index.

B Specify the name of the index.

C Type Y to indicate that the index is a clustering index.

D Type S to select Keys and TB Cols.

E Press Enter.

The Index Key Mixed List panel is displayed, as shown in Figure 142.

Figure 142 Index Key Mixed List panel
3 Specify the index key.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Type C in the Act column adjacent to the column name.

B Type A in the Act column adjacent to the Index Key.

C Press Enter.

The Index Key Mixed List panel is displayed as shown in Figure 143. The index key column is displayed below the index key.

**Figure 143 Index Key Mixed List panel**

D Press END.

The Create Index panel is displayed.

4 Type S to select Partitions.

The Index Parts List panel is displayed.

5 From the Index Parts List panel, select Partitioned for the partitioning status.

6 Type I in the Act column adjacent to a partition. Then, press Enter.
Converting a nonpartitioned table space to an index-controlled partitioned table space

A blank line is inserted after the line on which you typed I.

---

**NOTE**

You can also type L in the Act column to duplicate an existing partition definition. You can then edit the attributes of the partition by typing over them or by typing E in the ACT column adjacent to the partition that you want to modify.

---

7 Type the name of the Stogroup or VCAT and specify values for the following fields:

- **Volume**—the first volume upon which a VCAT partition resides
- **PriQty**—the primary quantity for the partition
- **SecQty**—the secondary quantity for the partition
- **Aloc Unit**—the allocation unit for the partition
- **Free Page**—the page increment for inserting a free page
- **Pct Free**—the percentage of free space to leave in each nonleaf page
- **Gbp Cach**—the group buffer pool cache setting for the partition
- **SMS Storclass**—for VCAT partitions, the SMS storage class for the partition
- **SMS Dataclass**—for VCAT partitions, the SMS data class for the partition
- **SMS Mgmtclass**—for VCAT partitions, the SMS management class for the partition
- **Limitkey**—the limit key value for the partition

---

**NOTE**

In the Limitkey field, which is displayed by typing the LK action code in the Act column, you must specify nonprintable and nonviewable characters, such as null and control characters, as external hexadecimal strings. For more information, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

---

8 Press END.

The Create Index panel is displayed. The panel indicates the number of partitions.

---

**NOTE**

Now that you have increased the number of partitions in the table space, you need to create auxiliary objects for the base table for each of the new partitions. If the base table in the table space already contains auxiliary objects, BMC recommends that you drop the existing auxiliary objects on the Auxiliary Objects List panel and then create the new auxiliary objects. (If you LIKEd the existing objects, you would need to go through the time-consuming and error-prone processes of ensuring that the names and locations of the parent table spaces are correct.) For information about creating auxiliary objects for a partitioned table space, see “Creating auxiliary objects for a partitioned table space” on page 525.

---

9 Press END until the WORKID Action Menu is displayed.
Converting a nonpartitioned table space to a table-controlled partitioned table space

Where to go from here

After you modify the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting a nonpartitioned table space to a table-controlled partitioned table space

In this task, you will

1. specify to convert the table space
2. specify partitioning key columns
3. specify the additional partitions
4. specify limit keys for each partition

To specify the conversion

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.
   
The WORKID Action Menu is displayed.

2. Type the name of a nonpartitioned table space (or use wildcard characters). Then, press Enter.
   
The Mixed List panel is displayed.

3. Specify the name of a nonpartitioned table space (or use wildcard characters). Then, press Enter.
   
The Table Detail panel is displayed.

4. Type E in the Act column adjacent to a table that you want to convert to table-controlled partitioning.
   
The Table Detail panel is displayed.
5 Type S to select Convert to Table Controlled Partitioning.

The Table Detail panel is displayed. A warning message is displayed on the panel, which states that you must add partitions for the table space, add limit keys for the partitions, and specify partitioning key columns.

6 Press Enter to dismiss the warning message.

The Table Detail panel now provides the following selections:

- Partitioning Key Columns
- Parts List and LimitKey Values

To specify partitioning key columns

1 On the Table Detail panel, type S to select Partitioning Key Columns.

The Partitioning Key Mixed List panel is displayed.

2 Specify the columns in the partitioning key.

   A Type C in the Act column adjacent to the column name that you want to use.

   **NOTE**

   To select more than one name, type CC in the Act column adjacent to the first column name and CC in the Act column adjacent to the last column name.

   B Type A in the Act column adjacent to the Partitioning Key.

   C Press Enter.

   The specified column is displayed below the partitioning key.

   D Press END.

To add partitions to the table space

1 On the Table Detail panel, type S to select Parts List and LimitKey Values.

The Tablespace Parts List is displayed.

2 Specify whether the table space will be a regular partitioned table space or a large partitioned table space.
Converting a nonpartitioned table space to a table-controlled partitioned table space

3 Specify the maximum size for each partition or data set.

If you plan for the table space to have more than 64 partitions, IBM recommends that you specify a value for Dssize.

4 For VCAT-defined table spaces, specify whether the table space should use a variable control interval size (CISIZE):

- Type Y to use a CISIZE that is equal to the page size of the table space (4, 8, 16, or 32 KB).
- Type N to use a CISIZE of 4 KB, regardless of the page size.
- Leave the Variable CISIZE field blank to have the product use the value of the DB2 subsystem parameter DSNZPARM to create the VSAM data sets.

5 Type I in the Act column adjacent to the existing partition. Then, press Enter.

A blank line is inserted after the line on which you typed I.

**NOTE**
You can also type L in the Act column to duplicate an existing partition definition. You can then edit the attributes of the partition by typing over them or by typing E in the ACT column adjacent to the partition that you want to modify.

6 Type the name of the Stogroup or VCAT and specify values for the following fields:

- **Volume**—the first volume upon which a VCAT partition resides
- **PriQty**—the primary quantity for the partition
- **SecQty**—the secondary quantity for the partition
- **Aloc Unit**—the allocation unit for the partition
- **Free Page**—the page increment for inserting a free page
- **Pct Free**—the percentage of free space to leave in each nonleaf page
- **Gbp Cach**—the group buffer pool cache setting for the partition
- **Comp**—the compression attribute for the partition
- **SMS Storclass**—for VCAT partitions, the SMS storage class for the partition
- **SMS Dataclass**—for VCAT partitions, the SMS data class for the partition
- **SMS Mgmtclass**—for VCAT partitions, the SMS management class for the partition
- **Track Mod**—the indicator for tracking page modifications in the space map pages

7 Repeat step 5 and step 6 for each partition that you want to create.
To specify limit keys for each partition

1. On the Tablespace Parts List panel, type LK in the Act column adjacent to the existing partition. Then, press Enter.

   The Limit Key Detail panel is displayed.

2. Type a value for New Limitkey.

3. Press END until the Table Detail panel is displayed.

   The Table Detail panel now displays the number of partitions, and the type of partitioning that is used.

4. Press END until the Mixed List panel is displayed.

5. On the Mixed List panel, drop or move all of the tables except the table that you want to convert.

   - To drop a table, type D in the Act column adjacent to the table and table’s dependent objects.

   - To move a table, type E in the Act column adjacent to the table, and specify a different table space name.

6. Press END until the WORKID Action Menu is displayed.

   The table will be converted to table-controlled partitioning when you execute the worklist.

To undo changes to the table

If you wanted to undo the changes to the table, you must also undo the changes to the table space. In the Mixed List panel, type one of the following commands:

- To undo changes to all of the objects listed, type UNDOALL on the Command line.

- To undo changes to one or more of the objects listed, type U in the Act column adjacent to each object
Where to go from here

After you convert the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting a nonpartitioned table space to a range-partitioned table space

The steps that you complete to convert a nonpartitioned table space to a range-partitioned table space are similar to the steps for converting a nonpartitioned table space to a table-controlled partitioned table space.

1 Follow the steps in “Converting a nonpartitioned table space to a table-controlled partitioned table space” on page 486.

2 On the Mixed List panel, drop or move all of the tables except the table that you want to convert.
   - To drop a table, type D in the Act column adjacent to the table and table’s dependent objects.
   - To move a table, type E in the Act column adjacent to the table, and specify a different table space name.

3 On the Tablespace Detail panel, verify that the value in the Segsize field is greater than 0.

Where to go from here

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting a nonpartitioned table space to a partition-by-growth table space

In this task, you will

1. display the list of nonpartitioned table spaces
2. move additional tables from the table space
3. convert to an explicit table space

**To display the list of nonpartitioned table spaces**

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of an alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a nonpartitioned table space (or use wildcard characters) and press Enter.

   The Mixed List panel is displayed.

**To move additional tables**

If multiple tables exist in the table space, move all of the tables except the table that you want to convert. To move a table on the Mixed List panel, type E in the Act column adjacent to the table, and specify a different table space name.

**To convert to an explicit table space**

1. Type E in the Act column adjacent to a table space that you want to convert.

   The Tablespace Detail panel is displayed.

2. Type S to select Tablespace Partition List.
Converting a nonpartitioned table space to a partition-by-growth table space

3 On the Tablespace Parts List panel, perform the following actions:

A For the partitioning status, select **Partition by Growth**.

A dialog is displayed (**Figure 144**).

**NOTE**
The messages that are displayed in the dialog vary, depending on the values that you have set for the DSSIZE and SEGSIZE.

**Figure 144** Partition-by-growth dialog

![Figure 144 Partition-by-growth dialog]

B To have the product perform the following actions, press **END**:

- Set the value of DSSIZE to 4 GB.
- Set the value of SEGSIZE to 4.

**TIP**
You can change the value of DSSIZE and SEGSIZE on the Tablespace Parts List panel.

C In the **Maxpartitions** field, type the maximum number of partitions for the partition-by-growth table space.

D Press **END** until the Mixed List panel is displayed.
Where to go from here

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting a partitioned table space to a nonpartitioned table space

In this task, you can convert an index-controlled partitioned, table-controlled partitioned, range-partitioned, or partition-by-growth table space to a nonpartitioned table space.

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.

   The WORKID Action Menu is displayed.

2. Type the name of an alter-type **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.

   The Object Specification panel is displayed.

3. Specify the name of a partitioned table space (or use wildcard characters). Then, press **Enter**.

   The Mixed List panel is displayed.

4. Type **E** in the **Act** column adjacent to a table space that you want to convert.

   The Tablespace Detail panel is displayed.

5. Type **S** to select **Tablespace Partition List**.

   The Tablespace Parts List panel is displayed.
Converting a partitioned table space to a nonpartitioned table space

6 On the Tablespace Parts List panel, perform the following actions:

A For the partitioning status, select **Non-Partitioned**.

B In the Dssize field, type 0.

C If you are converting from a partition-by-growth table space, in the Maxpartitions field, type 0.

D Drop all but one of the table space partitions.

E Press END until the Mixed List panel is displayed.

7 On the Mixed List panel, type E in the Act column adjacent to a partitioned index.

The Index Detail panel is displayed.

8 On the Index Detail panel, type S to select **Partitions**.

The Index Parts List panel is displayed.

9 On the Index Parts List panel, perform the following actions:

A For the partitioning status, select **Non-Partitioned**.

B Drop all but one of the index partitions.

C Press END.

**TIP**

If the table space that you are converting is in an implicit database, you must explicitly change the name of the database. On the Table Detail panel, change the database name to DSNDB04 or specify a user-defined name. The table space must exist already in the DB2 catalog, or you must create the table space in the work ID.

**Where to go from here**

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist</strong>.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in <strong>CHANGE MANAGER</strong>” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis</strong>.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Changing a partitioned table space to a range-partitioned table space by importing DML

You can import DML to change table-controlled or index-controlled partitioned table spaces to range-partitioned table spaces.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. In an ISPF editor, type the appropriate DML statements and provide the information that is unique to your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

   - To change existing table-controlled partitioned table spaces to range-partitioned table spaces, type the DML shown in Figure 145.

   **Figure 145  DML to change table-controlled partitioned table spaces to range-partitioned table spaces**

   ```sql
   UPDATE TABLESPACE
   SET
   SEGSIZE = 32  --a value > 0 indicates range partitioning
   WHERE
   DBNAME LIKE 'DEMTCP%'  --specify existing DB name; table spaces in this database use table-controlled partitioning
   ;
   ```

   - To change table-controlled table spaces by using a JOIN statement, type the DML shown in Figure 146. The WHERE clause in the DML statement selects the table spaces that contain a table with a value for the PARTKEYCOLNUM that is greater than 0. This value indicates that the table space uses table-controlled partitioning.

   **Figure 146  DML to change table-controlled partitioned table spaces to range-partitioned table space by using a JOIN (part 1 of 2)**

   ```sql
   UPDATE TABLESPACE
   JOIN SYSIBM.SYSTABLES TBLS
   SET
   SEGSIZE = 32  --a value > 0 indicates range partitioning
   WHERE
   SYSIBM.SYSTABLESPACE.DBNAME LIKE 'DEMO%'  --specify existing DB name
   ```
Changing a partitioned table space to a range-partitioned table space by importing DML

To change index-controlled partitioned table spaces to range-partitioned table spaces and to convert the index-controlled partitioned table spaces to table-controlled table spaces, type the DML shown in Figure 147.

**Figure 147  DML to change index-controlled partitioned table spaces to range-partitioned table spaces**

```sql
UPDATE TABLESPACE
SET
  SEGSIZE = 32          -- a value > 0 indicates range partitioning
WHERE
  DBNAME LIKE 'DEMICPA%'
  -- specify existing DB name
;
UPDATE TABLES
SET
  TCPART = 'Y'          -- converts index-controlled partitioning
  -- to table-controlled partitioning
WHERE
  DBNAME LIKE 'DEMICPA%'
  -- specify existing DB name
;
```

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting a table-controlled partitioned table space to an index-controlled partitioned table space

This task describes how you can convert a table in a partitioned table space that uses table-controlled partitioning to a table space that uses index-controlled partitioning. To accomplish this task, the product must generate a worklist that drops and creates the table space and the supporting objects.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of an alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a partitioned table space that uses table-controlled partitioning (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Verify that an index for the table is be defined as CLUSTER and partitioned, and that the key columns in the index match the partitioning key columns.

5. Type E in the Act column adjacent to a table that you want to convert to index-controlled partitioning.

   The Table Detail panel is displayed.

6. Type S to select Convert to Index Controlled Partitioning.

7. Press END until the Mixed List panel is displayed.

   In the Mixed List panel, an asterisk (*) and a change-level indicator are displayed adjacent to the table space, table, and index.

8. Press END until the WORKID Action Menu is displayed.

   The table will be converted to index-controlled partitioning when you execute the worklist.

   **TIP**

   If you are converting a range-partitioned table space to index-controlled partitioning, edit the table space and change the SEGSIZE to 0.
To undo changes to the table

If you wanted to undo the changes to the table, you must also undo the changes to the table space. In the Mixed List panel, type one of the following commands:

- To undo changes to all of the objects listed, type UNDOALL on the Command line.
- To undo changes to one or more of the objects listed, type U in the Act column adjacent to each object.

Where to go from here

After you convert the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting table-controlled partitioned table spaces to range-partitioned table spaces by importing DML

You can import DML to convert table-controlled partitioned table spaces to range-partitioned table spaces.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. Copy the DML statements from member ACMDMMLUB in the HLQ.DBCNTL data set.

2. Edit one of the DML statements (Figure 148) in an ISPF editor. Provide the information that is required for your site, and delete the other statement.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.
Chapter 4 Altering data structures in a database environment

Converting table-controlled partitioned table spaces to range-partitioned table spaces by importing DML

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Figure 148 DML to convert to range-partitioned table spaces

```
-- The first example performs the conversion by changing the
-- SEGSIZE of the table space to 32. The scope of the WHERE
-- clause includes all of the table-controlled partitioned
-- table spaces.
--
UPDATE TABLESPACE
SET SEGSIZE = 32
WHERE
  DBNAME LIKE 'DEMTCP%'
;
--
-- The second example also performs the conversion by changing
-- the SEGSIZE of the table space to 32. The scope of the WHERE
-- clause includes only the table spaces that contain
-- a table in which the PARTKEYCOLNUM is greater than 0. This
-- condition indicates that a table space uses table-controlled
-- partitioning.
--
UPDATE TABLESPACE
JOIN SYSIBM.SYSTABLES TBLS
SET
  SEGSIZE = 32
WHERE
  SYSIBM.SYSTABLESPACE.DBNAME LIKE 'DEMO%' AND
  TBLS.PARTKEYCOLNUM > 0 AND
  TBLS.DBNAME = SYSIBM.SYSTABLESPACE.DBNAME AND
  TBLS.TSNAME = SYSIBM.SYSTABLESPACE.NAME
;
```
Converting a range-partitioned table space to an index-controlled partitioned table space

The steps that you complete to convert a range-partitioned table space to an index-controlled partitioned table space are similar to the steps for converting a table-controlled partitioned table space to an index-controlled partitioned table space.

1. Follow the steps in “Converting a table-controlled partitioned table space to an index-controlled partitioned table space” on page 497.

2. On the Mixed List panel, type E in the Act column adjacent to the converted index-controlled partitioned table space.

   The Tablespace Detail panel is displayed.

3. On the Tablespace Detail panel, type 0 in the Segsize field.

Where to go from here

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting an index-controlled partitioned table space to a table-controlled partitioned table space

In this task, you can convert a table in a partitioned table space that uses index-controlled partitioning to a table space that uses table-controlled partitioning.

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of an alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.
3 Specify the name of a partitioned table space that uses index-controlled partitioning (or use wildcard characters) and the name of a table. Then, press Enter.

The Mixed List panel is displayed.

4 Type E in the Act column adjacent to a table that you want to convert to table-controlled partitioning.

The Table Detail panel is displayed.

5 Type S to select Convert to Table Controlled Partitioning.

One of the following panels is displayed:

- If one of the following conditions exists, the Table Detail panel is displayed, as shown in Figure 150:
  - The table space is defined as LARGE.
  - The DSSIZE for the table space is greater than zero.
  - The table contains a large object (LOB) column.

Go to step 7 on page 502.

- Otherwise, the Preserve Limitkey Values panel is displayed, as shown in Figure 149.

Figure 149  Preserve Limitkey Values panel
6 On the Preserve Limitkey Values panel choose whether to have the product preserve the limit key values of the last partition.

A Type Y or N at the prompt:

- Type Y to have the product preserve the values of the limit keys.
- Type N to have DB2 automatically assign MAXVALUE or MINVALUE (or high or low values) to the limit keys.

For more information about preserving limit keys, see “Preserving the values of limit keys” on page 323.

B Press END to exit the panel.

The Table Detail panel is displayed, as shown in Figure 150.

Figure 150 Table Detail panel

7 On the Table Detail panel, the product now provides the following selections:

- Partitioning Key Columns
- Parts List and LimitKey Values

Use these options to verify or modify the partitioning key columns and limit key values for the partitions.
Converting an index-controlled partitioned table space to a table-controlled partitioned table space

A To verify the partitioning key columns, type S to select Partitioning Key Columns.

The Partitioning Key Mixed List is displayed.

B To verify the limit key values, type S to select Parts List and LimitKey Values.

The Tablespace Parts List is displayed. Scroll to the right to view the values of the limit keys.

8 Press END until the Mixed List panel is displayed.

In the Mixed List panel, an asterisk (*) and a change-level indicator are displayed adjacent to the table space and the table.

9 Press END until the WORKID Action Menu is displayed.

The table will be converted to table-controlled partitioning when you execute the worklist.

Considerations for converting to table-controlled partitioning

When you convert a table in a nonpartitioned table space to table-controlled partitioning, consider the following items:

■ If you want to rename your partitioned table space, do so after you convert a table in the table space to table-controlled partitioning.

■ If you want to rename the table that you converted to table-controlled partitioning, do so after you create any indexes on the table.

■ If you want to make changes to LIKEd objects (such as table spaces, tables, or indexes), do so after you convert a table in the table space to table-controlled partitioning.

■ If you want to undo the changes to the table, you must also undo the changes to the table space. In the Mixed List panel, type one of the following commands:

— To undo changes to all of the objects listed, type UNDOALL on the Command line.

— To undo changes to one or more of the objects listed, type U in the Act column adjacent to each object.
Where to go from here

After you convert the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting index-controlled partitioned table spaces to table-controlled partitioned table spaces by importing DML

You can import DML to convert index-controlled partitioned table spaces to table-controlled partitioned table spaces.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 Copy the DML statement from member ACMDMLU9 in the HLQ.DBCNTL data set.

2 Edit the DML statement (**Figure 151**) in an ISPF editor. Provide the information that is required for your site.

   For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

   **Figure 151  DML to convert to table-controlled partitioned table spaces**

   ```sql
   UPDATE TABLES
   SET TCPART  = 'Y'
   WHERE DBNAME LIKE 'DEMICPA%'
   ;
   ```

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting an index-controlled partitioned table space to a range-partitioned table space

The steps that you complete to convert an index-controlled partitioned table space to a range-partitioned table space are similar to the steps for converting an index-controlled partitioned table space to a table-controlled partitioned table space.

1 Follow the steps in “Converting an index-controlled partitioned table space to a table-controlled partitioned table space” on page 500.

2 On the Mixed List panel, type **E** in the **Act** column adjacent to the converted table-controlled partitioned table space.

   The Tablespace Detail panel is displayed.

3 In the **Segsize** field, type a value greater than 0.

Where to go from here

After you convert the table, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting index-controlled partitioned table spaces to range-partitioned table spaces by importing DML

You can import DML to convert index-controlled partitioned table spaces to range-partitioned table spaces. The first DML statement modifies the SEGSIZE of the table space. The second DML statement converts all of the tables from index-controlled partitioned to table-controlled partitioned. The scope of the WHERE clause is the same for both statements.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. Copy the DML statements from member ACMDMLUA in the HLQ.DBCNTL data set.

2. Edit the DML statements (Figure 152) in an ISPF editor. Provide the information that is required for your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

**Figure 152** DML to convert to range-partitioned table spaces

```
UPDATE TABLESPACE
  SET
    SEGSIZE = 32
  WHERE
    DBNAME LIKE 'DEMICPA%'
;

UPDATE TABLES
  SET
    TCPART  = 'Y'
  WHERE
    DBNAME LIKE 'DEMICPA%'
;
```

3. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.
Converting a partitioned table space to a partition-by-growth table space

You can convert one of the following objects to a partition-by-growth table space:

- index-controlled partitioned table space
- table-controlled partitioned table space
- range-partitioned table space

In this task, you will

1. Display the list of partitioned table spaces.
2. Convert to an explicit table space, or to an implicit database or table space.
3. Change the partitioned indexes.

To display the list of partitioned table spaces

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of an alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a partitioned table space that uses index-controlled or table-controlled partitioning (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting a partitioned table space to a partition-by-growth table space

**To convert to an explicit table space**

1. Type **E** in the **Act** column adjacent to a table space that you want to convert.

   **TIP**
   
   The TSI object type indicates that the object is an index-controlled partitioned table space. The TST object type indicates that the object is a table-controlled partitioned table space or a range-partitioned table space.

   The Tablespace Detail panel is displayed.

2. Type **S** to select **Tablespace Partition List**.

3. On the Tablespace Parts List panel, perform the following actions:

   **A** For the partitioning status, select **Partition by Growth**.

   A dialog is displayed (Figure 153).

   **NOTE**
   
   The messages that are displayed in the dialog vary, depending on the values that you have set for the DSSIZE and SEGSIZE, and whether you are converting from a range-partitioned table space.

---

**Figure 153** Partition-by-growth dialog

<table>
<thead>
<tr>
<th>ALUSTSG R</th>
<th>Tablespace Parts List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command ===</td>
<td>Scroll. . CSR</td>
</tr>
<tr>
<td>WORKID . . . : RDACRJ.ICPPBG</td>
<td></td>
</tr>
<tr>
<td>Database Name . . : C9AIP1</td>
<td></td>
</tr>
</tbody>
</table>

Select partitioning status.

| ALUSTSG |
| Command === |

You have requested that this Tablespace be converted to Partition by Growth. The following actions must be taken prior to conversion:

* All but 1 partition must be dropped
* DSSize must have a non-zero value (Default is 4G)
* Segsize must have a non-zero value (Default is 4)

Would you like for the product to perform these actions for you?  
**Y** (Y-Yes, N-No)

Commands: **END, CANCEL**

---
Converting a partitioned table space to a partition-by-growth table space

B To have the product perform the following actions, press END:

- Delete all of the partitions except the first partition.
- Set the value of DSSIZE to 4 GB.
- Set the value of SEGSIZE to 4.

**TIP**
You can change the value of DSSIZE and SEGSIZE on the Tablespace Parts List panel.

C In the Maxpartitions field, type the maximum number of partitions for the partition-by-growth table space.

D Press END.

**To convert to an implicit database or table space**

If you are converting an index-controlled partitioned table space, perform one of the following steps:

- To make the database implicit, edit the table in the table space and change the name of the database to <DEFLT>. The table space name will automatically change to <DEFLT>.

- To make only the table space implicit, edit the table in the table space and change the name of the table space to <DEFLT>.

**To change the partitioned indexes**

If partitioned indexes exist in the table space, you must convert the indexes to nonpartitioned indexes.

1 From the Mixed List, type E in the Act column adjacent to an index that you want to convert.

   The Index Detail panel is displayed.

2 Type S to select Partitions.

   The Index Parts List is displayed.

3 In the Act column, type D adjacent to all of the partitions except the first one and press Enter.

4 For the partitioning status, select Non-Partitioned.
Converting a partition-by-growth table space to an index-controlled partitioned table space

5 Press END.

Where to go from here

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Converting a partition-by-growth table space to an index-controlled partitioned table space

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of an alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3 Specify the name of a partition-by-growth table space (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4 Type E in the Act column adjacent to the partition-by-growth table space.

   **TIP**

   The object type TSG indicates that the object is a partition-by-growth table space.

   The Tablespace Detail panel is displayed.

5 Type S to select Tablespace Partition List.
6 On the Tablespace Parts List panel, perform the following actions:

A For the partitioning status, select **Regular Partitioned**.

B In the **Segsize** field, type 0.

C In the **Maxpartitions** field, type 0.

D Press END.

The Tablespace Detail panel is displayed.

7 Create additional partitions for the table space and create a clustering partitioning index.

For information, see “Converting a nonpartitioned table space to an index-controlled partitioned table space” on page 480.

8 Press END until the WORKID Action Menu is displayed.

---

**TIP**

If the table space that you are converting is in an implicit database, you must explicitly change the name of the database. On the Table Detail panel, change the database name to DSNDB04 or specify a user-defined name. The table space must exist already in the DB2 catalog, or you must create the table space in the work ID.

---

**Where to go from here**

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting a partition-by-growth table space to a table-controlled partitioned table space

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of an alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name of a partition-by-growth table space (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

4 Type E in the Act column adjacent to a table in the table space.

The Table Detail panel is displayed.

5 Type S to select Convert to Table Controlled Partitioning.

6 Press Enter.

7 Specify partitioning key columns, add partitions to the table space, and specify limit keys for each partition.

For information, see “Converting a nonpartitioned table space to a table-controlled partitioned table space” on page 486.

8 Type E in the Act column adjacent to the partition-by-growth table space.

9 Type S to select Tablespace Partition List.

10 On the Tablespace Parts List panel, type 0 in the Segsize field.

TIP

The object type TSG indicates that the object is a partition-by-growth table space.
Converting a partition-by-growth table space to a range-partitioned table space

11 Type 0 in the Maxpartitions field.

12 Press END until the WORKID Action Menu is displayed.

---

**TIP**

If the table space that you are converting is in an implicit database, you must explicitly change the name of the database. On the Table Detail panel, change the database name to DSNDB04 or specify a user-defined name. The table space must exist already in the DB2 catalog, or you must create the table space in the work ID.

---

**Where to go from here**

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

---

**Converting a partition-by-growth table space to a range-partitioned table space**

The steps that you complete to convert a partition-by-growth table space to a range-partitioned table space are similar to the steps for converting a partition-by-growth table space to a table-controlled partitioned table space.

1 Follow the steps in “Converting a partition-by-growth table space to a table-controlled partitioned table space” on page 512.

2 On the Mixed List panel, type E in the Act column adjacent to the converted range-partitioned table space.

The Tablespace Detail panel is displayed.
3 On the Tablespace Detail panel, verify that the value in the Segsize field is greater than 0.

TIP
If the table space that you are converting is in an implicit database, you must explicitly change the name of the database. On the Table Detail panel, change the database name to DSNDB04 or specify a user-defined name. The table space must exist already in the DB2 catalog, or you must create the table space in the work ID.

Where to go from here

After you convert the table space, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Changing a single-table table space to a partition-by-growth table space by importing DML

You can import DML to change table-controlled or index-controlled partitioned table spaces or other single-table table spaces to partition-by-growth table spaces. CM/PILOT automatically deletes all but the first partition of a partitioned table space and any dependent indexes.

NOTE
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the appropriate DML statement (Figure 154) and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 154 DML to change a table space to a partition-by-growth table space (part 1 of 2)
Changing tables in an explicit database to an implicit database by importing DML

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Figure 154  DML to change a table space to a partition-by-growth table space (part 2 of 2)

```
MAXPARTITIONS = 10
WHERE
  DBNAME  = 'MG025'
  AND    NTABLES = 1
;
```

Changing tables in an explicit database to an implicit database by importing DML

You can import DML to change tables in an explicit database to an implicit database.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the appropriate DML statement and provide the information that is unique to your site.

   For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

   - To change tables from an explicit database to an implicit database, type the DML shown in Figure 155.
Changing tables in an explicit database to an implicit database by importing DML

To change tables from an explicit database that includes a table space with more than one table to an implicit database, type the DML shown in Figure 156. The second DML statement deletes the table spaces that contain more than one table.

**Figure 155  DML to change tables from an explicit database to an implicit database**

```
UPDATE TABLES
SET       
  DBNAME = ''
WHERE     
  DBNAME LIKE 'MG025%'
;
```

![Figure 155 DML to change tables from an explicit database to an implicit database](image)

To change tables from an explicit database that includes a table space with more than one table to an implicit database, type the DML shown in Figure 156. The second DML statement deletes the table spaces that contain more than one table.

**Figure 156  DML to change tables from an explicit database to an implicit database**

```
UPDATE TABLES
JOIN SYSIBM.SYSTABLESPACE TS
SET       
  DBNAME = ''
WHERE     
  TS.DBNAME LIKE 'MG%'
  AND TS.DBNAME = SYSIBM.SYSTABLES.DBNAME
  AND TS.NAME = SYSIBM.SYSTABLES.TSNAME
  AND TS.NTABLES > 1
;
DELETE TABLESPACE
WHERE      
  DBNAME LIKE 'MG%'
  AND NTABLES > 1
;
```

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting explicit databases and implicit table spaces to implicit databases and table spaces by importing DML

You can import DML to convert explicit databases and implicit table spaces to implicit databases and table spaces.

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. Copy the DML statements from member ACMDMLUD in the HLQ.DBCNTL data set.

2. Edit one of the DML statements (Figure 157) in an ISPF editor. Provide the information that is required for your site. Delete the other statement.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

**Figure 157  DML to convert to implicit databases and table spaces (part 1 of 2)**

```
-- The first example converts the explicit databases to implicit 
-- databases by setting the DBNAME of the tables in the databases 
-- to blanks.
--
UPDATE TABLES
SET
  DBNAME = ''
WHERE
  DBNAME LIKE 'MG025'
;
--
-- The first statement in the second example converts the 
-- tables in multi-table table spaces to implicit databases. 
-- The second DML statement deletes the table spaces that 
-- include more than one table.
--
UPDATE TABLES
JOIN SYSIBM.SYSTABLESPACE TS
SET
  DBNAME = ''
WHERE
  TS.DBNAME LIKE 'MG' AND 
  TS.DBNAME = SYSIBM.SYSTABLES.DBNAME AND 
  TS.NAME = SYSIBM.SYSTABLES.TSNAME AND 
  TS.NTABLES > 1
;
DELETE TABLESPACE
```
Changing the encoding scheme for a table space by importing DML

You can import DML to change the encoding scheme for a table space.

NOTE

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 Copy the DML statements from member ACMDMLUF in the HLQ.DBCNTL data set.

2 Edit one of the DML statements (Figure 157) in an ISPF editor. Provide the information that is required for your site. Delete the other statement.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 158 DML to change the encoding scheme for a table space (part 1 of 2)

```
-- DML in this file changes the encoding scheme for a table space
-- to UNICODE.
```
Adding a ROWID and a LOB column to create a base table

You can add a ROWID column and a LOB column to an existing table to create a base table. You can also create a unique index for a ROWID column that is defined as GENERATED BY DEFAULT. A base table must exist before you can create auxiliary objects.

In this task, you will

1. display the list of columns
2. create a ROWID column
3. create a LOB column
4. create a unique index for the ROWID column
5. create the auxiliary objects for the base table

Figure 158  DML to change the encoding scheme for a table space (part 2 of 2)

<table>
<thead>
<tr>
<th>UPDATE TABLESPACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET</td>
</tr>
<tr>
<td>ENCODING_SCHEME = 'U' -- sets ENCODING_SCHEME to UNICODE</td>
</tr>
<tr>
<td>WHERE</td>
</tr>
<tr>
<td>NAME = 'TS04P' AND</td>
</tr>
<tr>
<td>DBNAME = 'DEMO105G'</td>
</tr>
<tr>
<td>;</td>
</tr>
</tbody>
</table>

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Adding a ROWID and a LOB column to create a base table

**To display the list of columns**

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3. Specify the name of a table space (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4. Type CO in the Act column adjacent to a table.

   The Table Columns List panel is displayed.

**To create a ROWID column**

1. From the Table Columns List panel, type I in the Act column adjacent to a column name. Then, press Enter.

   A blank line is inserted after the line on which you typed I.

   **NOTE**

   You can also type L in the Act column to duplicate an existing column definition. You can then edit the attributes of the column by typing E in the ACT column adjacent to the column that you want to modify.

2. Specify the attributes for the ROWID column.

   **TIP**

   Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

   A. Specify the name of the ROWID column.

   B. Specify ROWID as the column type.

   C. Type N in the NI column to indicate that a null value is not allowed.

   D. Type D in the DF column to indicate that the column is defined as GENERATED BY DEFAULT.
Adding a ROWID and a LOB column to create a base table

NOTE
If the ROWID column is defined as GENERATED BY DEFAULT, a unique index is required for the column.

E Press Enter.

To create a LOB column

1 From the Table Columns List panel, type I in the Act column adjacent to a column name. Then, press Enter.

A blank line is inserted after the line on which you typed I.

NOTE
You can also type L in the Act column to duplicate an existing column definition. You can then edit the attributes of the column by typing E in the ACT column adjacent to the column that you want to modify.

2 Specify the attributes for the LOB column.

TIP
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Specify the name of the LOB column.

B Specify BLOB, CLOB, or DBCLOB as the column type.

C Specify the length of the column.

D Specify whether null values are allowed.

E Specify whether the column has a default value.

F Press Enter.

3 Press END.

The Mixed List panel is displayed. The table is now a base table with an object type of TBB.
To create a unique index for the ROWID column

If the ROWID column is defined as GENERATED BY DEFAULT, a unique index is required for the column.

1. From the Mixed List panel, type CIX in the Act column adjacent to the base table.

   The Create Index panel is displayed.

2. Specify the attributes for the index.

   **TIP**
   Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

   - **A** Specify the owner of the index.
   - **B** Specify the name of the index.
   - **C** Type U to indicate that the index is unique.
   - **D** Specify whether the parent table already exists in the DB2 catalog.
   - **E** Type S to select Keys and TB Cols.
   - **F** Press Enter.

   The Index Key Mixed List panel is displayed, as shown in Figure 159.
Adding a ROWID and a LOB column to create a base table

Figure 159  Index Key Mixed List panel

<table>
<thead>
<tr>
<th>Act</th>
<th>Colname</th>
<th>Column Type</th>
<th>UDT Length</th>
<th>Nl</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>INDEX KEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IXG_FOR_ROWID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>ROW_ID</td>
<td>ROWID</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>OTRACOL</td>
<td>CHAR</td>
<td></td>
<td>5</td>
<td>N</td>
</tr>
<tr>
<td>L</td>
<td>CLOB</td>
<td>CLOB</td>
<td></td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

3 Specify the index key.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Type C in the Act column adjacent to the ROWID column name.

B Type A in the Act column adjacent to the Index Key.

C Press Enter.

The Index Key Mixed List panel is displayed as shown in Figure 160. The index key column is displayed below the index key.
To create the auxiliary objects for the base table

To create the auxiliary objects for the base table, perform one of the following tasks:

- To create objects in a partitioned table space, see “Creating auxiliary objects for a partitioned table space” on page 525.

- To create objects in a nonpartitioned table space, see “Creating auxiliary objects for a nonpartitioned table space” on page 538.
Where to go from here

After you create the base table and the auxiliary objects, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating auxiliary objects for a partitioned table space

You can create auxiliary objects for an existing base table in a partitioned table space. The CAX action code is used to create the auxiliary table spaces, tables, and indexes that are required for a base table that contains a LOB column. In this task, name prefixing is used in a template to automatically and uniquely name the auxiliary objects.

In this task, you will

1. display the list of columns
2. create the auxiliary table spaces
3. create the auxiliary tables
4. create the auxiliary indexes
5. review the list of auxiliary objects
6. (optional) undo changes to the auxiliary objects

Before you begin

If you want to increase the number of partitions in the partitioned table space, do so before you begin this task.

To display the list of columns

1  On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.
2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

3 Specify the name of a partitioned table space (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

4 Type CO in the Act column adjacent to the base table (TBB) that contains a LOB column.

The Table Columns List panel is displayed.

5 Type CAX in the Act column adjacent to the LOB column name and press Enter.

The Create Auxiliary Tablespace Template panel is displayed.

To create the auxiliary table spaces

1 From the Create Auxiliary Tablespace Template panel, select Use name prefixing.

**NOTE**
Name prefixing will also be used for the auxiliary tables and indexes.

2 Specify the attributes for the table spaces.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Specify a prefix for the name of the table spaces. You can retain the first byte (L) and replace the three question mark (?) characters, or replace all four of the characters. Underscores are not allowed.

**NOTE**
If you use name prefixing, the prefix is limited to a maximum of four characters. If you do not use name prefixing, you must specify a name (up to eight characters) for the table spaces.

B Specify the owner of the table space.
Creating auxiliary objects for a partitioned table space

C Specify the attributes for the table spaces. If you need to specify additional attributes, type S to select More Tablespace Attributes.

D Specify whether the parent database already exists in the DB2 catalog.

E Press END.

The Auxiliary Tablespace List panel is displayed. A unique four-digit suffix is appended to the table space name prefix for each table space.

3 If necessary, modify the names of the auxiliary table spaces. Then, press END.

**NOTE**
If you did not use name prefixing, you must make the names of the table spaces unique.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

The table spaces are saved in the CD tables. The Create Auxiliary Table Template panel is displayed.

**To create the auxiliary tables**

1 From the Create Auxiliary Table Template panel, specify the attributes for the tables.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Specify the owner of the tables.

B Specify a prefix for the name of the tables.

**NOTE**
Because you chose to use name prefixing for the auxiliary table spaces, you must use name prefixing for the auxiliary tables.

If you do not delete the underscores at the end of the name of the tables, the underscores are included as part of the name.
C Specify a label for the tables.

D Specify whether the parent (auxiliary) table space already exists in the DB2 catalog.

---

**NOTE**

In this example, the auxiliary table space does not exist.

---

E To specify a comment for the tables, type S to select **Table Comment**.

---

**TIP**

Some object names might be too long to be displayed on a panel. To enter a long object name, position the cursor on the object name and press the ZOOM (F4) key.

---

F Press END.

The Auxiliary Table List panel is displayed.

2 If necessary, modify the names of the auxiliary tables. Then, press END.

---

**NOTE**

If you did not use name prefixing, you must make the names of the tables unique.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

---

The tables are saved in the CD tables. The Create Auxiliary Index Template panel is displayed.

**To create the auxiliary indexes**

1 From the Create Auxiliary Index Template panel, specify the attributes for the indexes.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.
A Specify the owner of the indexes.

B Specify a prefix for the name of the indexes.

**NOTE**
Because you chose to use name prefixing for the auxiliary table spaces, you must use name prefixing for the auxiliary indexes.

If you do not delete the underscores at the end of the name of the indexes, the underscores are included as part of the name.

C Specify values for the attributes. To specify additional attributes, type S to select **More Attributes**.

D Specify whether the parent (auxiliary) table already exists in the DB2 catalog.

**NOTE**
In this example, the auxiliary table does not exist.

E To specify a comment for the indexes, type S to select **Comment**.

F Press END.

The Auxiliary Index List panel is displayed.

2 If necessary, modify the names of the auxiliary indexes. Then, press END.

**NOTE**
If you did not use name prefixing, you must make the names of the indexes unique.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.

The indexes are saved in the CD tables. The Table Columns List panel is displayed.
Creating auxiliary objects for a partitioned table space

To review the list of auxiliary objects

1. From the Table Columns List panel, type AX in the Act column adjacent to the LOB column name to review the list of auxiliary objects for the base table.

   **TIP**

   Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

The Auxiliary Objects List panel is displayed. The auxiliary table space, table, and index for each partition are listed by the LOB column and partition, as shown in Figure 161.

**Figure 161  Auxiliary Objects List—partitioned table space**

2. Press END.

   The Table Columns List panel is displayed.

3. Press END.

   The Mixed List panel is displayed. The *@ marker adjacent to the base table (TBB) object indicates that auxiliary objects were created or modified for the base table.

4. Press END until the WORKID Action Menu is displayed.
Creating auxiliary objects for a single partition in a partitioned table space

To undo changes to auxiliary objects

If you wanted to undo the changes to the base table, you must undo the changes to the auxiliary objects. In the Mixed List panel, type AX in the Act column adjacent to the modified base table (TBB). In the Auxiliary Objects List panel, remove the requested changes to the auxiliary objects.

- To undo changes to all of the objects listed, type UNDOALL on the Command line.
- To undo changes to one or more of the objects listed, type U in the Act column adjacent to the auxiliary objects.

Where to go from here

After you create the auxiliary objects, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating auxiliary objects for a single partition in a partitioned table space

You can create a set of auxiliary objects for a single partition in a partitioned table space. The CAX action code is used to create the auxiliary table space, table, and index that are required for an existing base table that contains a LOB column.

In this task, you will

1. specify a template
2. create the auxiliary table space
3. create the auxiliary table
4. create the auxiliary index
5. review the list of auxiliary objects
6. (optional) undo changes to the auxiliary objects
Creating auxiliary objects for a single partition in a partitioned table space

**To specify a template**

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.
   The WORKID Action Menu is displayed.

2. Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.
   The Object Specification panel is displayed.

3. Specify the name of a partitioned table space (or use wildcard characters). Then, press **Enter**.
   The Mixed List panel is displayed.

4. Type **CO** in the **Act** column adjacent to the base table (TBB) that contains a LOB column.
   The Table Columns List panel is displayed.

5. Type **CAX** in the **Act** column adjacent to the LOB column name and press **Enter**.
   The Create Auxiliary Tablespace Template panel is displayed.

**To create the auxiliary table space**

1. From the Create Auxiliary Tablespace Template panel, select **Create single object**.

2. Specify the attributes for the table space.

   **TIP**
   Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.

A. Specify the name of the table space.

B. Specify the owner of the table space.

C. Specify the attributes for the table space. To specify additional attributes, type **S** to select **More Tablespace Attributes**.

D. Specify whether the parent database already exists in the DB2 catalog.
E Press END.

The auxiliary table space is saved in the CD tables. The Create Auxiliary Table panel is displayed.

To create the auxiliary table

From the Create Auxiliary Table panel, specify the attributes for the table.

--- TIP ---
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

1 Specify the owner of the table.
2 Specify the name of the table.
3 Specify the name of the table space.
4 Specify a label for the table.
5 Specify whether the parent (auxiliary) table space already exists in the DB2 catalog.

--- NOTE ---
In this example, the auxiliary table space does not exist.

6 Specify the partition number for which you want to create the auxiliary table.
7 To specify a comment for the table, type S to select Table Comment.

--- TIP ---
Some object names might be too long to be displayed on a panel. To enter a long object name, position the cursor on the object name and press the ZOOM (F4) key.

8 Press END.

The auxiliary table is saved in the CD tables. The Create Auxiliary Index panel is displayed.
Creating auxiliary objects for a single partition in a partitioned table space

**To create the auxiliary index**

From the Create Auxiliary Index panel, specify the attributes for the index.

---

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

---

1. Specify the owner of the index.
2. Specify the name of the index.
3. Specify the owner of the table.
4. Specify the name of the table.
5. Specify values for the attributes. To specify additional attributes, type S to select More Attributes.
6. Specify whether the parent (auxiliary) table already exists in the DB2 catalog.

---

**NOTE**

In this example, the auxiliary table does not exist.

---

7. To specify a comment for the index, type S to select Comment.
8. Press END.

The auxiliary index is saved in the CD tables. The Table Columns List panel is displayed.

---

**To review the list of auxiliary objects**

1. From the Table Columns List panel, type AX in the Act column adjacent to the LOB column name to review the list of auxiliary objects for the base table.

The Auxiliary Objects List panel is displayed. The auxiliary table space, table, and index for each partition are listed for the partition that you specified, as shown in Figure 162.
Creating auxiliary objects for a single partition in a partitioned table space

TIP

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

Figure 162  Auxiliary Objects List—single partition

<table>
<thead>
<tr>
<th>Act</th>
<th>Object-Type</th>
<th>Objects</th>
<th>Column</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CJACML01</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TBB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LATSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LATP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 Press END.

The Mixed List panel is displayed. The *@ marker adjacent to the base table (TBB) object indicates that auxiliary objects were created or modified for the base table.

3 Press END until the WORKID Action Menu is displayed.

To undo changes to auxiliary objects

If you wanted to undo the changes to the base table, you must undo the changes to the auxiliary objects. In the Mixed List panel, type AX in the Act column adjacent to the modified base table (TBB). In the Auxiliary Objects List panel, remove the requested changes to the auxiliary objects.

- To undo changes to all of the objects listed, type UNDOALL on the Command line.

- To undo changes to one or more of the objects listed, type U in the Act column adjacent to the auxiliary objects.
Liking a base table and creating auxiliary objects for a table space

Where to go from here

After you create the auxiliary objects, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Liking a base table and creating auxiliary objects for a table space

You can issue a LIKE command on an existing base table that contains LOB columns. After you issue the command on the table, you must create the new auxiliary objects and associate them with the new table. You cannot issue the LIKE command on the existing auxiliary objects.

1. On the CHANGE MANAGER Main Menu, select **WORKID**, and press **Enter**.
   
   The WORKID Action Menu is displayed.

2. Type the name of the **WORKID** or type a wildcard pattern to display a list of work IDs. Then, select **Specify DB2 definitions** and press **Enter**.
   
   The Object Specification panel is displayed.

3. Specify the name of a table space (or use wildcard characters). Then, press **Enter**.
   
   The Mixed List panel is displayed.

4. Type **L** in the **Act** column adjacent to the base table (TBB) that contains a LOB column.

   The Create Table panel is displayed.

5. Specify the attributes for the table.

   **TIP**
   Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the **ZOOM (F4)** key.
A Specify the owner of the table.

B Specify the name of the table.

C Specify the AUDIT, VALIDPROC, EDITPROC, LABEL, DATA CAPTURE, RESTRICT DROP, OBID, and VOLATILE attributes of the table.

D Specify whether the parent table space already exists in the DB2 catalog.

E To modify the columns in the table, type S to select Table Column List.

F To specify a comment for the table, type S to select Table Comment.

--- TIP ---

Some object names might be too long to be displayed on a panel. To enter a long object name, position the cursor on the object name and press the ZOOM (F4) key.

--- NOTE ---

The new table will not have a primary key, even if the base table contained a primary key. If you want a primary key for the table, you must create it.

G To create a primary key for the table, type S to select Primary Key Columns.

--- NOTE ---

No auxiliary objects exist in the LIKEd base table.

H Press END.

The Mixed List panel is displayed.

--- NOTE ---

To create the auxiliary objects for the new base table, perform one of the following tasks:

- To create objects in a partitioned table space, see “Creating auxiliary objects for a partitioned table space” on page 525.

- To create objects in a nonpartitioned table space, see “Creating auxiliary objects for a nonpartitioned table space” on page 538.
Creating auxiliary objects for a nonpartitioned table space

Where to go from here

After you like the base table and create the auxiliary objects, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating auxiliary objects for a nonpartitioned table space

You can create auxiliary objects for an existing base table in a nonpartitioned table space. The CAX action code is used to create the auxiliary table spaces, tables, and indexes that are required for a base table that contains a LOB column.

In this task, you will

1. specify a base table
2. create the auxiliary table space
3. create the auxiliary table
4. create the auxiliary index
5. review the list of auxiliary objects
6. (optional) undo changes to the auxiliary objects

To specify a base table

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.
3 Specify the name of a nonpartitioned table space (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

4 Type CO in the Act column adjacent to the base table (TBB) that contains a LOB column.

The Table Columns List panel is displayed.

5 Type CAX in the Act column adjacent to the LOB column name and press Enter.

The Create Auxiliary Tablespace panel is displayed.

**To create the auxiliary table space**

From the Create Auxiliary Tablespace panel, specify the attributes for the table space.

**Tip**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

1 Specify the name of the table space.

2 Specify the owner of the table space.

3 Specify the attributes for the table space. To specify additional attributes, type S to select More Tablespace Attributes.

4 Specify whether the parent database already exists in the DB2 catalog.

5 Press END.

The auxiliary table space is saved in the CD tables. The Create Auxiliary Table panel is displayed.
Creating auxiliary objects for a nonpartitioned table space

To create the auxiliary table

From the Create Auxiliary Table panel, specify the attributes for the table.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

1. Specify the owner of the table.
2. Specify the name of the table.
3. Specify the name of the table space.
4. Specify a label for the table.
5. Specify the partition number for which you want to create the auxiliary table. For nonpartitioned table spaces, the partition number should be 0 (zero).
6. Specify whether the parent table space already exists in the DB2 catalog.
7. To specify a comment for the table, type S to select Table Comment.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

8. Press END.

The auxiliary table is saved in the CD tables. The Create Auxiliary Index panel is displayed.

To create the auxiliary index

1. From the Create Auxiliary Index panel, specify the attributes for the index.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A. Specify the owner of the index.
B. Specify the name of the index.
C Specify the owner of the table.

D Specify the name of the table.

E Specify values for the attributes. To specify additional attributes, type S to select More Attributes.

F Specify whether the parent table already exists in the DB2 catalog.

G To specify a comment for the index, type S to select Comment.

H Press END.

The auxiliary index is saved in the CD tables. The Table Columns List panel is displayed.

2 Press END.

The Mixed List panel is displayed. The *@ marker adjacent to the base table (TBB) object indicates that auxiliary objects were created or modified for the base table.

To review the list of auxiliary objects

1 From the Mixed List panel, type AX in the Act column adjacent to the base table to review the list of auxiliary objects for the base table.

The Auxiliary Objects List is displayed. The new auxiliary table space, table, and index are listed, as shown in Figure 163.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.
Press END until the WORKID Action Menu is displayed.

To undo changes to auxiliary objects

If you wanted to undo the changes to the base table, you must undo the changes to the auxiliary objects. In the Mixed List panel, type AX in the Act column adjacent to the modified base table (TBB). In the Auxiliary Objects List panel, remove the requested changes to the auxiliary objects.

- To undo changes to all of the objects listed, type UNDOALL on the Command line.
- To undo changes to one or more of the objects listed, type U in the Act column adjacent to the auxiliary objects.

Where to go from here

After you create the auxiliary objects, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Converting a VARCHAR data type to a LOB data type

CHANGE MANAGER supports the conversion of a VARCHAR or LONG VARCHAR data type to a BLOB or CLOB data type. Converting the data type requires two alter-type work IDs: one to add a ROWID column and unique index for the ROWID column, and another to change the column’s data type. To convert the data type to a LOB data type, you must analyze and execute both work IDs and worklists. For more information about unloading data that is contained in LOB columns, see “Unloading Unicode, LOB, and XML data” on page 694.

In this task, you will

1. specify the column
2. add a ROWID column and a unique index
3. convert the data type

**NOTE**

If you want to convert the data type without saving the data that is contained in the VARCHAR column, you can perform these steps in a single alter-type work ID.

**To specify the column**

1. On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

   The WORKID Action Menu is displayed.

2. Type the name of the WORKID and select Create A NEW WORKID. Then, press Enter.

   The Create WORKID panel is displayed.

3. Specify the information for the work ID.

   A Select Alter for the Type.

   B (optional) Specify a Comment to describe the work ID.

   C Press Enter.

4. Press END.

   The WORKID Action Menu is displayed. A message displayed in the upper-right corner indicates that the work ID has been added.
Converting a VARCHAR data type to a LOB data type

5 Type the name of the WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

The Object Specification panel is displayed.

6 Specify the name of a table space (or use wildcard characters). Then, press Enter.

The Mixed List panel is displayed.

7 Type CO in the Act column adjacent to the table that contains the column defined as a VARCHAR or LONG VARCHAR.

The Table Columns List panel is displayed.

To add a ROWID column and unique index

1 From the Table Columns List panel, type I in the Act column adjacent to a column name. Then, press Enter.

A blank line is inserted after the line on which you typed I.

2 Specify the attributes for the ROWID column.

   A Specify the name of the ROWID column.
   B Specify ROWID as the column type.
   C Type N in the Nl column to indicate that a null value is not allowed.
   D Type D in the Df column to indicate that the column is defined as GENERATED BY DEFAULT.

   **NOTE**

If the ROWID column is defined as GENERATED BY DEFAULT, a unique index is required for the column.

   E Press Enter.

3 Press END.

The Mixed List panel is displayed.

4 From the Mixed List panel, type CIX in the Act column adjacent to the table to which you added the ROWID column.

The Create Index panel is displayed.
5 Specify the attributes for the index.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Specify the owner of the index.

B Specify the name of the index.

C Type U to indicate that the index is unique.

D Specify whether the parent table already exists in the DB2 catalog.

E Type S to select **Keys and TB Cols**.

F Press Enter.

The Index Key Mixed List panel is displayed.

6 Specify the index key.

**TIP**

Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

A Type C in the **Act** column adjacent to the ROWID column name.

B Type A in the **Act** column adjacent to the Index Key.

C Press Enter.

The Index Key Mixed List panel is displayed. The index key column is displayed below the index key.

D Press END.

The Create Index panel is displayed.

7 Press END.

The Mixed List panel is displayed.
To convert the data type

1 Repeat step 1 through step 4 to create a second alter-type work ID.

2 Type the name of the second alter-type WORKID or type a wildcard pattern to display a list of work IDs. Then, select Specify DB2 definitions and press Enter.

   The Object Specification panel is displayed.

3 Specify the name of a table (or use wildcard characters). Then, press Enter.

   The Mixed List panel is displayed.

4 Type CO in the Act column adjacent to the table that contains the column defined as a VARCHAR or LONG VARCHAR.

   The Table Columns List panel is displayed.

5 From the Table Columns List panel, type E in the Act column adjacent to the name of the column that is defined as a VARCHAR. Then, press Enter.

   The Table Column Detail panel is displayed.
6 Change the attributes for the column.

**TIP**
Some object names might be too long to be displayed on a panel. To view a long object name, position the cursor on the object name and press the ZOOM (F4) key.

**A** Specify the name of the LOB column.

**B** If necessary, specify the schema name.

**C** Specify BLOB or CLOB as the data type for the column.

**D** Specify the length of the column.

**E** Specify whether null values are allowed.

**F** Specify whether the column has a default value.

**G** Press END.

The Table Columns List panel is displayed.

7 Press END.

The Mixed List panel is displayed. The table is now a base table with an object type of TBB.

8 To create the auxiliary objects for the base table, perform one of the following tasks:

- To create objects in a partitioned table space, see “Creating auxiliary objects for a partitioned table space” on page 525.

- To create objects in a nonpartitioned table space, see “Creating auxiliary objects for a nonpartitioned table space” on page 538.
Creating a new database and new table spaces for tables by importing DML

Where to go from here

After you convert the data type, perform the tasks in the following table to analyze the second work ID and execute the worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Creating a new database and new table spaces for tables by importing DML

You can import DML to create a new database and table space for tables. The script migrates all of the tables that have a specified number of rows from a database and table space to a new database in which each table resides in a single table space. The example provided shows how to specify the primary quantity for the new table space and to determine the row threshold for reassigning certain tables of a multi-table table space to new single-table table spaces. This task is especially useful to DBAs who implement and manage an ERP application.

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 164. Customize the database, table space, and table names to match your local requirements.

For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**TIP**
You can copy the DML statement to create a new database and table space for tables from the member ACMDMLU2 in the HLQ.DBCNTL data set.

Figure 164  DML for creating a database and table spaces (part 1 of 2)

```
SET SPACE ESTIMATION --space estimation turned on
;
LIKE DATABASE
```
### Figure 164 DML for creating a database and table spaces (part 2 of 2)

```
SET NAME = 'TESTCOPY' --specify new DB name
WHERE NAME = 'AKMGQA55' -- specify existing DB name
;
LIKE TABLESPACE
JOIN SYSIBM.SYSTABLES
SET DBNAME = 'TESTCOPY' --specify new DB name from first step
, NAME = SUBSTR(SYSIBM.SYSTABLES.NAME,1,2)
CONCAT DIGITS(SYSIBM.SYSTABLES.OBID)
WHERE SYSIBM.SYSTABLESPACE.DBNAME = 'AKMGQA55' --specify existing DB name
AND SYSIBM.SYSTABLESPACE.NAME = 'S6' -- specify existing TS name
AND SYSIBM.SYSTABLESPACE.DBNAME = SYSIBM.SYSTABLES.DBNAME
AND SYSIBM.SYSTABLESPACE.NAME = SYSIBM.SYSTABLES.TSNAME
AND SYSIBM.SYSTABLES.CARDF > 100 -- specify table row size threshold
AND SYSIBM.SYSTABLES.TYPE = 'T'
;
LIKE TABLEPART
JOIN SYSIBM.SYSTABLES
SET DBNAME = 'TESTCOPY' --specify new DB name from first step
, TSNAME = SUBSTR(SYSIBM.SYSTABLES.NAME,1,2)
CONCAT DIGITS(SYSIBM.SYSTABLES.OBID)
WHERE SYSIBM.SYSTABLEPART.DBNAME = 'AKMGQA55' --specify old DB name
AND SYSIBM.SYSTABLEPART.TSNAME = 'S6' -- specify old TS name
AND SYSIBM.SYSTABLEPART.DBNAME = SYSIBM.SYSTABLES.DBNAME
AND SYSIBM.SYSTABLEPART.TSNAME = SYSIBM.SYSTABLES.TSNAME
AND SYSIBM.SYSTABLES.TYPE = 'T'
AND SYSIBM.SYSTABLES.CARDF > 100 -- specify table row size threshold
;
UPDATE TABLE
SET DBNAME = 'TESTCOPY' -- specify new DB name from first step
, NAME = SUBSTR(NAME,1,2) CONCAT DIGITS(OBID)
WHERE DBNAME = 'AKMGQA55' -- specify old DB name
AND TSNAME = 'S6' -- specify old TS name
AND CARDF > 100 -- specify table row size threshold
;
```

In the sample DML that is displayed in Figure 164, note the following items:

- **CHANGE MANAGER** uses space estimation to explicitly estimate the space for only the table spaces that are specified in the LIKE TABLEPART DML statement.
- Table space names are based on the first two characters of the table name concatenated with the table OBID.
Changing multi-table table spaces to single-table table spaces by importing DML

**WARNING**

When you specify new table space names, substring table names can produce errors if you use underscores in the positions being selected. You might want to use a literal or other positions of the table name. Table OBIDs are unique only within the DBID or database in which they currently reside.

- Only tables that have more than 100 rows are reassigned to a new table space. The CARDF > 100 statement specifies the table cardinality for the DML trigger.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Changing multi-table table spaces to single-table table spaces by importing DML

For multi-table table spaces, you can import DML to move all of the tables to a new database in which each table resides in a single table space. This task is especially useful to DBAs who implement and manage an ERP application.

**NOTE**

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 165. Customize the database, table space, and table names to match your local requirements.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.
Changing multi-table table spaces to single-table table spaces by importing DML

Chapter 4  Altering data structures in a database environment  551

The DML creates a new database and table space, unloads the data from the tables, drops and creates tables, and reloads the data into the tables. No partitioned table spaces or related tables are migrated. It is assumed that a partitioned table space would not be empty.

**TIP**

You can copy the DML statement to change all tables in a multi-table table space to single-table table spaces from the member ACMDMML1 in the HLQ.DBCNTL data set.

Figure 165  DML to change multi-table table spaces to single-table table spaces (part 1 of 2)

```
LIKE DATABASE
SET NAME = 'TESTCOPY' -- specify new DB name
WHERE NAME = 'AKMGQA55' -- specify existing DB name
;
LIKE TABLESPACE
JOIN SYSIBM.SYSTABLES
SET DBNAME = 'TESTCOPY' -- specify new DB name from first step
, NAME = SUBSTR(SYSIBM.SYSTABLES.NAME,1,2)
CONCAT DIGITS(SYSIBM.SYSTABLES.OBID)
WHERE SYSIBM.SYSTABLESPACE.DBNAME = 'AKMGQA55' -- specify existing DB name
AND SYSIBM.SYSTABLESPACE.DBNAME = SYSIBM.SYSTABLES.DBNAME
AND SYSIBM.SYSTABLESPACE.NAME = SYSIBM.SYSTABLES.TSNAME
AND SYSIBM.SYSTABLESPACE.TYPE = 'T'
AND SYSIBM.SYSTABLESPACE.NTABLES > 1 -- specify number of tables in TS
;
LIKE TABLEPART
JOIN SYSIBM.SYSTABLES, SYSIBM.SYSTABLESPACE
SET DBNAME = 'TESTCOPY' -- specify new DB name
, TSNAME = SUBSTR(SYSIBM.SYSTABLES.NAME,1,2)
CONCAT DIGITS(SYSIBM.SYSTABLES.OBID)
-- User determines space allocated
-- In this example, the primary quantity is set to the value
-- from the CHANGE MANAGER space estimation program.
-- , PQTY = BMCSPACE
WHERE SYSIBM.SYSTABLEPART.DBNAME = 'AKMGQA55' -- specify existing DB name
AND SYSIBM.SYSTABLEPART.DBNAME = SYSIBM.SYSTABLES.DBNAME
AND SYSIBM.SYSTABLEPART.TSNAME = SYSIBM.SYSTABLES.TSNAME
AND SYSIBM.SYSTABLEPART.DBNAME = SYSIBM.SYSTABLESPACE.DBNAME
AND SYSIBM.SYSTABLEPART.TSNAME = SYSIBM.SYSTABLESPACE.NAME
AND SYSIBM.SYSTABLEPART.TYPE = 'T'
AND SYSIBM.SYSTABLESPACE.NTABLES > 1 -- specify number of tables in TS
AND PARTITION=0 -- includes only non-PT TS
;
UPDATE TABLE
JOIN SYSIBM.SYSTABLESPACE
SET TSNAME = SUBSTR(SYSIBM.SYSTABLES.NAME,1,2)
CONCAT DIGITS(SYSIBM.SYSTABLES.OBID)
, DBNAME = 'TESTCOPY' -- specify new DB name
WHERE SYSIBM.SYSTABLES.DBNAME = 'AKMGQA55' -- specify existing DB name
```
Moving all empty tables in a database to one table space by importing DML

You can import DML to move all of the empty tables in a database to one table space. This task is especially useful to DBAs who manage an ERP application and want to limit creating empty image copy data sets.

In the sample DML that is displayed in Figure 165, note the following items:

- Table space names are based on the first two characters of the table name concatenated with the table OBID.

  **WARNING**

  When you specify new table space names, substring table names can produce errors if you use underscores in the positions being selected. You might want to use a literal or other positions of the table name. Table OBIDs are unique only within the DBID or database in which they currently reside.

- The PQTY is set to the value that is estimated by CHANGE MANAGER space estimation.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

**Figure 165  DML to change multi-table table spaces to single-table table spaces (part 2 of 2)**

AND SYSIBM.SYSTABLES.DBNAME = SYSIBM.SYSTABLESPACE.DBNAME
AND SYSIBM.SYSTABLES.TSNAME = SYSIBM.SYSTABLESPACE.NAME
AND SYSIBM.SYSTABLES.TYPE = 'T'
AND SYSIBM.SYSTABLESPACE.NTABLES > 1 -- specify number of tables in TS
;
Moving all empty tables in a database to one table space by importing DML

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 166. Customize the database, table space, and table names to match your local requirements.

For information about DML syntax, see the **ALTER and CHANGE MANAGER for DB2 Reference Manual**.

**TIP**
You can copy the DML statement to move all of the empty tables in a database to one table space from member ACMDMILL2 in the **HLQ.DBCNTL** data set.

The DML creates a new database and table space, unloads the data from the tables, drops and creates tables, and reloads the data into the tables. No partitioned table spaces or related tables are migrated. It is assumed that a partitioned table space would not be empty.

**Figure 166  DML to move empty tables in a database (part 1 of 2)**

```
LIKE DATABASE
SET NAME = 'TESTCOPY' -- specify new DB name
WHERE NAME = 'AKMGQA55' -- specify existing DB name
;
LIKE TABLESPACE
SET DBNAME = 'TESTCOPY', -- same as DB name in previous step
NAME = 'new name' -- select new table space name
WHERE DBNAME = 'AKMGQA55'
AND NAME = 'old name' -- select an existing tablespace to like
;
LIKE TABLEPART
JOIN SYSIBM.SYSTABLESPACE
SET DBNAME = 'TESTCOPY', -- same as DB name in first step
TSNAME = 'new name'.
PQTY = 12 -- assign primary quantity
WHERE SYSIBM.SYSTABLEPART.DBNAME='AKMGQA55'
AND NAME = 'old name' -- select an existing table space to like
AND SYSIBM.SYSTABLEPART.DBNAME=SYSIBM.SYSTABLESPACE.DBNAME
AND SYSIBM.SYSTABLEPART.TSNAME=SYSIBM.SYSTABLESPACE.NAME
AND PARTITION = 0 -- guarantees TS is not partitioned
;
UPDATE TABLE
JOIN SYSIBM.SYSTABLESPACE, SYSIBM.SYSTABLEPART
SET DBNAME='TESTCOPY', -- same as DB name in first step
TSNAME = 'new name' -- same as TS name in second step
```

Chapter 4  Altering data structures in a database environment  553
Figure 166  DML to move empty tables in a database (part 2 of 2)

WHERE SYSIBM.SYSTABLES.DBNAME = 'AKMGQA55'
AND SYSIBM.SYSTABLES.DBNAME = SYSIBM.SYSTABLESPACE.DBNAME
AND SYSIBM.SYSTABLES.DBNAME = SYSIBM.SYSTABLEPART.DBNAME
AND SYSIBM.SYSTABLES.TSNAME = SYSIBM.SYSTABLESPACE.NAME
AND SYSIBM.SYSTABLES.TSNAME = SYSIBM.SYSTABLEPART.TSNAME
AND SYSIBM.SYSTABLEPART.PARTITION = 0 -- guarantees TB is not part of
   -- a partitioned TS
AND SYSIBM.SYSTABLES.TYPE = 'T' -- guarantees object is a table
AND SYSIBM.SYSTABLES.CARDF = 0 -- guarantees TB is empty as long as
   -- IBM RUNSTATS has been run
;

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in
   CHANGE MANAGER” on page 379.

Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a
work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
</table>
| Select Analyze WORKID and create a worklist. | “Analyzing an alter-type work ID and generating a
   worklist in CHANGE MANAGER” on page 560       |
| Select Execute WORKID worklist created by Analysis. | “Executing an alter-type worklist” on page 562 |

Setting table spaces and indexes to DEFINE NO by importing DML

You can import DML to set one or more table spaces and their dependent indexes to
DEFINE NO. This task is especially useful to DBAs who implement and manage an
ERP application and who want to reduce the creation of empty data sets.

NOTE

You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For
more information, see “Changing data structures by using DML” on page 415.
1. In an ISPF editor, type the DML statements to match those shown in Figure 167 and provide the information that is unique to your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

--- TIP ---
You can copy the DML statement to perform this task from member ACMDMLU1 in the HLQ.DBCNTL data set.

**Figure 167** DML to update table spaces and indexes with DEFINE NO

```sql
UPDATE TABLESPACE
SET
  DEFINE = 'N'
WHERE
  NAME = 'TSNAME' --specify existing TS name
  AND DBNAME = 'DBNAME' --specify existing DB name
;
UPDATE INDEXES
JOIN SYSIBM.SYSTABLES TB
SET
  DEFINE = 'N'
WHERE
  SYSIBM.SYSINDEXES.TBNAME = TB.NAME
  AND SYSIBM.SYSINDEXES.TBCREATOR = TB.CREATOR
  AND TB.DBNAME = 'DBNAME' --specify existing DB name
  AND TB.TSNAME = 'TSNAME' --specify existing TS name
;
```

2. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Updating table spaces and indexes based on cardinality of zero

To update one or more table spaces and their dependent indexes to DEFINE NO based on the cardinality of 0 (zero) of the underlying tables, type the DML statements shown in Figure 168.

**TIP**
You can copy the DML statement to perform this task from member ACMDMLU4 in the HLQ.DBCNTL data set.

---

**Figure 168  DML to update table spaces and indexes with DEFINE NO based on cardinality**

```
UPDATE TABLESPACE
JOIN SYSIBM.SYSTABLES TB
SET
  DEFINE = 'N'
WHERE
  SYSIBM.SYSTABLESPACE.NAME = TB.TSNAME
AND SYSIBM.SYSTABLESPACE.DBNAME = TB.DBNAME
AND TB.CARDF = 0
AND TB.DBNAME = 'PSOFT01' -- specify existing DB name
  AND TB.TSNAME = 'FSAPP' -- specify existing TS name
;
UPDATE INDEXES
JOIN SYSIBM.SYSTABLES TB
SET
  DEFINE = 'N'
WHERE
  SYSIBM.SYSINDEXES.TBNAME = TB.NAME
AND SYSIBM.SYSINDEXES.TBCREATOR = TB.CREATOR
AND TB.CARDF = 0
AND TB.DBNAME = 'PSOFT01' -- specify existing DB name
AND TB.TSNAME = 'FSAPP' -- specify existing TS name
;
```
You can import DML to update table spaces or indexes that were created with the DEFINE NO clause and for which a physical data set has not been created.

**NOTE**
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1. In an ISPF editor, type the DML statements to match those shown in Figure 169 and provide the information that is unique to your site.

   For information about DML syntax, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

   **Figure 169  DML to locate table spaces defined with DEFINE NO**

   ```sql
   UPDATE TABLESPACE SET
   DEFINE             =    'Y'
   WHERE
   DBNAME             =    'ERP'
   -- The following subSELECT statement finds the table spaces
   -- that were defined with the DEFINE NO clause and for which a
   -- physical data set has not been created.
   AND EXISTS (SELECT * FROM SYSIBM.SYSTABLEPART TP
   WHERE TP.SPACE = -1 AND
   SYSIBM.SYSTABLESPACE.DBNAME = TP.DBNAME AND
   SYSIBM.SYSTABLESPACE.NAME =   TP.TSNAME )
   ;
   ```

2. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

**Where to go from here**

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>
Updating the length of an inline LOB column by importing DML

For an inline LOB column in a table that is in a partition-by-growth or range-partitioned table space, you can import DML to update the column’s length.

NOTE
You can also perform this task by using the CM/PILOT DML_STRUCTURE_CHG script. For more information, see “Changing data structures by using DML” on page 415.

1 Create a DML statement in an ISPF editor. Provide the information that is required for your site.

For information about DML syntax, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

Figure 170 illustrates how to set the length of an inline CLOB column to the system default, which is greater than 4. CHANGE MANAGER stores the length as -2 in the CD tables.

<table>
<thead>
<tr>
<th>Figure 170</th>
<th>DML to set the length of an inline CLOB to the system default</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE COLUMNS -- updates all tables with LOB columns</td>
<td></td>
</tr>
<tr>
<td>-- that match the results of the WHERE clause</td>
<td></td>
</tr>
<tr>
<td>SET         -- sets the length to the system default &lt;DEFLT&gt;</td>
<td></td>
</tr>
<tr>
<td>LENGTH = -2  -- indicates that the length is an inline LOB</td>
<td></td>
</tr>
<tr>
<td>WHERE</td>
<td></td>
</tr>
<tr>
<td>LENGTH &gt; 4 AND</td>
<td></td>
</tr>
<tr>
<td>COLTYPE IN ('CLOB') AND</td>
<td></td>
</tr>
<tr>
<td>TBCREATOR = 'COAIN1'</td>
<td></td>
</tr>
<tr>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>

Figure 171 illustrates how to change an inline CLOB column to a non-inline CLOB column.

<table>
<thead>
<tr>
<th>Figure 171</th>
<th>DML to change an inline CLOB column to a non-inline CLOB column</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE COLUMNS</td>
<td></td>
</tr>
<tr>
<td>SET         -- sets length for an inline LOB to 0,</td>
<td></td>
</tr>
<tr>
<td>-- which changes the inline LOB to a LOB</td>
<td></td>
</tr>
<tr>
<td>LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td>WHERE       -- indicates that the length is an inline LOB</td>
<td></td>
</tr>
<tr>
<td>LENGTH &gt; 4 AND</td>
<td></td>
</tr>
<tr>
<td>COLTYPE IN ('CLOB') AND</td>
<td></td>
</tr>
<tr>
<td>TBCREATOR = 'D2NIC1'</td>
<td></td>
</tr>
<tr>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>
Figure 172 illustrates how to change the length for non-inline LOB columns to inline LOB columns.

**Figure 172  DML to change non-inline LOB columns to inline LOB columns**

```
UPDATE COLUMNS
SET
  LENGTH = 104
    -- sets length for LOB columns to 4 plus the inline
    -- length (in bytes); length must be greater than 4
    -- to be an inline LOB column
WHERE
  COLTYPE IN ('CLOB', 'BLOB', 'DBCLOB') AND
    -- changes all non-inline LOB
    -- columns to inline LOB columns
  TBCREATOR = 'C1AIB1';
```

Figure 173 illustrates how to update the length of the LOB columns that are in tables in a partition-by-growth or range-partitioned table space. You can use the DML when several types of table spaces exist in the same database.

**Figure 173  DML to change the length of LOB columns in universal table spaces**

```
UPDATE COLUMNS
JOIN SYSIBM.SYSTABLES TBS,
     SYSIBM.SYSTABLESPACE TS
SET
  LENGTH = 54
    -- sets length for LOB columns to 4 plus the inline
    -- length (in bytes); length must be greater than 4
    -- to be an inline LOB column
WHERE
  COLTYPE IN ('CLOB', 'BLOB', 'DBCLOB')
    -- for all LOB columns
  AND SYSIBM.SYSCOLUMNS.TBCREATOR = TBS.CREATOR
  AND SYSIBM.SYSCOLUMNS.TBNAME = TBS.NAME
  AND TBS.DBNAME = TS.DBNAME
  AND TBS.TSNAME = TS.NAME
  AND TS.TYPE IN ('G','R')
    -- in partition-by-growth or range-
    -- partitioned table spaces
  AND TS.DBNAME = 'C0AIN1';
```

Figure 174 illustrates how to set the length of an inline CLOB column based on the maximum length of the data contained in the column. If the length of the CLOB is greater than 5,000, CHANGE MANAGER sets the length to 500; otherwise, the product sets the length to 1/10 of the length of the CLOB.

**Figure 174  DML to set the length of the CLOB column based on the maximum length (part 1 of 2)**

```
UPDATE COLUMNS
SET
  LENGTH =
CASE
```
Where to go from here

After you import the DML file, perform the tasks in the following table to analyze a work ID and execute a worklist.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 562</td>
</tr>
</tbody>
</table>

Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER

You can analyze the change requests, determine their validity, and generate a worklist.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.

2 Type the name of an alter-type WORKID and select Analyze WORKID and create a worklist. Then, press Enter.

The Analysis Alter Worklist Interface panel is displayed.
3 Specify the information to analyze the work ID.

A Select **Generate a worklist which will convert and reload current data.**

---

**NOTE**

To generate a worklist that will create a full-recovery baseline, see Chapter 2, “Taking a snapshot of a database environment.” To generate a worklist that will allow you to fall back to a full-recovery baseline, see Chapter 5, “Maintaining database environments.”

B Select the **Run Type.**

C Press **Enter.**

The Analysis JCL Processing Interface panel is displayed.

4 Specify the data set names to be used by Analysis.

A *(batch run type)* Type the data set name for the JCL that is generated by this process.

B Type the data set name for the **Worklist** that is generated by this process.

C Type one of the following options for **Diagnostics:**

- *(foreground run type)* To display the diagnostics on the terminal, type **TERM.**
- To write the diagnostics to a sequential file, type the name of the data set.
- To write the diagnostics to a print data set, type **SYSOUT.**

D *(foreground run type)* If you specified **SYSOUT** in step 4C, specify the **Sysout Class.**

5 Select your options for processing the work ID.

A To override the options for analyzing the work ID, type **S** to select **Override.** The Analysis Options panels are displayed. For more information about overriding options for analysis, see “Overriding the default processing options” on page 61.

B To create the input to Analysis, select one of the following options:

- *(foreground run type)* Type **S** to select **Create analysis input.**
- *(batch run type)* Type **S** to select **Create JCL.**
Executing an alter-type worklist

C To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit analysis input.
- (batch run type) Type S to select Edit JCL.

D To create a worklist, select one of the following options:

- (foreground run type) Type S to select Run analysis.
- (batch run type) Type S to select Submit JCL.

E To modify the worklist, type S to select Edit Worklist.

F Press Enter.

6 Press END.

The WORKID Action Menu is displayed.

Where to go from here

After you analyze an alter-type work ID and generate a worklist, select Execute WORKID worklist created by Analysis to execute a worklist.

TIP
To execute a worklist, see page 562.

Executing an alter-type worklist

You can execute an alter-type worklist that is generated by Analysis. When you execute the JCL that is created, the changes in the worklist are made to the DB2 catalog. Occasionally you might need to restart a worklist or start a worklist over.

TIP
To restart a worklist, see page 164. To start a worklist over, see page 166.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.
2 Type the name of the WORKID and select Execute WORKID worklist created by Analysis. Then, press Enter.

The Execution JCL Build Interface panel is displayed.

3 Specify the information to build and run the JCL.

   A Select Build Initial JCL to build the JCL for the first time.

   B Select a Run Type.

   C Press Enter.

   The Execution Pre- and Post-Processing Interface panel is displayed.

4 (CHANGE MANAGER only) Specify your options for a multi-step job.

   **TIP**
   
   To create a multi-step job, see page 564.

5 Press Enter.

   The Execution JCL Processing Interface panel is displayed.

6 On the Execution JCL Processing Interface panel, specify the data set names to be used by Execution.

   A Type the data set name for the Execution JCL that is generated by this process.

   B Type the data set name for the Worklist that is generated by this process. (The name of the file that was built by Analysis should be displayed.)

   C Type one of the following options for Diagnostics:

   - To write the diagnostics to a sequential file, type the name of the data set.
   - To write the diagnostics to a print data set, type SYSOUT.

   D (batch run type) Type the name of the data set to contain the Batch JCL Job. The batch job is used to generate the Execution JCL.

7 Select the method of sizing the data sets. For more information about data set sizing, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.
Generating a multi-step job

8 Select the options for processing the worklist that was generated by Analysis.

A To override the options for executing the worklist, type S to select Override Defaults. The Execution Options panels are displayed. For more information about overriding options for execution, see “Specifying options for Execution” on page 82.

B To edit the worklist that was generated by Analysis, type S to select Edit Worklist.

C To create the input to Execution, select one of the following options:

- (foreground run type) Type S to select Build Execution JCL.
- (batch run type) Type S to select Build Batch JCL.

D To review or modify input to the process, select one of the following options:

- (foreground run type) Type S to select Edit Execution JCL.
- (batch run type) Type S to select Edit Batch JCL.

E Select one of the following options:

- (foreground run type) To execute the worklist, type S to select Submit Execution JCL.
- (batch run type) To submit the batch JCL, type S to select Submit Batch JCL.

F Press Enter.

9 (batch run type) To execute the worklist, type S to select Edit New Execution JCL. From the ISPF editor, submit the JCL.

Generating a multi-step job

You can specify additional steps to be included in the Execution job for CHANGE MANAGER. You can specify whether to perform a comparison before and after the worklist is executed. You can also specify whether to build a baseline before and after the worklist is executed.

1 On the CHANGE MANAGER Main Menu, select WORKID, and press Enter.

The WORKID Action Menu is displayed.
2. Type the name of the WORKID (or type a wildcard pattern) and select **Execute WORKID worklist created by Analysis**. Then, press **Enter**.

   The Execution JCL Build Interface panel is displayed.

3. Specify the information to build and run the JCL.

   **A** Select a JCL Type.

   **B** Select a Run Type.

   **C** Press **Enter**.

   The Execution Pre- and Post-Processing Interface panel is displayed.

4. Specify your options for performing a comparison or building a baseline before the worklist is executed.

   **A** Select one of the following options regarding a comparison before the worklist is executed:

   - Select **Do not do a compare before execution** to not perform a comparison.
   - Select **Compare two previous baselines** to identify the differences between two existing baselines at two points in time.
   - Select **Compare a previous baseline to the current catalog** to identify the differences between an existing baseline and the current DB2 catalog at two points in time.

   **B** Select one of the following options to create a baseline before the worklist is executed:

   - Select **Do not build a baseline before execution** to not build a baseline.
   - Select **Build a baseline before execution** to create a structure-only catalog baseline. This baseline serves as a record of the data structure definitions before the worklist is executed.

   **NOTE**

   This baseline cannot be used for a full recovery.
5 Specify your options for performing a comparison or building a baseline after the worklist is executed.

A Select one of the following options to perform a comparison after the worklist is executed:

- Select **Do not do a compare after execution** to not perform a comparison.
- Select **Compare a previous baseline to the new catalog structures** to identify the differences between an existing baseline and the new DB2 catalog structure that was altered by Execution.
- Select **Compare the baseline built during a previous job step to the new catalog structures** to identify the differences between the new baseline that was created by the Baseline component and the new DB2 catalog structure that was altered by Execution. Compare builds the CDL that is needed to recover back to the data structure definitions that were captured in the Pre-Execution Baseline step.
- Select **Create CDL to fall back to a previous baseline** to identify the differences between the DB2 catalog and an existing catalog baseline that was created in the Pre-Execution Baseline step. Compare builds the CDL that is needed to recover the structures and data.

B Select one of the following options to create a baseline after the worklist is executed:

- Select **Do not build a baseline after execution** to not build a baseline.
- Select **Build a baseline after execution** to create a structure-only catalog baseline.

---

**NOTE**

This baseline cannot be used for a full recovery.

---

6 Press Enter.
Depending on the options that you selected, the following panels are displayed:

- Create Baseline Interface panel, on which you specify the name of the baseline profile, the name of the baseline (or a defined baseline template), and the name of the diagnostics data set

- Execution Compare Interface panel, on which you specify the name of the baseline, the name of the data set for the generated CDL, and the name of the diagnostics data set

- Execution JCL Processing Interface panel, on which you specify the data set names for JCL and the worklist, and select the worklist processing options

**Where to go from here**

After you generate a multi-step job, see “Executing an alter-type worklist” on page 562 to execute the worklist.

**Where to go from here**

Now that you have altered your data structures, you can accomplish the goals that are listed in the following table.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate database environments</td>
<td>Chapter 1</td>
</tr>
<tr>
<td>Take a snapshot of a database environment</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>Compare database environments</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Maintain database environments</td>
<td>Chapter 5</td>
</tr>
</tbody>
</table>
Where to go from here
Managing environments with ALTER and CHANGE MANAGER

This part presents the following topics:

Chapter 5
  Maintaining database environments ................................................. 571

Chapter 6
  Managing your PeopleSoft environment with CHANGE MANAGER ........ 609
Maintaining database environments

This chapter presents the following topics:

Before you begin ................................................................. 571
Overview ............................................................................. 572
Migrating changes to data structures ...................................... 572
  Acting as the sending subsystem ...................................... 573
  Acting as the receiving subsystem .................................... 575
Maintaining applications ....................................................... 577
  Implementing a strategy ................................................... 578
Recovering data structures .................................................... 579
Feeding back changes to data structures ............................... 580
Using data modeling tools ..................................................... 581
  Importing DDL from a data modeling tool .......................... 582
  Maintaining catalog information ..................................... 583
Accomplishing your goals ...................................................... 583
  Migrating data structure changes to other subsystems .......... 584
  Merging subsystems by using ALTER .............................. 585
  Receiving data structure changes from a strong sender ........ 587
  Receiving data structure changes from a strong receiver ....... 590
  Restoring data structures with current data ......................... 595
  Recovering data structures with current data ..................... 596
  Restoring data structures with old data ............................. 601
  Recovering data structures with old data ......................... 602
Where to go from here .......................................................... 607

Before you begin

Before you read this chapter, you should be familiar with the concepts in the migration, baseline, comparison, and alter processes. For information, see this user guide.
Overview

The CHANGE MANAGER product enables you to manage your application’s data structures and data over multiple DB2 subsystems. The product can manage complex environments in which you can design multiple data structures, install these structures on multiple DB2 subsystems, vary the structures at each DB2 location to meet the requirements for that site, and implement version upgrades over the life of the application. CHANGE MANAGER can also interact with common data modeling tools that are used to design an application’s data structures. You can use the data modeling tools to make design changes, and then use CHANGE MANAGER to upgrade the DB2 subsystems with those changes.

This chapter discusses how you can use CHANGE MANAGER to migrate changes to other subsystems, maintain your applications, recover data structures, and feed back changes to a subsystem.

Migrating changes to data structures

CHANGE MANAGER enables you to move data structure changes from one subsystem to another. Migrating changes instead of migrating complete data structures is more efficient and has the following advantages:

- Local modifications or structure dependents are retained. These elements might include tuning or sizing modifications, as well as data structures that are created for local use.
- Local data in the structures is retained.
- Implementing changes can use fewer system resources than a full data-structure migration.
- Change migration is faster and easier to implement when only small changes (such as emergency changes between major releases of an application) are necessary.

The method for migrating changes is the Change Definition Language (CDL) that is generated by the Compare component of CHANGE MANAGER. When you use the change migration process, you can be the sending subsystem, the receiving subsystem, or both.

**NOTE**

Both the sending and the receiving subsystems must have CHANGE MANAGER installed.
Acting as the sending subsystem

When you are the sending subsystem in a change migration process, you must generate and transmit CDL, as well as establish the procedures for managing the flow of changes. As the sending subsystem, you should consider the following types of changes:

- version updates, which are transmitted whenever the application’s software is upgraded to incorporate new features or to correct errors
- emergency changes, which are performed to fix problems in the data structure design or implementation that were not fixed in or were caused by the most recent version update

These changes are usually required to ensure the continued operation of the system. For example, if an employee tracking system uses a three-digit employee ID number, you will need to make an immediate change on the day your company hires its 1,000th employee.

- minor changes, which are similar to emergency changes, except that they are usually not as critical

To avoid an infinite flow of changes from the sending subsystem to the receiving subsystem, you should use the following guidelines:

- Determine when the changes are transmitted. Monthly updates can be acceptable for some applications, while other applications require more immediate attention.
- Determine what severity of change requires immediate action. If the problem does not cause severe performance penalties or data loss, it might be more efficient to wait to transmit the change until the next regular update.
- Determine what types of changes are transmitted. If each production site has its own DASD administrator, you might not need to transmit storage group changes.

Establishing versions of applications

You use baselines to capture data structures and data at a specific point in time. Baselines act as control points in the change migration process and establish a static set of data structures for an application version. If you make a change that has unwanted results, you can restore the data structures and data to a prior baseline. You can also compare two baselines in order to generate changes that you can migrate to another subsystem.
You can establish baselines on the DB2 catalog (catalog baselines) and on a data definition language (DDL) file or worklist (DDL or worklist baselines). Because DDL baselines are usually more applicable for a receiving subsystem, this discussion assumes that you are using catalog baselines. Before you can establish a baseline, you must create a baseline profile. A catalog baseline profile contains header information such as the date and time the profile was created, the date and time it was last used, and an optional baseline name template. In addition, a catalog baseline profile contains a scope that defines the set of data structures to include in the baseline.

When you create a baseline profile, consider the following information:

- Baseline naming conventions can help you to identify the baseline that contains a particular version of an application’s data structures.

- You can use the baseline name template to automatically generate baseline names that contain the date of the baseline or an ascending sequence of numbers (#) with arbitrary text.

- You can use either a sequence of six or eight at symbols (@) to include the date as part of the baseline name.

- You can combine dates and numbers in a baseline name template by using both sets of special characters.

- You can edit the baseline profile and change the template name whenever a major change occurs in the application.

- You can define the scope for a catalog baseline profile explicitly, or the baseline profile can reference the scope in an outbound migrate profile.

  Referencing the scope can avoid problems that might arise when the scope in the outbound migrate profile does not match the scope in the baseline profile.

For more information about baselines, see Chapter 2, “Taking a snapshot of a database environment.”

**Generating changes**

The two primary methods used for generating CDL for changes from the sending subsystem include

- comparing the baseline that was created from the previous version of the application to the current version

- converting a work ID to CDL, so that you can send changes between major version updates
When you generate CDL by comparing two versions of an application’s data structures, you should know the current version status of the receiving subsystem. When you transmit emergency changes by converting a work ID into CDL, you can isolate those changes from other potential changes by using a separate work ID. For example, if you need to add a column to a table (emergency change) and increase the primary and secondary allocation quantities for a table space (routine maintenance change), specify those changes under separate work IDs so that you do not transmit the table space changes to the receiving subsystems.

To generate changes for multiple receiving locations or to customize the changes for the specific requirements one receiving location, use an outbound migrate profile in the comparison. If you already have an outbound migrate profile for use in the migration process, you can use the same migrate profile to customize the CDL that is generated by the comparison.

If you use the work ID to CDL conversion procedure for transmitting changes, create the work ID name according to the logical naming conventions so that you can easily identify the work later.

**Acting as the receiving subsystem**

When you are the receiving subsystem in a change migration process, you must perform the following tasks:

1. Import CDL to an alter-type work ID.
2. Analyze the work ID.
3. Execute the generated worklist.

The following variations can apply:

- If the sending subsystem transmits a DDL file or a worklist, you can convert it to CDL by comparing it to a previous version.
- If the transmitted changes contain elements that do not conform to local requirements, such as object names or allocation quantities, you can apply an inbound migrate profile to automatically transform the imported data structures.
- You can also filter out changes in the CDL that you import by using suppress-type change rules in the inbound migrate profile.
Receiving a DDL file or a worklist

If the sending subsystem transmits a DDL file or a worklist, you can transform it into CDL by comparing it to a prior version. The prior version can be stored in a DDL or worklist file or can be captured in a DDL baseline. In either case, it is important that you compare the prior version to the new version, not to the current DB2 catalog. Although CHANGE MANAGER enables you to compare a DDL file or worklist to the local catalog, you should consider several factors before you perform this type of comparison.

The DDL must contain a complete description of the application’s data structures as if they had not been installed. The DDL that is generated for some applications contains a series of DROP statements at the beginning of the file that discard the old version’s structures, followed by CREATE statements for the structures. This is sometimes called *incremental DDL*. Incremental DDL causes errors when you import it into CHANGE MANAGER. If you have a file of incremental DDL, edit it to remove the DROP statements before using it in the change migration process.

For example, you receive version 4 of an application’s DDL on February 1 and immediately install it. You receive version 5 on March 1. If you execute the DDL, you lose any local modifications—authorizations, views, or synonyms—that were created during the month. In addition, you must manually unload, convert, and reload data.

If you compare the current catalog to version 4 of the DDL, the resulting CDL specifies the changes for converting version 4 to version 5, but causes the loss of all local modifications because they are not contained in the DDL file. If you compare version 4 of the DDL to version 5 of the DDL, the resulting CDL contains only the changes necessary to transform version 4 to version 5, retaining local modifications. Version 4 of the DDL can be either the original DDL file or a DDL baseline. A DDL baseline is preferable because it is stored permanently in DB2. On many systems, files that are unused are archived and often accessible only with extra effort.

If you use a DDL baseline instead of a DDL file, the Compare component can resolve renamed objects by using the CHANGE MANAGER rename table. For example, you receive version 6 of the DDL on April 1 and version 7 of the DDL on May 1. If you changed the name of a table and properly updated the rename table, the Compare component generates CDL showing the table name change. If you simply compared DDL to DDL, Compare has to generate a DROP statement for the table with the old name and a CREATE statement with the new name. This action can result in the loss of all of the data in the table.
Evaluating the CDL file

Evaluate the CDL before you import it. You can edit the CDL to remove unwanted statements or elements. You can also filter out changes that are specified in the import CDL by using suppress-type change rules in an inbound migrate profile. For example, if the CDL reduces the allocation quantity on a large table space, you might want to filter out that change to retain your current values. CDL is contained in a sequential data set or PDS member, from which you can view or edit it.

Using an inbound migrate profile

If the sender is unaware of the requirements of the receiving subsystem, you might need to transform the received changes to meet local requirements. Although it is possible to simply edit the CDL file, you can use an inbound migrate profile to automate the changes. After you create the profile, you can reuse it whenever you perform the same change migration.

Implementing the changes

After you import the changes into an alter-type work ID on the receiving subsystem, the process continues exactly like a local change. Submit the work ID to Analysis to generate a worklist, and execute the worklist to implement the changes.

Maintaining applications

In the course of maintaining applications across several DB2 sites, you must implement a migration strategy. Every migration strategy establishes a control point.

A strong sender migration strategy, for example, might use the development subsystem as the control point for defining data structure definitions. Change control over the application originates at the development subsystem. When the application needs to be established at another subsystem, migration of the application’s structures, data, or both takes place using a migrate worklist. When an application needs to be maintained at another subsystem, CDL is used to migrate the application’s data structures.

When the control point of a migration process is at the receiving subsystem, then the migration strategy is referred to as a strong receiver. The subsystem that receives the migrate worklist applies rules that enforce local requirements for the application’s structure definitions. This strategy is sometimes preferred when DB2 software is purchased from a third-party vendor or when application data structures are implemented over a wide network of DB2 systems.
Implementing a strategy

The following scenario provides examples of how you can prepare for structure recovery, and one method for establishing and maintaining an application over several subsystems.

The USA Credit company has data centers in Austin, Texas and Dallas, Texas. Recently, the company expanded operations and needs to set up a new site in Houston, Texas. DB2 application development is maintained in Austin, while the Dallas and Houston sites are maintained as production-only environments.

As the database administrator, you must have the current version of the new application up and running in Houston and Dallas by the end of the week. The safest and most efficient method to create the same DB2 data structures in Houston as those in Austin is to perform a migration process. The test subsystem in Austin is the control point for defining data structure definitions and implementing subsequent versions of the application. In this instance, the Austin site is acting as a “strong sender.”

When you establish the application in Houston, you also want to automate the application of certain changes. You know, for example, that you want the database name and the table names in Houston to use a different naming convention than the ones used in Austin. You also want to be sure that you move a specific set of data structures. In addition, you know that some changes to the application are planned in the coming months, and when the application is in test and running smoothly you will want to migrate the changes to the application to Houston again.

Planning to establish and maintain the application

Basically, this scenario can be divided into two categories of activities:

- establishing the application on another subsystem
- moving changes from the control node to the receiving subsystems on an ongoing basis

In planning to properly establish the application, you perform the following tasks:

- Create the catalog baseline profile and the migrate profile.
- Establish a baseline of the current application version.
- Perform a full migration process to establish the application in Houston.
In the future, after you have made changes to the application and want to apply these changes to the application across several subsystems, you perform the following tasks:

- Establish a baseline of the changed application.
- Perform a change migration process.

**Using an outbound migrate profile**

An outbound migrate profile provides a means of control when performing a migration process and a change migration process. You can define the scope of the outbound migrate profile to select the same DB2 object structures each time the migration is performed. The change rules that are defined for the outbound migrate profile can ensure the implementation of the Houston naming conventions consistently.

**Establishing a baseline for control and recovery**

When you think about the task of implementing ongoing changes, you realize that you would like to set up a way to automate the synchronization of versions of the application across multiple subsystems. Knowing that it is impossible to anticipate every consequence of performing a change, you also see that you might need to fall back to a previous version of the application in the future.

Establishing a baseline provides a way to create a point of recovery. By taking baselines of each version of the application, you can perform a comparison between two versions of an application. The CDL that is generated during this process can then be applied to each receiving subsystem. You can use this strategy to apply upgrades to applications running on other subsystems without running the risk of losing any local modifications.

**Recovering data structures**

Data structure recovery enables you to return to a previous version of a data structure within a single DB2 subsystem. Recovery is necessary in the following circumstances:

- A failure occurs when you implement a change to a data structure.
- Results are unsatisfactory after a change has been completed.
- A change was completed satisfactorily, but other (later) changes have created unsatisfactory results.
Data structure recovery also enables you either to return to the data that you had before the changes were applied or to retain current data while you recover the earlier structure definitions. If a data loss occurs as a result of the change (such as when a column or table is dropped), CHANGE MANAGER recovers data to the point at which the change was applied.

The first step in the recovery process is to establish a full-recovery baseline of the data structure and the data before the proposed changes are made. To recover data to the same point, you should make a copy of the data sets. If recovery becomes necessary after you implement the data structure changes, then compare the current DB2 catalog with the previously-generated baseline to determine the changes that must be reversed. The change definitions are converted to change requests by Import and then verified by Analysis. Analysis then produces a worklist, which Execution uses to restore the previous version of the data structure and to restore the data.

**NOTE**

The worklist contains commands to drop and create only the tables that were changed and also includes commands to perform the following tasks:

- For tables that are in multi-table table spaces, drop the tables and load the data with the RESUME YES parameter.
- For tables that are in partitioned table spaces or single-table table spaces, load the data with the REPLACE parameter.

**Feeding back changes to data structures**

Changes do not always flow smoothly from the development system to the test system, and finally to the production system. Often, you need to make changes on the receiving subsystem to correct a problem or improve performance. The change feedback process enables you to copy the changes that have been made locally to data structures that were received from another subsystem, and then send those changes back to that subsystem for application to the original data structure.

In the change feedback process you must determine whether the changes are global (those that apply to multiple subsystems) or local (those that apply only to a single subsystem). For example, authorizations are usually a local issue, while referential integrity features such as indexes and foreign keys are a global issue. Table space allocation quantities can be either local or global issues, depending on how you manage your system.
You must also consider the following items:

- when changes are transmitted
- the severity of change that requires immediate action
- the types of changes that are transmitted

You also need to consider the effect of change rules on the change feedback process. If you used change rules in a migrate-type work ID or an outbound migrate profile to change object attributes when you migrate to the receiving subsystem, you must reverse the process when you move changes back into the control subsystem. For example, if you changed your table names from TEST.* to PROD.* when you moved them from test to production, you must change the names from PROD.* to TEST.* when you feed back changes.

Typically, you make changes locally on the production system (by changing the data structures) after a baseline has been taken. After the changes have been made, Compare identifies the differences between the pre-change baseline and the DB2 catalog. Those differences are written to a CDL file. After you have determined the type of changes to transmit to the control system, evaluate the CDL to see if it contains unwanted elements and edit the CDL file to remove those elements. (Alternatively, you can use suppress-type change rules in an outbound migrate profile to suppress the unwanted elements.) On the development system, Import converts the change definitions in the CDL file to change requests. These changes are verified by Analysis, which then produces a worklist that Execution uses to make the appropriate updates to the data structures.

**Using data modeling tools**

If you use a data modeling tool to design your database, you can use several of the processes that are discussed in this chapter and this user guide to streamline the administration of your system. For example, you can:

- use the change migration process to update your system with later changes
- use the migrate process to install initial application versions on your system
- use the baseline process to establish recovery points on the data modeling tool’s data structures and to restore your system to prior versions if necessary
- use the change feedback process to transmit changes back to the data modeling tool
Importing DDL from a data modeling tool

Several types of data modeling tools are available. Some are mainframe-based and interact directly with DB2; others run on workstations or personal computers and communicate with DB2 only through DDL. Even for systems that run on the mainframe, the data structure design output from a data modeling tool is usually DDL.

To install an initial version of an application from a data modeling tool, you perform the following steps:

1. Import the DDL to an alter-type work ID (using Import).
2. Submit the work ID to generate a worklist (using Analysis).
3. Execute the worklist (using Execution).

**WARNING**

When you import DDL to a work ID, you lose any authorizations that are granted in the DDL file.

When you receive the DDL file for a new version of your application’s data structures, you can perform a change migration process by

- comparing the DDL to the previous version (DDL or baseline)
- evaluating the generated CDL
- importing the CDL to a work ID, running Analysis, and then running Execution

If the data modeling tool generates CDL, you need to evaluate, import, analyze, and execute the CDL.

Whenever you import a new or updated version of the data structures that are generated by the data modeling tool, establish one or more baselines to allow for recovery in case you discover errors.

**NOTE**

CHANGE MANAGER uses the term scope to mean the set of DB2 objects that are affected by an operation, where the operation can be a migration, a comparison, or the establishment of a baseline. Although CHANGE MANAGER selects structures using scope rules, the method of structure selection is sometimes different in the data modeling tool and yields a different set of objects. You should keep this difference in mind when you read this user guide or the documentation for a data modeling tool.
Maintaining catalog information

You can maintain a copy of the DB2 catalog information for the structures that data modeling tools generate. This maintenance can be performed by either creating a migrate worklist and converting the worklist to DDL (which contains authorizations), or by generating DDL through a baseline report (which does not contain authorizations).

If you use a data modeling tool, a copy of DB2’s catalog information is maintained for the structures that are generated by the design. You can quickly and easily update the data modeling tool’s catalog information. The Compare component can extract the information from the catalog and transmit it to the data modeling tool in a CDL file. In addition, if you are a licensed user of DASD MANAGER PLUS, Compare can transmit information from DASD MANAGER PLUS statistics tables.

Accomplishing your goals

ALTER and CHANGE MANAGER enable you to maintain applications across several DB2 sites. Table 42 lists the tasks that you can perform to maintain your database environments.

Table 42  Maintenance tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Migrating data structure changes to other subsystems”</td>
<td>page 584</td>
</tr>
<tr>
<td>“Merging subsystems by using ALTER”</td>
<td>page 585</td>
</tr>
<tr>
<td>“Receiving data structure changes from a strong sender”</td>
<td>page 587</td>
</tr>
<tr>
<td>“Receiving data structure changes from a strong receiver”</td>
<td>page 590</td>
</tr>
<tr>
<td>“Restoring data structures with current data”</td>
<td>page 595</td>
</tr>
<tr>
<td>“Recovering data structures with current data”</td>
<td>page 596</td>
</tr>
<tr>
<td>“Restoring data structures with old data”</td>
<td>page 601</td>
</tr>
<tr>
<td>“Recovering data structures with old data”</td>
<td>page 602</td>
</tr>
</tbody>
</table>
You can migrate data structure changes from a control subsystem to receiving subsystems. This procedure consists of establishing a baseline of the changed application and performing a change migration process.

1 Establish a catalog baseline of the changed application.

   A Create a baseline profile and use the scope rules to control the objects that are included in the baseline. For more information, see “Creating a catalog baseline profile” on page 198.

   B Create a baseline from the baseline profile that you created in step 1A. See “Creating a structure-only catalog baseline” on page 201.

2 Compare two versions of a catalog baseline.

   A Create an outbound migrate profile that specifies your receiving subsystems (locations). For more information, see “Creating an outbound migrate profile” on page 146.

   B Compare a baseline to a baseline, using the outbound migrate profile that you created in step 2A. For more information, see “Comparing a baseline to a baseline” on page 283.

3 Import the CDL from the comparison in step 2 to an alter-type work ID.

   A Create an alter-type work ID. For more information, see “Creating an alter-type work ID” on page 357.

   B Import the CDL to the work ID that you created in step 3A. For more information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

4 Analyze the alter-type work ID and generate a worklist. For more information, see “Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560.

5 Execute the worklist. For more information, see “Executing an alter-type worklist” on page 562.
Merging subsystems by using ALTER

You can merge subsystems by using the ALTER product with the unload and load utilities. Before you begin you should have access to the IBM documentation.

1 Choose a DB2 subsystem’s catalog to be the original catalog for the data sharing group. For information about merging existing DB2 subsystems, see the IBM documentation. The subsystem that you select as the original will be referred to as the target, while the other subsystems will be referred to as the source subsystems.

2 Perform the following steps for each source subsystem.

A Identify objects from the other subsystems to move to the target subsystem. Determine whether name conflicts will occur when all of these objects are merged.

Determine how each duplicate object will be handled. Answer the following questions:

■ Should one of them be renamed?
■ Can one set be identified as the master copy?
■ Does data from multiple tables need to be merged? If yes, identify how this will be accomplished.

WARNING

Merging data from multiple tables can be very complex if the tables have application referential integrity.

B Grant system privileges of the source subsystems on the target subsystem, if they do not already exist.

Resolve authorization name conflicts on all authorization types. Answer the following questions:

■ Should duplicates be eliminated?
■ Does one of the IDs need to be renamed?

C Create a migrate-type work ID with ALTER. The work ID name must be unique when it is received on the target subsystem. Set all of the migrate options in the work ID to Yes so that dependent objects, authorizations, and data will be migrated.
Enable the buffer pools that are needed by objects from the source subsystems on the target subsystem. Buffer pool authorizations must be granted on the target subsystem. You can use migrate-type work ID change rules to change the buffer pools as the objects are migrated.

For more information, see “Creating a migrate-type work ID” on page 114.

D Using the Specification component in ALTER, migrate the DB2 objects that were specified in step 2A on page 585. For more information, see “Specifying the data structures in a migrate-type work ID” on page 119.

The dependents for an object can be omitted in either of two ways:

- To omit all objects of a specific type, change the migrate options for an object. For example, all synonyms for a specific table can be omitted by typing the MO line command next to the table on the Mixed List panel. Set the include option for synonyms to No. This action will omit that object type and all of its dependents. If tables are omitted for a table space, the indexes, views and other dependents are also omitted.

- To omit a specific dependent object, type the DROP command next to the dependent’s name on the Mixed List panel. This action works like an exclude in a migrate-type work ID, and that object and its dependents are excluded from the worklist.

E Analyze the migrate-type work ID and generate a worklist. For more information, see “Analyzing a migrate-type work ID and generating a worklist” on page 153.

Consider the following items before analyzing the work ID:

- You can migrate plans and packages for these objects by selecting Bind on the Analysis Worklist Command Options panel. You can also migrate plan and package authorizations by selecting Bindauth on this panel.

- Analysis builds a worklist that contains the DDL that is needed to create these objects and their authorizations on the target subsystem. This worklist contains the utilities necessary to move the data. If the BMC utilities are available, verify that they are selected on the Analysis Utility Options panel. For more information, see Chapter B, “Using utilities with ALTER and CHANGE MANAGER.”
Receiving data structure changes from a strong sender

The CM/PILOT REC_CHG_SS script contains the steps to import a CDL file or DDL file that was sent from a DB2 subsystem that controls the changes to be applied to the local subsystem. The script then creates an Analysis worklist to apply the changes.

In this task, you will create and execute task ID.

Before you begin

Ensure that a CDL or DDL file exists. If you want to use change rules, use CHANGE MANAGER to create an inbound migrate profile. For information, see “Creating an inbound migrate profile in CHANGE MANAGER” on page 361.

To create a task ID

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.
Receiving data structure changes from a strong sender

2 Select TASKIDs and press Enter.

The TASKID Action Menu is displayed.

3 Type the name of a new TASKID and select Create a TASKID. Then, press Enter.

The Script Selection List is displayed.

4 Select Receive data structure changes (strong sender) and press Enter.

The Create TASKID panel is displayed.

5 Specify the information for a task ID.

A CM/PILOT uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for WORKID or type the name of a work ID name template.

B (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

TIP
If you frequently use this script to receive changes for the same DB2 application, you can create an inbound migrate profile with the change rules that you need, define it to a CM/PILOT Application, and then specify the CM/PILOT Application each time that you create a task ID for this job.

C (optional) Specify a Comment to describe the task ID.

D Press Enter. Then, press END.

The task ID is created and the TASKID Action Menu panel is displayed.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT Worklist.

B Type the data set name for the CM/PILOT JCL.
C Type the data set name for **Diagnostics** or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

A Type **S** to select **Override CHANGE MANAGER options** and then press **Enter**.

The Override CHANGE MANAGER Options panel is displayed.

B Select **Override CHANGE MANAGER Options**.

C Type **S** to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press **Enter**.

For more information about the CHANGE MANAGER options, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*

| **WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type **S** to select **Create Worklist**. Then, press **Enter**.

The Step Settings panel is displayed.

B (optional) To modify the Analysis step, type **S** to select **Step 2 Analysis step** and press **Enter**.

C (optional) To modify the Execution step, type **S** to select **Step 3 Execution step** and press **Enter**.

D Press **Enter** until the TASKID Migrate Profiles panel is displayed.
RECEIVING DATA STRUCTURE CHANGES FROM A STRONG RECEIVER

**E** *(optional)* Specify the name of an inbound migrate profile or a wildcard pattern. Then, press **Enter**.

The CHANGE MANAGER Datasets panel is displayed.

**F** Specify the CDL file and the data set names to be used by the Import, Analysis, and Execution components of CHANGE MANAGER. Then, press **Enter** until the TASKID Interface panel is displayed.

6 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type **S** to select **Edit Worklist**.

B To create the JCL to run the worklist, type **S** to select **Create JCL**.

C To review or modify the JCL, type **S** to select **Edit JCL**.

D To submit the JCL to run the CM/PILOT worklist, type **S** to select **Submit JCL**.

E Press **Enter**.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

7 To receive the data structure changes, execute the Analysis worklist.

---

**NOTE**

The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for **JCL Dataset Name** in the CHANGE MANAGER Datasets panel.

For more information about executing Analysis worklists, see “Executing an alter-type worklist” on page 562.

---

**RECEIVING DATA STRUCTURE CHANGES FROM A STRONG RECEIVER**

The CM/PILOT REC_CHG_SR_BL script contains the steps to import only the changes that are required to upgrade an application’s data structures to a new version and to create an Analysis worklist to apply the changes. The new version of the data structures must be contained in a DDL file or worklist that was generated by CHANGE MANAGER. Local modifications that you have made to the existing version of the data structures are retained in the new version.

In this task, you will create and execute task ID.
Before you begin

The following CHANGE MANAGER objects are required for this script:

- a DDL baseline from the existing version of the data structures
- a DDL baseline profile
- an inbound migrate profile, if you want to use its change rules
- a new version of data structures contained in a DDL file or worklist that was generated by CHANGE MANAGER

To create a task ID

1. On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.
   
   The CM/PILOT Main Menu is displayed.

2. Select **TASKIDs** and press **Enter**.

   The TASKID Action Menu is displayed.

3. Type the name of a new **TASKID** and select **Create a TASKID**. Then, press **Enter**.

   The Script Selection List is displayed.

4. Select **Receive data structure changes (strong receiver), create Baseline** and then press **Enter**.

   The Create TASKID panel is displayed.

5. Specify the information for a task ID.

   A. **CM/PILOT** uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for **WORKID** or type the name of a work ID name template.

   B. *(optional)* Type the name of a CM/PILOT **Application**. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   **TIP**

   If you frequently use this script to receive changes for the same DB2 application, you can create an inbound migrate profile with the change rules that you need, define it to a CM/PILOT Application, and then specify the Application each time that you create a task ID for this job.

   C. *(optional)* Specify a **Comment** to describe the task ID.
Receiving data structure changes from a strong receiver

D Press Enter. Press END.

The task ID is created and the TASKID Action Menu panel is displayed.

To execute the task ID

1 Select Execute a TASKID and press Enter.

The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

A Type the data set name for the CM/PILOT Worklist.

B Type the data set name for the CM/PILOT JCL.

C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

A Type S to select Override CHANGE MANAGER options and then press Enter.

The Override CHANGE MANAGER Options panel is displayed.

B Select Override CHANGE MANAGER Options.

C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

WARNING

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.
5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B Type S to select the steps in the script that you want to modify, and press Enter.

C To display subsequent Step Settings panels, press Enter.

After the Step Settings panels are displayed, the Taskid Compare CDL Options panel is displayed.

6 On the Taskid Compare CDL Options panel, specify the object types and attributes for the comparison.

A Type Y or N for each object type that you want to include in or exclude from the comparison.

B Type S to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

C If you selected to display a list of the object attributes, specify the object attributes.

1. To include an object attribute in the comparison, type Y adjacent to the attribute.

2. If you specified to include the partition attributes, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

3. Press Enter to return to the Taskid Compare CDL Options panel.

D Press Enter.

The TASKID Migrate Profiles panel is displayed.

E (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

F (optional) Specify the name of a second outbound migrate profile. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects before the comparison.
G (optional) Specify the name of an inbound migrate profile. The inbound migrate profile can contain change rules. It can be used when the CDL that is generated from the comparison is imported.

H Press Enter.

The TASKID Baseline Profiles panel is displayed.

I On the TASKID Baseline Profiles panel, type the name of the Baseline Profile to be used in this task ID or type a wildcard pattern and select a profile from the list. Then, press Enter.

The TASKID New Baselines panel is displayed.

J On the TASKID New Baselines panel, specify the information for the new baseline.

**NOTE**

If the baseline profile that you specified in step I on page 594 uses a baseline profile template, the name of the template is displayed. To use the baseline profile template, do not specify the name of the baseline. To override the baseline profile template, type a name or a template for Baseline Name.

If you are using baseline name templates to create names for baselines, the latest baseline that is created from the baseline profile that is specified for this script is used in the comparison step.

1. (optional) Type the name of a new baseline.

2. To protect the baseline from deletion, type Y to select Protect.

3. (optional) Specify Comments to describe the baseline.

4. Press Enter.

The TASKID Existing Baselines panel is displayed.

K Specify the name of the baseline and press Enter.

The CHANGE MANAGER Datasets panel is displayed.

L Specify the DDL and CDL files, and the data set names to be used by the Import, Analysis, and Execution components of CHANGE MANAGER. Then, press Enter until the TASKID Interface panel is displayed.
7 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

8 To receive the data structure changes, execute the Analysis worklist.

**NOTE**
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.

For more information about executing Analysis worklists, see “Executing an alter-type worklist” on page 562.

### Restoring data structures with current data

You can restore a previous version of your data structures from a catalog baseline and use the data that is currently stored in DB2. This process is sometimes referred to as “falling back” to a previous version. To automate this process with a script, see “Recovering data structures with current data” on page 596.

**Before you begin**

Ensure that a catalog baseline exists. For information about catalog baselines, see Chapter 2, “Taking a snapshot of a database environment.”
Recovering data structures with current data

To restore data structures

1. Compare the DB2 catalog to the baseline. See “Comparing a DB2 catalog to a baseline” on page 290.

2. Import the CDL from the comparison in step 1 to an alter-type work ID.
   A. Create an alter-type work ID. See “Creating an alter-type work ID” on page 357 for more information.
   B. Import the CDL to the work ID that you created in step 2A. For more information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.

3. Analyze the alter-type work ID and generate a worklist. For more information, see “Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560.

   **NOTE**
   On the Analysis Alter Worklist Interface panel, select Generate a worklist which will convert and reload current data. If the scope rules for a baseline profile have changed since the last baseline was built, Analysis issues a warning message in the diagnostic output file.

4. Execute the worklist. For more information, see “Executing an alter-type worklist” on page 562.

Recovering data structures with current data

The CM/PILOT RCV_STRUC_CUR_DATA script contains the steps to recover data structures from a catalog baseline and use the data that is currently stored in DB2. The data is converted and reloaded into the recovered data structures.

To accomplish this task, the product compares the DB2 catalog to a baseline and generates a CDL file if differences exist between the data structures. If the CHANGE MANAGER Compare component finds no differences, CM/PILOT worklist processing stops and an Analysis worklist is not created. If differences exist, a CDL file is created and imported, and an Analysis worklist is created to apply the changes.

In this task, you will create and execute task ID.
Before you begin

Using CHANGE MANAGER, create a catalog baseline profile and a catalog baseline. For information, see “Creating a catalog baseline profile” on page 198 and “Creating a structure-only catalog baseline” on page 201.

To create a task ID

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3. Type the name of a new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

4. Select Recover data structures with current data and press Enter.

   The Create TASKID panel is displayed.

5. Specify the information for a task ID.

   A. CM/PILOT uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for WORKID or type the name of a work ID name template.

   B. (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C. (optional) Specify a Comment to describe the task ID.

   D. Press Enter. Then, press END.

   The task ID is created and the TASKID Action Menu panel is displayed.
To execute the task ID

1. Select Execute a TASKID and press Enter.

   The TASKID Interface panel is displayed.

2. On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.
   
   A. Type the data set name for the CM/PILOT Worklist.
   
   B. Type the data set name for the CM/PILOT JCL.
   
   C. Type the data set name for Diagnostics or type SYSOUT.

3. Select the method of executing the CM/PILOT worklist.

   For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4. (optional) Override the options that are set in CHANGE MANAGER.

   A. Type S to select Override CHANGE MANAGER options and then press Enter.

      The Override CHANGE MANAGER Options panel is displayed.

   B. Select Override CHANGE MANAGER Options.

   C. Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

      For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

---

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D. Press END until the TASKID Interface panel is displayed.
5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select **Create Worklist**. Then, press **Enter**.

The Step Settings panel is displayed.

B *(optional)* To modify the Analysis step, type S to select **Step 2 Analysis step** and press **Enter**.

C *(optional)* To modify the Execution step, type S to select **Step 3 Execution step** and press **Enter**.

D Press **Enter** until the Taskid Compare CDL Options panel is displayed.

6 On the Taskid Compare CDL Options panel, specify the object types and attributes for the comparison.

A Type Y or N for each object type that you want to include in or exclude from the comparison.

B Type S to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

C If you selected to display a list of the object attributes, specify the object attributes.

1. To include an object attribute in the comparison, type Y adjacent to the attribute.

2. If you specified to include the partition attributes, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

3. Press **Enter** to return to the Taskid Compare CDL Options panel.

D Press **Enter**.

The TASKID Migrate Profiles panel is displayed.
On the TASKID Migrate Profiles panel, specify the migrate profiles.

A (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

B (optional) Specify the name of a second outbound migrate profile. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects before the comparison.

C (optional) Specify the name of an inbound migrate profile. The inbound migrate profile can contain change rules. It can be used when the CDL that is generated from the comparison is imported.

D Press Enter.

The TASKID Existing Baselines panel is displayed.

E Specify the name of the baseline and press Enter.

The CHANGE MANAGER Datasets panel is displayed.

F Specify the CDL file and data set names to be used by the Import, Analysis, and Execution components of CHANGE MANAGER. Then, press Enter until the TASKID Interface panel is displayed.

On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.
To recover the data structures, execute the Analysis worklist.

**NOTE**
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for **JCL Dataset Name** in the CHANGE MANAGER Datasets panel.

For more information about executing Analysis worklists, see “Executing an alter-type worklist” on page 562.

**Restoring data structures with old data**

You can restore a previous version of your data structures from a full-recovery baseline and load the data that was unloaded when the full-recovery baseline was created. This process is sometimes referred to as “falling back” to a previous version. To automate this process with a script, see “Recovering data structures with old data” on page 602.

**Before you begin**

Ensure that a full-recovery baseline exists. For information about full-recovery baselines, see Chapter 2, “Taking a snapshot of a database environment.”

**To restore data structures**

1. Compare the DB2 catalog to the full-recovery baseline. See “Comparing a DB2 catalog to a baseline” on page 290.

2. Import the CDL from the comparison in step 1 to an alter-type work ID.

   A. Create an alter-type work ID. For more information, see “Creating an alter-type work ID” on page 357.

   B. Import the CDL to the work ID that you created in step 2A. For more information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 379.
3 Analyze the alter-type work ID and generate a worklist. For more information, see “Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 560.

**NOTE**

From the Analysis Alter Worklist Interface panel, select **Generate a worklist which will reload data from a previous full-recovery baseline**. If the scope rules for a baseline profile have changed since the last baseline was built, Analysis issues a warning message in the diagnostic output file.

All of the data that is saved in the baseline is recovered.

4 Execute the worklist. For more information, see “Executing an alter-type worklist” on page 562.

**Recovering data structures with old data**

The CM/PILOT RCV_STRUC_OLD_DATA script contains the steps to recover data structures from a full-recovery baseline and load the data that was unloaded when the baseline was created. The old data is loaded into the recovered data structures. All of the data that was saved in the original full-recovery baseline and the changed data structures are loaded into the tables. The referential integrity of the application’s data for the given point in time is maintained.

To accomplish this task, the product compares the DB2 catalog to a full-recovery baseline and generates a CDL file if differences exist between the data structures. If the CHANGE MANAGER Compare component finds no differences, CM/PILOT worklist processing stops and an Analysis worklist is not created. If differences exist, a CDL file is created and imported, and an Analysis worklist is created to apply the changes.

In this task, you will create and execute task ID.

**Before you begin**

Using CHANGE MANAGER, create a catalog baseline profile and a full-recovery baseline. For information, see “Creating a catalog baseline profile” on page 198 and “Creating a full-recovery baseline” on page 203.
To create a task ID

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.
   The TASKID Action Menu is displayed.

3 Type the name of a new TASKID and select Create a TASKID. Then, press Enter.
   The Script Selection List is displayed.

4 Select Recover data structures with old data and press Enter.
   The Create TASKID panel is displayed.

5 Specify the information for a task ID.
   
   A CM/PILOT uses the name of the task ID as the name for a new alter-type work ID. You can type a different name for WORKID or type the name of a work ID name template.

   B (optional) Type the name of a CM/PILOT Application. An Application groups a set of CHANGE MANAGER profiles. For more information about Applications, see Chapter A, “Using scripting tools to automate change management.”

   C (optional) Specify a Comment to describe the task ID.

   D Press Enter. Then, press END.
   
   The task ID is created and the TASKID Action Menu panel is displayed.

To execute the task ID

1 Select Execute a TASKID and press Enter.
   The TASKID Interface panel is displayed.

2 On the TASKID Interface panel, specify the data set names to be used by CM/PILOT.

   A Type the data set name for the CM/PILOT Worklist.

   B Type the data set name for the CM/PILOT JCL.
Recovering data structures with old data

C Type the data set name for Diagnostics or type SYSOUT.

3 Select the method of executing the CM/PILOT worklist.

For more information about executing a CM/PILOT worklist, see “Processing CM/PILOT worklists” on page 642.

4 (optional) Override the options that are set in CHANGE MANAGER.

A Type S to select Override CHANGE MANAGER options and then press Enter.

The Override CHANGE MANAGER Options panel is displayed.

B Select Override CHANGE MANAGER Options.

C Type S to select the CHANGE MANAGER Analysis and Execution options that you want to override, and press Enter.

For more information about the CHANGE MANAGER options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

WARNING
BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

D Press END until the TASKID Interface panel is displayed.

5 On the TASKID Interface panel, create a CM/PILOT worklist.

A Type S to select Create Worklist. Then, press Enter.

The Step Settings panel is displayed.

B (optional) To modify the Execution step, type S to select Step 3 Execution step and press Enter.

C Press Enter until the Taskid Compare CDL Options panel is displayed.
6 On the Taskid Compare CDL Options panel, specify the object types and attributes for the comparison.

A Type Y or N for each object type that you want to include in or exclude from the comparison.

B Type S to display a list of the object attributes to include in or exclude from the comparison.

The Compare Object Attributes panel is displayed.

C If you selected to display a list of the object attributes, specify the object attributes.

1. To include an object attribute in the comparison, type Y adjacent to the attribute.

2. If you specified to include the partition attributes, you can exclude one or more of the partition attributes. Type N adjacent to the attributes that you want to exclude.

3. Press Enter to return to the Taskid Compare CDL Options panel.

D Press Enter.

The TASKID Migrate Profiles panel is displayed.

7 On the TASKID Migrate Profiles panel, specify the migrate profiles.

A (optional) Specify the name of an outbound migrate profile. The outbound migrate profile can contain change rules and locations. CHANGE MANAGER applies the rules after it performs the comparison (when it generates the CDL).

B (optional) Specify the name of a second outbound migrate profile. The outbound migrate profile must contain change rules to resolve the names of the objects and creators. CHANGE MANAGER applies the change rules to the objects before the comparison.

C (optional) Specify the name of an inbound migrate profile. The inbound migrate profile can contain change rules. It can be used when the CDL that is generated from the comparison is imported.

D Press Enter.

The TASKID Existing Baselines panel is displayed.
E Specify the name of the full-recovery baseline and press Enter.

The CHANGE MANAGER Datasets panel is displayed.

F Specify the CDL file and the data set names to be used by the Import, Analysis, and Execution components of CHANGE MANAGER. Then, press Enter until the TASKID Interface panel is displayed.

8 On the TASKID Interface panel, select your options for processing the CM/PILOT worklist.

A To edit the CM/PILOT worklist, type S to select Edit Worklist.

B To create the JCL to run the worklist, type S to select Create JCL.

C To review or modify the JCL, type S to select Edit JCL.

D To submit the JCL to run the CM/PILOT worklist, type S to select Submit JCL.

E Press Enter.

CM/PILOT creates an Analysis worklist and the Execution JCL that is required to run the worklist.

9 To recover the data structures, execute the Analysis worklist.

**NOTE**

The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for JCL Dataset Name in the CHANGE MANAGER Datasets panel.

For more information about executing Analysis worklists, see “Executing an alter-type worklist” on page 562.
Where to go from here

This chapter discussed how you could migrate data structures, establish baselines, compare data structures, and import external files. For more information on these topics, see the following table.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate database environments</td>
<td>Chapter 1</td>
</tr>
<tr>
<td>Take a snapshot of a database environment</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>Compare a database environments</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Create or alter objects in a database environment</td>
<td>Chapter 4</td>
</tr>
</tbody>
</table>
Managing your PeopleSoft environment with CHANGE MANAGER

This appendix presents the following topics:

Overview ................................................................. 609
Working in a PeopleSoft environment ............................... 610
Meeting the challenges of a PeopleSoft environment .......... 611
   Taking a snapshot of the initial environments ................. 611
   Creating copies of an existing environment ................... 613
   Optimizing the design of an environment ..................... 618

Overview

Companies throughout the world are using Enterprise Resource Planning (ERP) systems to consolidate various business processes and functions, such as human resources, finance, distribution, and manufacturing. PeopleSoft is one such ERP system. At the core of a PeopleSoft system is a very large relational database application in which thousands of tables are integrated to allow many users at different locations to perform varied tasks.

Administering a database application this large can be a daunting task. Fortunately, BMC provides the tools to ensure the availability of your PeopleSoft system by simplifying the administrative tasks. This appendix describes how you can use the current version of the CHANGE MANAGER product to manage your PeopleSoft environments on OS/390 or z/OS. This appendix assumes that you are already familiar with PeopleSoft.
Working in a PeopleSoft environment

The PeopleSoft application suite consists of the following applications:

- Human Resources Management (HRMS)
- Financials (FIN)
- Supply Chain Management (SCM)
- Customer Relationship Management (CRM)

An initial implementation of one of these applications consists of a very large database. This database contains a few table spaces and thousands of tables. These tables might contain more than 20 object types and more than 15,000 objects. For example, each testing and production environment for a large distribution warehouse might contain the following objects:

- 50 databases
- 400 table spaces
- 10,000 tables
- 7,000 indexes
- 10,000 views

A PeopleSoft database contains three categories of tables:

- DB2 system catalog tables contain characteristics of tables, columns, and views, and store indexes.

- PeopleTools tables, which are internal tables that PeopleSoft supplies, contain object-related data, such as definitions of fields, records, panels, menus, and projects. You use PeopleSoft tools such as the Application Designer to modify these tables.

- Application (data) tables contain data that the user creates. You can use CHANGE MANAGER to modify the definition of these tables.

As a result of having a large number of tables, objects, and transactions, the PeopleSoft database administrator (DBA) faces many challenges:

- creating copies of an existing environment
  - making mass changes to data definition language (DDL) attributes
  - maintaining the data integrity of the PeopleSoft application
  - identifying dependent objects
  - managing complicated views
Meeting the challenges of a PeopleSoft environment

The BMC CHANGE MANAGER for DB2 product can help you meet the challenges of your PeopleSoft environment. The CHANGE MANAGER product enables you to:

- take a snapshot of your initial demonstration and system environments
- create copies of an existing environment by copying both the data structures and data from one subsystem to another or within the same subsystem
- optimize the design of your application databases in an efficient and error-free manner by automating complex changes to data structures

This section describes how you can use CHANGE MANAGER to meet these challenges.

Taking a snapshot of the initial environments

Your company just purchased an ERP system. You need to implement the PeopleSoft application suite and create a demonstration (DMO) database environment and a system (SYS) environment.

- The DMO environment includes PeopleTools tables that are loaded with data and application data tables that are loaded with data supplied by PeopleSoft. This environment is not customized; it is used only as a demonstration of the functionality of PeopleSoft.

- The SYS environment includes PeopleTools tables that are loaded with data and application data tables that are not loaded with data. This environment is used as the basis for development and production environments.

To create the environments, you perform vanilla implementations of the applications (that is, you implement the applications without customizing them or modifying the supplied scripts or DDL). You perform these vanilla implementations by using tools that are delivered with PeopleSoft.
After you create the DMO and SYS environments, you can use CHANGE MANAGER to take a snapshot (that is, create a baseline) of the DB2 catalog in each of the environments. CHANGE MANAGER enables you to establish full-recovery baselines to capture data structures and data at a specific point in time. These baselines act as control points in the change migration process and establish a static set of data structures for an application version.

**Tasks to perform using CHANGE MANAGER**

Baseline tables (named BL_<name>) are stored in CHANGE MANAGER. These tables hold critical information that is used to recover database structures. If you regularly establish baselines, these baseline tables grow over time. You must ensure that these tables are reorganized and that the size of the tables is expanded on a regular basis.

In addition, CHANGE MANAGER stores the table data in UNLOAD data sets and stores the specified unload options with the baseline. You must retain the UNLOAD data sets for as long as a recovery point might be needed.

**Take a snapshot of an environment**

Perform the following tasks in CHANGE MANAGER to take snapshots of the initial environments:

1. Create a DB2 catalog baseline profile that uses scope rules for the DMO environment.

   A baseline profile contains information about how baselines are named and established and may also contain scope rules. Scope rules indicate which DB2 objects to include in a baseline. A baseline profile can be created for each environment and can be used repeatedly.

2. Create a DB2 catalog baseline of the DMO environment, using the catalog baseline profile to define the scope.

3. Create a DB2 catalog baseline profile that uses scope rules for the SYS environment.

4. Create a DB2 catalog baseline of the SYS environment, using the second catalog baseline profile to define the scope.

   You can also use CHANGE MANAGER to generate baseline reports from the baselines that you created. A baseline report includes general information about the baseline, DDL and commented-out GRANT statements for each object, and the unload data set names that are associated with a full-recovery baseline.
Creating copies of an existing environment

When you first implemented your PeopleSoft application suite, you created DMO and SYS environments. PeopleSoft suggests that you create several environments for upgrades, development, testing, and production, and so on. Table 43 lists these environments and the abbreviation that PeopleSoft uses to identify them.

You now need to copy the SYS environment (that is, the initial data structures and data) to create a DEV environment. You can later use the DEV environment to create CONV, TST, and PROD environments. In CHANGE MANAGER, the process of copying or creating the data structures and data is called migration.

How the process works

Figure 175 illustrates the process of creating copies of an existing environment.
Creating copies of an existing environment

Figure 175  Creating copies of an environment

Tasks to perform using CHANGE MANAGER

CHANGE MANAGER contains internal SQL that is tuned to provide increased performance. To leverage these performance settings, you must create some indexes on the DB2 catalog. For information about creating these indexes, see the BMIDB2XA or BMIDB2X9 member in the HLQ.UDBCNTL data set.
Create a copy of the environment

Perform the following tasks in CHANGE MANAGER to create copies of an existing environment:

1. Create an outbound migrate profile that includes the scope of the migration, change rules, and locations.

   - **Scope**

     To begin the migration process, you need to determine the scope of the migration. Which parent objects do you want to migrate? Which dependent objects? If you plan to migrate your application’s data structures often, you can use the scope of an outbound migrate profile to automate the object selection process.

   - **Change rules**

     If you have a naming standard or convention for tables, views, or other PeopleSoft objects, you can specify the conventions with change rules in a CHANGE MANAGER outbound migrate profile. The change rules automatically change the value of an attribute of an existing data structure. Change rules are commonly used to change the owner of an object or the name of an object to avoid duplicate names. For example, you might want to change the owner of a table in your SYS environment from HR8.SALPLAN to HR8SYS.SALPLAN in your DEV environment.

     To reduce the creation of empty data sets, create change rules for your table spaces and their dependent indexes that set the DEFINE attribute to NO. When DEFINE is set to NO, DB2 does not create the data sets for a table space or an index until a row is inserted or loaded into a table.

---

**TIP**

After DB2 has created the data sets for the table spaces and indexes, you can also use the CM/PILOT DML_STRUCTURE_CHG script to set one or more table spaces and their dependent indexes to DEFINE NO. In addition, you can use the script to update one or more table spaces and their dependent indexes to DEFINE NO based on the cardinality of 0 of the underlying tables. You can create the statements yourself or you can modify the DML statements in the ACMDMLU1 or ACMDMLU4 member of the HLQ.DBCNTL data set. Run statistics on the environment before you execute the script.
Creating copies of an existing environment

- **Locations**

  If you want to migrate the structures to more than one environment (for example, from DEV to CONV, TST, and PROD), you can specify locations in the outbound migrate profile. Each location represents an environment to which you want to migrate data structures. The locations can have change rules that tailor the worklist to the environment. You can generate a worklist for each location or use the same worklist for multiple locations.

  **NOTE**

  Generating a worklist for each location or using the same worklist for multiple locations is not recommended for migrations that have a large scope.

2. Create a migrate-type work ID.

3. Estimate the required space.

   When you create your environment, you need to determine the amount of space that a table space or index requires based on the data structure definitions and their usages. You can use the space estimation feature of CHANGE MANAGER to estimate the space for table spaces and indexes. CHANGE MANAGER also enables you to propagate the number of rows, allocation unit, and estimated primary and secondary space to other partitions in a partitioned table space or index.

   **TIP**

   Alternatively, you can use the CM/PILOT SET SPACE ESTIMATION DML statement to estimate the space for all table spaces and indexes in a database. The statement can appear anywhere in the CM/PILOT DML_STRUCTURE_CHG script or the DML_MIGRATE script and can apply to all of the DML statements in the script.

4. Specify your Analysis options.

   You should specify the following options for the migration:

   - Sync point frequency

     The sync point frequency (ALUIN keyword SYNCPOINT parameter) is an Analysis worklist option that creates additional -SYNC commands in a worklist, based on the number of -SQL commands since the last -SYNC command. The -SYNC command creates a sync table entry and executes a DB2 COMMIT WORK statement. The default value of parameter is 10.
Creating copies of an existing environment

If you have to terminate the worklist for any reason, changing the sync point frequency to increase the number of COMMIT WORK statements in the worklist during the Analysis of the work ID reduces the amount of time that DB2 spends performing a rollback operation.

- Dynamic allocation of image copy and unload data sets

The dynamic image copy (COPY) and dynamic unload (SYSRE) data set allocation options (ALUIN keywords DYNCOPY and DYNUNLD) are Analysis utility data set options that cause the data sets to be dynamically allocated. For information about naming conventions for the prefix for the data set, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.*

5 Create the JCL or the input for Analysis.

After you specify your Analysis options, you should edit the ALUIN input stream in your JCL or your input to Analysis:

- Add the BMCFASTLOAD keyword to use the fast unload and load (high-speed data migration) feature of the BMC UNLOAD PLUS and LOADPLUS utilities.

When you use the UNLOAD PLUS and LOADPLUS utilities with CHANGE MANAGER, you can quickly unload data from one table and load it into another table that has a similar structure (encoding scheme) and that has the same CCSID definitions.

**NOTE**

The BMCFASTL parameter might be set to N in the installation option module. If so, you can manually change the NOBMCFASTLOAD keyword in the ALUIN input stream to BMCFASTLOAD to use this feature.

- Specify DISCARDS 1 to reduce the number of allowable discard data sets and to stop the execution of the worklist as soon as a row is discarded.

By default, CHANGE MANAGER creates a discard data set for each unload (SYSREC) data set. Using this default, CHANGE MANAGER might have to size and allocate thousands of data sets in a PeopleSoft environment. Setting the DISCARDS keyword to 1 in the ALUIN input stream causes CHANGE MANAGER to create one output file in the JCL for discarded records and to stop the execution of a worklist if a row is discarded.
6 Submit the JCL or run Analysis to analyze the migrate-type work ID and generate a worklist.

When data is loaded by the worklist commands, the existing VSAM data sets for the table space and index spaces can be deleted and redefined. When the LOADPLUS utility is used to load data, the CHANGE MANAGER product automatically sets the REDEFINE option to NO in the -BMCL worklist command.

7 Generate the JCL to run the worklist.

8 Submit the JCL to execute the worklist.

9 Create a DB2 catalog baseline profile that uses scope rules.

After you migrate the SYS environment to a DEV environment or migrate the DEV environment to another environment, create a catalog baseline profile for the new environment. You can specify scope rules in the baseline profile that indicate which DB2 objects to include in a baseline.

10 Create a DB2 catalog baseline of the new environment, using the catalog baseline profile to define the scope.

For more information

For more information about creating a copy of an environment, see Chapter 1, “Migrating database environments.”

Optimizing the design of an environment

Your PeopleSoft applications might consist of one database, a few table spaces, and thousands of tables. To manage the applications more efficiently and effectively, you can divide the physical design of the system into separate and smaller segments. For example, you can perform the following tasks:

- create a new database and table spaces for tables

Many table spaces in a PeopleSoft application contain thousands of tables. By moving any tables that equal a specified cardinality to a new database in which each table resides in a single table space, you can reduce the size of the DB2 database descriptor (DBD) for each database. DBDs contain definitions of table spaces, tables, indexes, and object relationships in a DB2 database. The size of the DBD is based on an estimate of the number of columns per table and the number of tables per database. By reducing the number of the tables contained in each database, you can reduce the size of the DBD.
- move all of the empty tables in a database to one table space

Many tables in a PeopleSoft application might be empty or might not be used. Taking image copies of these tables is time consuming and costly. By moving the empty tables to one table space, you can take image copies of only the table spaces that contain tables with data.

To accomplish these tasks, you can use specially designed scripts in the CM/PILOT component of CHANGE MANAGER.

**How the process works**

Figure 176 illustrates the process of optimizing the design of an environment.

**Figure 176 Optimizing the design of the environment**
Tasks to perform using CHANGE MANAGER

The DML_STRUCTURE_CHG script in CM/PILOT contains the steps to create an Analysis worklist to update and delete data structures on a local DB2 subsystem. This script uses a Data Manipulation Language (DML) trigger to determine whether data structures exist. If data structures are found, an Analysis worklist is created to apply the changes.

Create a new database and table spaces for tables

Perform the following tasks in CHANGE MANAGER to create a new database and table spaces:

1 Run statistics on the PeopleSoft environment.

2 Create a baseline of the data structures.

Before you modify the design of your environment, establish a full-recovery baseline of the data structure definitions and data in your environment. If you have any problems after you modify the design of the system, you can easily recover to this version by using the baseline as the recovery point. When you establish the baseline, specify the Do not unload tables Analysis worklist option (ALUIN keyword NOUNLOADEMPTY). This option bypasses the unloading and loading of empty tables, thus saving a tremendous amount of time in an ERP environment.

3 Create a task ID in the CM/PILOT component of CHANGE MANAGER.

4 Use a script to create DML statements that create a new database and table spaces for tables.

When you use the DML_STRUCTURE_CHG script, you can specify the following information:

- the primary quantity for the new table space

  CHANGE MANAGER uses its space estimation feature to explicitly estimate the space for only the table spaces that are specified in the LIKE TABLEPART DML statement.

- the cardinality of the table

  Only tables that have the number of rows that you specify are reassigned to a new table space.

You can create the statements yourself, or you can modify the DML statements in the ACMDMLU2 member of the HLQ.DBCNTL data set.
5 Execute the task ID and to create a CM/PILOT worklist and the JCL to run the worklist.

6 Execute the CM/PILOT worklist to create an Analysis worklist and the Execution JCL to run the worklist.

7 Execute the Analysis worklist.

8 Create a baseline of the changed environment.

**Move all empty tables in a database to a table space**

Perform the following tasks in CHANGE MANAGER to move all of the empty tables in a database to one table space:

1 Run statistics on the PeopleSoft environment.

2 Create a baseline of the data structures.

Before you modify the design of your environment, establish a baseline of the data structure definitions in your environment. If you have any problems after you modify the design of the system, you can easily recover to this version by using the baseline as the recovery point. When you establish the baseline, specify the **Do not unload tables** Analysis worklist option (ALUIN keyword NOUNLOADEMPTY). This option bypasses the unloading and loading of empty tables, thus saving a tremendous amount of time in an ERP environment.

3 Create a task ID in the CM/PILOT component of CHANGE MANAGER.

4 Use a script to create DML statements that move all of the empty tables in a database to one table space.

When you use the DML_STRUCTURE_CHG script, the UPDATE TABLE DML statement in the script ensures that the table is not part of a partitioned table space and that the table is empty. You can create the statements yourself, or you can modify the DML statements in the ACMDMML2 member of the HLQ.DBCNTL data set.

**TIP**
You can also move empty tables to one table space by using the Specification component of CHANGE MANAGER. Identify the tables that are empty. Then, on the Mixed List panel, use the QE action command on a table and specify the name of the existing database and table space.

5 Execute the task ID and to create a CM/PILOT worklist and the JCL to run the worklist.
For more information

For more information about optimizing the design of an environment, see Chapter 4, “Altering data structures in a database environment.”
Part 3 Using tools and utilities with ALTER and CHANGE MANAGER

This part presents the following topics:

Appendix A
   Using scripting tools to automate change management ......................... 625

Appendix B
   Using utilities with ALTER and CHANGE MANAGER  ......................... 673

Appendix C
   Using the skeleton library compiler ............................................. 719

Appendix D
   Using worklist parallelism ......................................................... 727
# Using scripting tools to automate change management

This appendix presents the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>626</td>
</tr>
<tr>
<td>Using task IDs</td>
<td>627</td>
</tr>
<tr>
<td>Creating a task ID</td>
<td>628</td>
</tr>
<tr>
<td>Listing task IDs</td>
<td>630</td>
</tr>
<tr>
<td>Editing a task ID</td>
<td>631</td>
</tr>
<tr>
<td>Executing a task ID</td>
<td>632</td>
</tr>
<tr>
<td>Viewing the execution status of a task ID</td>
<td>639</td>
</tr>
<tr>
<td>Browsing a task ID</td>
<td>640</td>
</tr>
<tr>
<td>Deleting a task ID</td>
<td>641</td>
</tr>
<tr>
<td>Processing CM/PILOT worklists</td>
<td>642</td>
</tr>
<tr>
<td>Restarting a CM/PILOT worklist</td>
<td>642</td>
</tr>
<tr>
<td>Starting a CM/PILOT worklist over</td>
<td>644</td>
</tr>
<tr>
<td>Using scripts</td>
<td>646</td>
</tr>
<tr>
<td>Creating a script</td>
<td>648</td>
</tr>
<tr>
<td>Listing scripts</td>
<td>659</td>
</tr>
<tr>
<td>Editing a script</td>
<td>660</td>
</tr>
<tr>
<td>Listing the steps for a script</td>
<td>661</td>
</tr>
<tr>
<td>Executing a user-defined script</td>
<td>662</td>
</tr>
<tr>
<td>Browsing a script</td>
<td>662</td>
</tr>
<tr>
<td>Deleting a script</td>
<td>663</td>
</tr>
<tr>
<td>Creating a task ID from a script</td>
<td>664</td>
</tr>
<tr>
<td>Listing task IDs for a script</td>
<td>665</td>
</tr>
<tr>
<td>Copying a script supplied by BMC</td>
<td>665</td>
</tr>
<tr>
<td>Copying a user-defined script</td>
<td>666</td>
</tr>
<tr>
<td>Using applications</td>
<td>666</td>
</tr>
<tr>
<td>Creating an application</td>
<td>668</td>
</tr>
<tr>
<td>Listing applications</td>
<td>669</td>
</tr>
<tr>
<td>Editing an application</td>
<td>669</td>
</tr>
<tr>
<td>Browsing an application</td>
<td>670</td>
</tr>
<tr>
<td>Deleting an application</td>
<td>671</td>
</tr>
</tbody>
</table>
The CM/PILOT component of the CHANGE MANAGER product enables you to quickly and easily utilize the vast change management power of CHANGE MANAGER. By following the panels that are provided in the CM/PILOT component’s scripts or in a script that you create, you can specify, analyze, and execute a worklist to accomplish the following tasks:

- change data structures
- migrate data structures
- receive data structures
- receive DDL to create data structures
- create full-recovery baselines
- recover data structures with current or old data
- create a user-defined script
- replicate work IDs

CM/PILOT accomplishes these tasks by using the following features:

- task IDs, which are units of work with unique identifiers
- worklists, which are data sets that contain commands for CHANGE MANAGER components and information that is provided through a dialog
- scripts, which are dialogs that enable you to combine various CHANGE MANAGER functions into one job
- Data Manipulation Language (DML), which is an SQL-like language that is used by CM/PILOT to update, delete, and migrate data structures

DML uses syntax that is slightly different than DB2 DML. For information about the DML used by CM/PILOT, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.

- applications, which are sets of CHANGE MANAGER profiles that can be used repeatedly for the change management tasks of a specific DB2 application
When you want to perform a change management task, CM/PILOT takes care of the processes that CHANGE MANAGER will use and ensures that they are performed in the correct sequence. CM/PILOT enables you to set forth tasks that can be performed later by someone else or through job scheduling. By reusing the task ID, you can ensure that the change management task is performed the same way every time.

When you use CM/PILOT, you can easily prepare for various CHANGE MANAGER tasks that can be processed on demand. The CM/PILOT dialog is task-oriented and easy to use. It describes the change management tasks that can be performed and directs you to provide information for preparing worklists that can be processed at any time.

This appendix discusses each of the preceding features (with the exception of DML) and how you can use them. Tasks for using the CM/PILOT component to perform a specific change management function are described in the appropriate chapters of this book.

**NOTE**
For more information, view the Quick Course CM/PILOT Overview. You must have a BMC Support ID to view the Quick Course.

---

**Using task IDs**

A task ID is a unit of work in the CM/PILOT component. Each task ID has a unique name and contains information that you provide through the dialog to perform a CHANGE MANAGER process. The information that you need to provide is based on the script that you select for the task ID. All work in CM/PILOT is accomplished by processing task IDs. Actions that are performed with task IDs usually begin at the TASKID Action Menu.

Table 44 describes the actions that you can perform with task IDs. The procedural steps that are needed to perform actions, along with the related information, are described.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating a task ID”</td>
<td>page 628</td>
</tr>
<tr>
<td>“Listing task IDs”</td>
<td>page 630</td>
</tr>
<tr>
<td>“Editing a task ID”</td>
<td>page 631</td>
</tr>
<tr>
<td>“Executing a task ID”</td>
<td>page 632</td>
</tr>
<tr>
<td>“Viewing the execution status of a task ID”</td>
<td>page 639</td>
</tr>
</tbody>
</table>
Creating a task ID

You can create a task ID. Creating a task ID is the first step in using the CM/PILOT component for performing CHANGE MANAGER processes. All work in CM/PILOT is accomplished by processing a task ID.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3 Type a name for the new TASKID and select Create a TASKID. Then, press Enter.

   The Script Selection List is displayed.

4 Select a script and press Enter.

   The Create TASKID panel is displayed.

5 CM/PILOT uses the name of the task ID as the name for a new work ID. You can type a different name for WORKID or type the name of a work ID name template.

   **NOTE**

   Every Task ID requires a work ID. The script that you use determines the type of work ID that you need. CM/PILOT creates the correct type of task ID for a new work ID.

**Table 44  Task ID tasks (part 2 of 2)**

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Browsing a task ID”</td>
<td>page 640</td>
</tr>
<tr>
<td>“Deleting a task ID”</td>
<td>page 641</td>
</tr>
</tbody>
</table>

**NOTE**

The abbreviated script names that are listed in the table are displayed on various panels in the dialog.
When you specify a name for a new work ID, you can use a name template. The template characters are resolved when the worklist is processed and the work ID is created. The template characters are ####, @@@@@@, and @@@@@@@@.

- The #### characters are replaced by an ascending sequence of numbers. For example, if you specify MJF.DDL#### for a new work ID, MJF.DDL0001 is created. If MJF.DDL0001 already exists, MJF.DDL0002 is created.

- The @@@@@@ and @@@@@@@@ characters are replaced by the current date. For example, if you specify MJF.DDL@@@@@@@@ for a new work ID, MJF.DDL20000428 is created. If MJF.DDL20000428 already exists, an error is issued when the CM/PILOT worklist is processed. The format of @@@@@@ is the date MMDDYY. The format of @@@@@@@@ is the date MMDDYYYY.

The use of name templates has the following restrictions:

- Work ID names must begin with a letter. You can specify name template characters anytime after the first letter of a work ID name.

- One group of the same template characters in a work ID must be used. For example, do not create a new work ID of MJF.DDL####TEST####. You can specify two different groups of template characters in a work ID. For example, you can specify a new work ID of MJF.DDL####TEST@@@@@@@@ that can resolve to a work ID such as MJF.DDL0001TEST04282000.

### Table 45 Work ID types for scripts

<table>
<thead>
<tr>
<th>Script name</th>
<th>Work ID type</th>
<th>New or existing work ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP_STRUCTURE_CHG</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>DML_STRUCTURE_CHG</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>DML_MIGRATE</td>
<td>migrate-type</td>
<td>New</td>
</tr>
<tr>
<td>WKID_STRUCTURE_CHG</td>
<td>alter-type</td>
<td>Existing</td>
</tr>
<tr>
<td>WKID_MIGRATE</td>
<td>migrate-type</td>
<td>Existing</td>
</tr>
<tr>
<td>REC_CHG_SS</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>REC_CHG_SR_BL</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>REC_DDL</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>CREATE_FULL_RCV_BL</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>RCV_STRUCT_CUR_DATA</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>RCV_STRUCT_OLD_DATA</td>
<td>alter-type</td>
<td>New</td>
</tr>
<tr>
<td>REPLICATE_WORKID</td>
<td>alter-type</td>
<td>New</td>
</tr>
</tbody>
</table>
NOTE
You can use a sequence of eight at symbols (@) to include the date as part of the name template.

For task IDs that require existing work IDs, you must have created those work IDs with CHANGE MANAGER and used the CHANGE MANAGER Specification component to select the data structures to change or migrate. You can use a wildcard pattern to create a list of existing Work IDs from which to select.

6 (optional) Type the name of a CM/PILOT Application. An application groups a set of CHANGE MANAGER profiles. For more information about applications, see “Using applications” on page 666.

NOTE
For a new work ID, you can specify a work ID name template in an application instead of either an explicit name or work ID name template in the WORKID field. To do so, define the work ID name template in the application, and then specify the application name in the Application field on this panel.

If you want to list the applications, type a wildcard pattern for Application, and press Enter. The Application Selection List panel is displayed. Type S in the Act column adjacent to the application that you want to use and press Enter.

7 (optional) Specify a Comment to describe the task ID.

8 If you are creating a task ID that uses Data Manipulation Language (DML), type S to select Create or Edit DML. For more information, see “Migrating data structures by using DML” on page 123 or “Changing data structures by using DML” on page 415.

9 Press END.

The task ID is created and the TASKID Action Menu is displayed.

Listing task IDs

You can perform various actions from a list of task IDs. The task ID list displays information about the status of the task ID, the name of the script used, and comments that describe the task ID.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

The CM/PILOT Main Menu is displayed.
Editing a task ID

You can change information about a task ID.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.
   The TASKID Action Menu is displayed.

3 Specify a wildcard pattern for TASKID (for example, AUC.*) and select List TASKIDs. Then, press Enter.
   The TASKID Selection List is displayed.

4 Type an action code in the Act column adjacent to the task ID.

5 Press END.
   The TASKID Action Menu is displayed.

NOTE
If you are creating a task ID that uses Data Manipulation Language (DML), type S to select Create or Edit DML. For more information, see “Migrating data structures by using DML” on page 123 or “Changing data structures by using DML” on page 415.
Executing a task ID

You can execute a task ID. Executing a task ID consists of creating a CM/PILOT worklist and the JCL for processing the worklist, and then processing the CM/PILOT worklist. The results of a successful execution are an Analysis worklist and the Execution JCL to process the worklist.

If the worklist fails to process successfully, you must correct the error and specify either to restart the processing from the last sync point before the failure or to start the worklist processing over from the beginning. For more information, see “Restarting a CM/PILOT worklist” on page 642 and “Starting a CM/PILOT worklist over” on page 644.

In addition to the information that you specified for the task ID before selecting to execute it, you need to provide CM/PILOT with data set names and other processing options.

To execute a task ID

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.
   The TASKID Action Menu is displayed.

3 Type the name of a TASKID or a specify a wildcard pattern and select Execute a TASKID. Then, press Enter.
   The TASKID Interface panel is displayed.

4 Type the name of a WORKID, if it is not already specified.

5 Specify the data set names to be used to process the CM/PILOT worklist.

   A Type the name of the data set that will contain the CM/PILOT Worklist. The data set can be partitioned or sequential, and you can use symbolic variables in the name.

   ■ If you specify a partitioned data set (PDS), you can either provide a member name or leave it blank to create a list of existing members from which to select. The data set must be fixed block (FB) and have a logical record length (LRECL) of 80.
If you specify a sequential data set that does not exist, it is created when the CM/PILOT worklist is created. If you specify a sequential data set that does exist, the disposition of the data set is set to shared and any existing data in the data set is overwritten.

B Type the name of the data set that will contain the JCL to process the CM/PILOT worklist. The data set can be partitioned or sequential, and you can use symbolic variables in the name.

- If you specify a PDS, you can either provide a member name or leave it blank to create a list of existing members from which to select.

- If you specify a sequential data set that does not exist, it is created when the JCL is created. If you specify a sequential data set that does exist, the disposition of the data set is set to shared and any existing data in the data set is overwritten.

C Type the name of the data set for Diagnostics. You can specify either a sequential data set or SYSOUT for the diagnostic output from CM/PILOT worklist processing. You can also use symbolic variables to specify a sequential data set.

If you specify a sequential data set that does not exist, it is created when the CM/PILOT worklist is processed. If you specify a sequential data set that does exist, its disposition is set to SHR and any existing data in the data set is overwritten.

For more information about the diagnostic output, see “To review the diagnostic output” on page 637.

6 Select the method of executing the worklist. For more information about worklists, see “Viewing the execution status of a task ID” on page 639.

- **Initial** indicates that this is the initial processing job for the CM/PILOT worklist.

- **Restart** indicates that this is a restart of the processing job for the CM/PILOT worklist. The initial job had previously failed and is being restarted from a sync point.

- **Start Over** indicates that this CM/PILOT worklist processing job is being started over from the beginning. The job either previously failed and you have determined that it cannot be restarted from a sync point, or you want to reuse the Task ID.

7 To override the options that are set in CHANGE MANAGER to process the CM/PILOT worklist, type **S** to select **Override CHANGE MANAGER options** and then press Enter.

The Override CHANGE MANAGER Options panel is displayed.
8. On the Override CHANGE MANAGER Options panel, select **Override CHANGE MANAGER Options**.

9. Type **S** to select the options that you want to override, and press **Enter**.

   For more information about overriding the CHANGE MANAGER options, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*.

---

**WARNING**

BMC does not recommend that you select the Autosubmit override option. This powerful option automatically submits the JCL, which is created from the CM/PILOT generated worklist, to Execution. When you use the Autosubmit override option, use caution because any error in your specifications might cause significant damage to your schemas and a loss of data.

---

10. Press END until the TASKID Interface panel is displayed.

11. On the TASKID Interface panel, type **S** to select **Create Worklist**. Then, press **Enter**.

   The Step Settings panel is displayed.

12. Type **S** to select the steps in the script that you want to modify, and press **Enter**.

   The subsequent Step Settings panels that are displayed vary depending on the script used for the task ID.

---

**NOTE**

A maximum of two settings are shown for each step on the Step Settings panel. Additional settings that you can modify on subsequent panels for each step might exist.

---

The panels that might be displayed enable you to specify:

- the scope for a catalog to catalog comparison
- outbound and inbound migrate profiles
- baseline profiles
- baseline names for new baselines that will be created
- baseline names for existing baselines that will be used
- work ID to use in a work ID comparison
- data set sizing

If a script requires profiles, and the profiles are defined in the application that the task ID is using, the profile names appear on the appropriate panels. You can change the profile names if necessary.
After the Step Settings panels are displayed, the CHANGE MANAGER Datasets panel is displayed.

13 Specify the data set names to contain items such as the Analysis worklist, the Execution JCL for processing the Analysis worklist, and the CDL file.

---
**NOTE**
The data sets that are shown vary, depending on the script that is used for the task ID.

14 Press END.

The TASKID Interface panel is displayed.

15 Select your options for processing the worklist.

A To edit the worklist, type S to select **Edit Worklist**.

---
**NOTE**
In columns 73 through 80 of the last line of each command in the CM/PILOT worklist, a hash verification number exists. CM/PILOT uses the number for diagnostics to detect changed commands and inserted commands. Do not modify this number.

B To create the JCL for processing the CM/PILOT worklist, type S to select **Create JCL**.

C To review or modify the JCL, type S to select **Edit JCL**.

D To submit the JCL to process the CM/PILOT worklist, type S to select **Submit JCL**.

E Press Enter.

16 Execute the Analysis worklist. For more information about executing Analysis worklists, see “Restarting a worklist” on page 164 or “Executing an alter-type worklist” on page 562.

---
**NOTE**
The JCL that you use to execute the Analysis worklist is contained in the data set that you specified for **JCL Dataset Name** in the CHANGE MANAGER Datasets panel.
To create a multi-step job

You can create additional job steps in the Execution JCL that is created to process the Analysis worklist. Such steps include creating a baseline and performing a comparison before or after Analysis worklist processing. For example, updates, establishing recovery points, and generating change records can be combined into one job.

You can specify additional job steps by selecting to modify the step settings for the Execution step on the Step Settings panel. Then, you select Specify additional steps to include in job. You specify the additional steps on the Execution Pre- and Post-Processing Interface panel, which is shown in Figure 177. You are prompted for additional information on subsequent panels, depending on the job steps that you choose. For more information, see “Generating a multi-step job” on page 564.

Figure 177  Creating additional job steps in the Execution interface

TIP

Consider the following items when you execute a task ID:

- You can view the execution status of a task ID by issuing the VIEW command-line command from the TASKID Interface panel.
- If you directed the diagnostic output to a sequential data set, you can browse it from the TASKID Interface panel by issuing the BROWSE command-line command.
To reuse a task ID

If you use a script to perform a change management task repeatedly, you can reuse the same task ID each time. When you execute the task ID, you can use one of the following options:

- Create a new work ID by changing the work ID name on the TASKID Interface panel.
- Reuse the existing work ID by leaving the work ID name as it is displayed.

**NOTE**

When you reuse a work ID, the change definition (CD) table entries for the existing work ID are deleted before the new work ID is created.

You can only reuse a work ID with those scripts that create a new work ID. You cannot reuse a work ID with the following scripts:

- Change Data Structures Using an Existing Work ID (WKID_STRUCTURE_CHG)
- Migrate Data Structures Using an Existing Work ID (WKID_MIGRATE)

To review the diagnostic output

The CM/PILOT component provides diagnostic output in the AEXPRINT file. Diagnostic output contains information about the processing of a CM/PILOT worklist. If the Task ID contains DML, the diagnostic output includes a DML report. You can direct the diagnostic output to SYSOUT or to a sequential data set by specifying your choice on the TASKID Interface panel.

Diagnostic output from the CHANGE MANAGER Execution job that processes the CHANGE MANAGER worklist is also placed in the AEXPRINT file. For more information about the CHANGE MANAGER diagnostics that the Execution component provides, see “Reviewing the results of a migration” on page 103 or “Reviewing the results of your changes” on page 353.

DML report

The DML report that is included as part of the CM/PILOT diagnostic output contains information about the data structures that are affected by the DML statements. You can view this report and evaluate the changes before you execute the changes. You can also determine which data structures are affected by accessing the CHANGE MANAGER Worklist Specification, which marks the affected objects as “Changed.”
If you want to produce a DML report without creating an Analysis worklist, you can create a trial DML report. To create a trial DML report, you must select to change the Analysis step options when you create the CM/PILOT worklist. Then select the option to produce a trial DML report. The CM/PILOT worklist will contain only the necessary commands to produce the report. If the results of the report are not what you intended, edit the WHERE clause in the DML statement to narrow the scope. If you want to execute the task ID after creating a trial DML report, you must re-create the CM/PILOT worklist.

**NOTE**

Although the Analysis worklist is not created when you create a trial report, CHANGE MANAGER updates the CD tables for the work ID.

The DML report contains the following sections: summary, DML, and detail.

- **Summary section**

  The summary section of the report provides the data structures that are affected by each type of DML statement in the Task ID. For example, this section shows how many table spaces and other data structures would be affected by an UPDATE TABLESPACES statement.

- **DML section**

  The DML section of the report provides the actual DML statements, with statement numbers, for a Task ID. The statement numbers are referenced by any problems that are listed in the WARNING, ERRORS and CONFLICTS DETECTED subsections. The following list describes warning conditions, error conditions, and conflicts:

  - A warning condition occurs when the results of DML statement actions might be different than the anticipated results. A CHANGE MANAGER worklist is created.
  
  - An error condition denotes a problem that you must correct before a CHANGE MANAGER worklist is created.
  
  - A conflict occurs when two or more statements assign conflicting values to an object.

- **Detail section**

  The detail section of the DML report provides a detailed listing of the UPDATE, DELETE, and MIGRATE statements for a Task ID. This information provides the old and new values of the attributes that are being updated.
Viewing the execution status of a task ID

You can view the execution status for a task ID by viewing the sync table entries. Sync table entries are created by CM/PILOT at various intervals during worklist processing. They are used to mark restart points in case of a failure during processing. By viewing the sync table entries you can see the processing status of a CM/PILOT worklist and determine a point of failure if you must correct an error.

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.

3. Type the name of a TASKID or a specify a wildcard pattern and select View Execution Status of a TASKID. Then, press Enter.

   The TASKID Execution Status panel is displayed, as shown in Figure 178.

Figure 178    TASKID Execution Status panel

If no changes are made to the LABEL attribute of tables, views, columns, and aliases, the LABEL attribute is not shown in the report. This omission enhances the readability of the report.
Browsing a task ID

Table 46 describes the fields on the TASKID Execution Status panel.

Table 46  Status of task ID execution

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executor</td>
<td>user ID of the person who submitted the CM/PILOT worklist for processing</td>
</tr>
<tr>
<td>Sequence</td>
<td>sequence numbers of the sync points in the CM/PILOT worklist [Sync points are identified by the -SYNC command in the worklist.]</td>
</tr>
<tr>
<td>Status</td>
<td>status of the CM/PILOT worklist processing job [The following statuses are valid for the job: [Exec Strt (Execution started) [Exec Comp (Execution complete)]</td>
</tr>
<tr>
<td>Step</td>
<td>the CM/PILOT worklist command that is in effect</td>
</tr>
<tr>
<td>Sync Point</td>
<td>the text of the -SYNC command in the CM/PILOT worklist</td>
</tr>
<tr>
<td>TASKID</td>
<td>the name of the task ID that is associated with the CM/PILOT worklist</td>
</tr>
</tbody>
</table>

**TIP**

Consider the following items when you view the execution status of a task ID:

- You can also view the execution status of a task ID by issuing the VIEW command-line command from the TASKID Interface panel.

- You can view the execution status of task IDs for a specific application by listing the task IDs from the Application Selection List. For more information, see “Listing applications” on page 669 and “Listing task IDs for an application” on page 672.

Browsing a task ID

Browsing a task ID enables you to view read-only information about the task ID.

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select TASKIDs and press Enter.

   The TASKID Action Menu is displayed.
Deleting a task ID

3 Type the name of a TASKID or a specify a wildcard pattern and select Browse a TASKID. Then, press Enter.

The Browse TASKID panel is displayed.

4 If you are browsing a task ID that contains DML, select Browse DML.

The List DML Statements panel is displayed.

5 Type S adjacent to the statements that you want to browse. After browsing a statement, press END. When you are finished browsing DML statements from the List DML Statements panel, press END.

TIP
You can browse the task IDs that use a specific application by listing the task IDs from the Application Selection List. For more information, see “Listing applications” on page 669 and “Listing task IDs for an application” on page 672.

Deleting a task ID

To help control the size of the CP_TASKID and CP_SYNC tables, as well as other product CD tables, you can delete any unwanted task IDs. For information about controlling the size of sync tables, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

The CM/PILOT Main Menu is displayed.

2 Select TASKIDs and press Enter.

The TASKID Action Menu is displayed.

3 Type the name of a TASKID or a specify a wildcard pattern and select Delete a TASKID. Then, press Enter.

The TASKID Delete Confirmation panel is displayed.
4 Select Delete TASKID and press Enter.

Tip
You can delete task IDs for a specific application by listing the task IDs from the Application Selection List. You can also delete task IDs by listing them from other panels. For more information, see “Listing task IDs” on page 630 and “Listing task IDs for an application” on page 672.

Processing CM/PILOT worklists

A CM/PILOT worklist contains commands for CHANGE MANAGER components and passes to the components the information that you provide through the panel. The JCL for processing the CM/PILOT worklist contains information such as the data sets that are needed to store the CHANGE MANAGER worklist and its associated CHANGE MANAGER Execution JCL. When a CM/PILOT worklist is processed successfully, an Analysis worklist is created, along with the associated Execution JCL to process the Analysis worklist. You can submit the Execution JCL to process the worklist whenever you like. The actual changes to data structures occur when the worklist is processed.

Restarting a CM/PILOT worklist

If a CM/PILOT worklist does not process successfully, you can usually restart it from a sync point. If you cannot restart the worklist from a sync point, the component displays a message. For more information if you cannot restart a failed job, see “Starting a CM/PILOT worklist over” on page 644.

Examine the diagnostic output (AEXPRINT) to determine whether the failure was caused by an error that you must correct in the CM/PILOT worklist. You can also view the sync table for the task ID to determine the point of failure. For more information about viewing the sync table, see “Viewing the execution status of a task ID” on page 639.
Whether you can restart a CM/PILOT worklist depends on the status of the task ID and the status of the associated work ID. The task ID must have a status of Exec Strt (execution started). The work ID can have the following statuses:

- If the work ID has the same status when you attempt to restart the job as it had when the job failed, no additional action is required concerning the work ID.

- If the work ID has a more advanced status when you attempt to restart the job than it had when the job failed, the product asks you whether you want to override the work ID’s current status and continue to use the same work ID. Work ID status values are (shown in order of progression):
  1. Created
  2. Analyzed
  3. Exec Strt (CHANGE MANAGER execution started)
  4. Exec Comp (CHANGE MANAGER execution completed)

- If the work ID has a lesser status when you attempt to restart the job than it had when the job failed, you cannot restart the job. You must define a different work ID to the task ID and re-create the CM/PILOT worklist. You can specify a different work ID on the TASKID Interface panel.

Table 47 shows the situations in which you can restart CM/PILOT worklist processing.

<table>
<thead>
<tr>
<th>Task ID Status</th>
<th>Work ID Status</th>
<th>Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec Strt (execution started)</td>
<td>Created</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Analyzed</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Exec Strt</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Exec Comp</td>
<td>Yes</td>
</tr>
<tr>
<td>Exec Comp (execution completed)</td>
<td>Created</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Analyzed</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Exec Strt</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Exec Comp</td>
<td>No</td>
</tr>
</tbody>
</table>

Restart processing is activated by the RESTART keyword in the AEXIN input stream of the JCL for processing the CM/PILOT worklist, as shown in Figure 179. Selecting Restart for Worklist Execution on the TASKID Interface panel inserts this keyword into the input stream.
Starting a CM/PILOT worklist over

If a CM/PILOT worklist failed to process successfully and you cannot restart from a sync point, or it failed and you do not want to use restart processing, you can use startover processing. Processing of the worklist starts over from the beginning of the worklist.

Examine the diagnostic output to determine whether the failure was caused by an error that you must correct in the CM/PILOT worklist. You can also view the sync table for the task ID to determine the point of failure. For more information about viewing the sync table, see “Viewing the execution status of a task ID” on page 639.

You can start CM/PILOT worklist processing over regardless of the status of the task ID. If the work ID status is Exec Strt (CHANGE MANAGER execution started), then you cannot use the same work ID to start over the processing of the CM/PILOT worklist. You must define a different work ID to the task ID and re-create the CM/PILOT worklist. You can specify a different work ID on the TASKID Interface panel.

NOTE
You should always determine the extent of processing done by CHANGE MANAGER when a work ID has a status of Exec Strt. Actual changes to DB2 data structures could have already been performed.
Table 48 shows the situations in which you can start the CM/PILOT worklist processing over.

Table 48  Startover processing of a CM/PILOT worklist

<table>
<thead>
<tr>
<th>Task ID status</th>
<th>Work ID status</th>
<th>Start over</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec Strt (execution started)</td>
<td>Created</td>
<td>Yes</td>
</tr>
<tr>
<td>Exec Comp (execution completed)</td>
<td>Analyzed</td>
<td>Yes</td>
</tr>
<tr>
<td>Exec Strt</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Exec Comp</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Startover processing is activated by the STARTOVER keyword in the AEXIN input stream of the JCL for processing the CM/PILOT worklist, as shown in Figure 180. Selecting Start Over for Worklist Execution on the TASKID Interface panel inserts this keyword into the AEXIN input stream.

Figure 180  AEXIN input stream for starting over a worklist

```
//AEXIN DD *
SSID DEBA WORKID RDAMAF2.DMLMIG
DASDLOPT DS631EBA STARTOVER
```

For work IDs that use name templates with sequence number characters (####), CM/PILOT automatically increments the number when you use startover processing. For example, if you start worklist processing over for work ID WK#### and work ID WK0001 was created in the initial job, CM/PILOT will create work ID WK0002 for the startover job. You must delete work ID WK0001 if you no longer need it. For information about the -WKID CM/PILOT worklist command, which creates a new work ID, see the ALTER and CHANGE MANAGER for DB2 Reference Manual.
Using scripts

A script is a predefined set of ordered steps that is needed to perform a CHANGE MANAGER process (see Figure 181). Each task ID that you create uses a script. The script that you choose determines which panels are displayed and their sequence in creating and executing a Task ID.

Figure 181  Script steps

Some scripts require existing CHANGE MANAGER objects. For example, a script might require that an alter-type work ID with data structure changes specified in it exists. When you create a task ID that uses this script, you specify the alter-type work ID.

You select a script from the Script Selection List (see Figure 182), which is displayed when you create a new Task ID.
If you use a script to perform a change management task repeatedly, you can reuse the same task ID each time. For more information, see “To reuse a task ID” on page 637.

**Table 49** describes the actions that you can perform with scripts. The procedural steps that are needed to perform actions, along with the related information, are described.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating a script”</td>
<td>page 648</td>
</tr>
<tr>
<td>“Listing scripts”</td>
<td>page 659</td>
</tr>
<tr>
<td>“Editing a script”</td>
<td>page 660</td>
</tr>
<tr>
<td>“Listing the steps for a script”</td>
<td>page 661</td>
</tr>
<tr>
<td>“Executing a user-defined script”</td>
<td>page 662</td>
</tr>
<tr>
<td>“Browsing a script”</td>
<td>page 662</td>
</tr>
<tr>
<td>“Deleting a script”</td>
<td>page 663</td>
</tr>
<tr>
<td>“Creating a task ID from a script”</td>
<td>page 664</td>
</tr>
<tr>
<td>“Listing task IDs for a script”</td>
<td>page 665</td>
</tr>
<tr>
<td>“Copying a script supplied by BMC”</td>
<td>page 665</td>
</tr>
<tr>
<td>“Copying a user-defined script”</td>
<td>page 666</td>
</tr>
</tbody>
</table>
Creating a script

You can create CM/PILOT scripts to perform many of the CHANGE MANAGER processes. You can create a new script, or you can copy a BMC-supplied script or a user-defined script, modify it, and save it with a new name.

To create a script

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select Scripts and press Enter.

   The Script Action Menu is displayed.

3. Type a name for the Script and select Create a Script. Then, press Enter.

   The Create Script panel is displayed.

4. Specify the information to create the script.

   A. Specify the WORKID Type.

      - Alter requests changes to object definitions that reside within the same DB2 subsystem.

      - Migrate specifies data structure definitions to be migrated from one DB2 subsystem to another, or duplicates structures within the same subsystem.

      - Receive receives migration when data structures are being created on a different subsystem.

   B. Specify whether DML statements are required in the script. DML is an SQL-like data manipulation language, with statements to update, delete, and migrate data structures.

   C. (optional) Specify a Comment to describe the script.

   D. Type S to select Create or Edit Script Steps.

   E. Press Enter.

      The Step Selection List panel is displayed.
5 Type E in the Act column adjacent to the step that you want to edit.

The Edit Step panel is displayed.

6 Specify the information for the step in the script.

A Specify the CHANGE MANAGER process that you want to include in the script.
  - To create a baseline step, see “To create a baseline step” on page 650.
  - To create a comparison step, see “To create a comparison step” on page 651.
  - To create an import step, see “To create an import step” on page 653.
  - To create an analysis step, see “To create an analysis step” on page 655.
  - To create an execution step, see “To create an execution step” on page 656.
  - To create a baseline report step, see “To create a baseline report step” on page 657.
  - To create a replicate work ID step, see “To create a replicate work ID step” on page 658.

B Type S to select Specify options for this step.

C Press Enter.

7 When you have specified the information for a step for the script, press END.

The Step Selection List is displayed.

8 Type an action code.
  - To add an additional step, type I in the Act column adjacent to a step.
  - To edit the options in a step, type O in the Act column adjacent to the step.

9 Press END.

The Create Script panel is displayed.

10 Press END.
To create a baseline step

1 To create a step in the script that performs the baseline process, select the Baseline Process in the Edit Step panel (see step 6 on page 649).

The Baseline Step Options panel is displayed, as shown in Figure 183.

Figure 183 Baseline Step Options panel

![Baseline Step Options panel](image)

2 Specify the information for a baseline. For more information about creating baselines, see Chapter 2, “Taking a snapshot of a database environment.”

If you want to prevent others from overriding the options that you specify in this panel, type Y to select Lock Options.

NOTE
When you select Lock Options, the baseline step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press Enter to validate your options, and then press END.

The Step Baseline Profile Options panel is displayed, as shown in Figure 184.
Figure 184  Step Baseline Profile Options panel

```
AUCFSTA1 ------------------ Step Baseline Profile Options ---------------------
Command ===>  

Script ..... : RDACRJ.SCR01  
Step Number . : 1  
Process . . . : BASELINE  

Select type of Step Baseline Profile option.  

1. Prompt for a Baseline Profile for this step when generating a CM/Pilot Worklist.  
2. Specify a default Baseline Profile for this step.  
3. Specify a Baseline Profile that will always be used in this step. Cannot override when generating a CM/Pilot Worklist.  

Type a Baseline Profile, or type a wildcard pattern for a selection list, if Step Baseline Profile Option is 2 or 3. Then press Enter.  

Baseline Profile  

Commands: HELP END CANCEL```

4  Specify the options for the baseline profile, and then press END.

To create a comparison step

1  To create a step in the script that performs the comparison process, select the Compare Process in the Edit Step panel (see step 6 on page 649).

The Compare Step Options panel is displayed, as shown in Figure 185.
Figure 185 Compare Step Options panel

AUCFSTCO ---------------------- Compare Step Options ----------------------

Command ===>

Script . . . . . . . : RDA.CJ.SCR01
Step Number . . . . . : 1

Select Compare1 type, where the resulting CDL will be applied.
Compare1 Type .. 2 1. Baseline
2. Catalog
3. DDL
4. Worklist
5. WORKID

Select Compare2 type, where the objects already have the desired structure definitions.
Compare2 Type .. 2 1. Baseline
2. Catalog
3. DDL
4. Worklist

Select Compare step options.
Lock Step Options . . _ Enter Y to lock step options

Commands: HELP END CANCEL

2 Specify the information for performing a comparison. For more information about comparing data structures, see Chapter 3, “Comparing database environments.”

If you want to prevent others from overriding the options that you specify in this panel, type Y to select Lock Step Options.

NOTE

When you select Lock Step Options, the compare step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press Enter to validate your options, and then press END.

The Step Migrate Profile Options panel is displayed, as shown in Figure 186.
4 Specify the options for an outbound migrate profile, and then press END.

To create an import step

1 To create a step in the script that performs the import process, select the Import Process in the Edit Step panel (see step 6 on page 649).

**NOTE**

If your script requires DML, you cannot create a step in the script that performs the import process.

The Import Step Options panel is displayed, as shown in Figure 187.
2 Specify the information for importing a CDL or DDL file. For more information about importing external files, see Chapter 4, “Altering data structures in a database environment.”

If you want to prevent others from overriding the options that you specify in this panel, type Y to select Lock Step Options.

**NOTE**
When you select Lock Step Options, the import step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press Enter to validate your options, and then press END.

The Step Migrate Profile Options panel is displayed, as shown in Figure 188.
4 Specify the options for an inbound migrate profile, and then press END.

To create an analysis step

1 To create a step in the script that performs the analysis process, select the Analysis Process in the Edit Step panel (see step 6 on page 649).

The Analysis Step Options panel is displayed, as shown in Figure 189.
Figure 189  Analysis Step Options Panel

2 Specify the information for analyzing a work ID and generating a worklist. For more information about the analysis process, see Chapter 1, “Migrating database environments.”

If you want to prevent others from overriding the options that you specify in this panel, type Y to select Lock Options.

NOTE
When you select Lock Options, the analysis step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press Enter to validate your options, and then press END.

To create an execution step

1 To create a step in the script that performs the execution process, select the Execution Process in the Edit Step panel (see step 6 on page 649).

The Execution Step Options panel is displayed, as shown in Figure 190.
Figure 190  Execution Step Options panel

AUCFSTEO --------------------- Execution Step Options ----------------------

Command ===>  
Script ........ : RDACRJ.SCROI
Step Number ... : 1

Select execution step options. Then press Enter.

Dataset Sizing  1 1. No 2.Runstats 3.BMC Stats 4.Sample VSAM Object

Pre-Execution Compare  1 1. Do not do a compare before execution
  2. Compare two previous baselines
  3. Compare a previous baseline to the current catalog

Pre-Execution Baseline  1 1. Do not build a baseline before execution
  2. Build a baseline before execution

Post-Execution Compare  1 1. Do not do a compare after execution
  2. Compare previous baselines to the new catalog
  3. Compare the baseline built during a previous job step to the new catalog

Post-Execution Baseline  1 1. Do not build a baseline after execution
  2. Build a baseline after execution

Lock Step Options ... N (Y/N)

Commands: HELP END CANCEL

2 Specify the information for a executing a worklist. For more information about the execution process, see Chapter 1, “Migrating database environments.”

If you want to prevent others from overriding the options that you specify in this panel, type Y to select Lock Step Options.

**NOTE**

When you select Lock Step Options, the execution step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press Enter to validate your options, and then press END.

**To create a baseline report step**

1 To create a step in the script that creates a baseline report, select the Baseline Report Process in the Edit Step panel (see step 6 on page 649).

The Baseline Report Step Options panel is displayed, as shown in Figure 191.
2 If you want to prevent others from overriding the options that you specify in this panel, type Y to select **Lock Step Options**.

**NOTE**

When you select **Lock Step Options**, the baseline report step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press **Enter** to validate your options, and then press **END**.

**To create a replicate work ID step**

1 To create a step in the script that replicates a work ID, select the Replicate WORKID Process in the Edit Step panel (see step 6 on page 649).

   The Replicate WORKID Step Options panel is displayed, as shown in Figure 192.
2 Specify the information for replicating a work ID. For more information about replicating work IDs, see Chapter 4, “Altering data structures in a database environment.”

If you want to prevent others from overriding the options that you specify in this panel, type Y to select **Lock Step Options**.

---

**NOTE**

When you select **Lock Step Options**, the replicate work ID step does not appear in the Step Settings panel when you execute the task ID for the script. The only steps that appear in this panel are those that can be modified.

3 Press **Enter** to validate your options, and then press **END**.

### Listing scripts

You can perform various actions from a list of scripts. The User-Defined Script Selection List panel displays information about the script owner, name, availability, date created, and comments.

1 On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.

The CM/PILOT Main Menu is displayed.
2 Select Scripts and press Enter.

The Script Action Menu is displayed.

3 Specify a wildcard pattern for Script and select List Scripts. Then, press Enter.

The User-Defined Script Selection List is displayed.

4 Type an action code adjacent to the script.

5 Press END.

The Script Action Menu is displayed.

---

**TIP**

Consider the following items when you list scripts:

- You can also list scripts when you create or edit a Task ID. Specify a wildcard pattern in the Script field of the Create TASKID or Edit TASKID panels.

- You can also list the task IDs that use a script by selecting List TASKIDs for a Script on the Script Action Menu panel.

---

**Editing a script**

You can edit information about a script.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

The CM/PILOT Main Menu is displayed.

2 Select Scripts and press Enter.

The Script Action Menu is displayed.

3 Type the name of a Script or specify a wildcard pattern for Script and select Edit a Script. Then, press Enter.

The Edit Script panel is displayed.
4 You can change the type of work ID, indicate whether DML is required, and alter the comments about the script. Then, press END.

---

**NOTE**
To create or edit steps for the script, type Y to select Create or Edit Script Steps. For more information, see “Creating a script” on page 648.

---

**TIP**
You can also edit a script by selecting List Scripts on the Script Action Menu panel. For more information, see “Listing scripts” on page 659.

---

**Listing the steps for a script**

You can list all of the steps for a script.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2 Select Scripts and press Enter.

   The Script Action Menu is displayed.

3 Type the name of a Script or specify a wildcard pattern for Script and select List Steps for a Script. Then, press Enter.

   The Step Selection List panel is displayed.

4 Press END.
Executing a user-defined script

You can execute a script that you create.

1 Create a task ID by performing the steps in “Creating a task ID” on page 628.

   **NOTE**
   In step 4 on page 628, select **Select a user defined script** and type a wildcard pattern for **Script**, or type the name of a **Script**.

2 Execute the task ID by performing the steps in “Executing a task ID” on page 632.

   **TIP**
   You can also execute a script in the User-Defined Script Selection List panel by typing X in the **Act** column adjacent to the script.

Browsing a script

You can browse information about a script. Browsing a script enables you to view read-only information about the scripts and the steps in a script.

1 On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.

   The CM/PILOT Main Menu is displayed.

2 Select **Scripts** and press **Enter**.

   The Script Action Menu is displayed.

3 Type the name of a **Script** or specify a wildcard pattern for **Script** and select **Browse a Script**. Then, press **Enter**.

   The Browse Script panel is displayed.

4 Type S to **Select to Browse Script Steps** and then press **Enter**.

   The Step Selection List is displayed.

5 Press END.

   The Browse Script panel is displayed.
Deleting a script

You can delete scripts when you no longer need them.

---

**NOTE**

You cannot delete a script that is used by a task ID. You must first either change or delete the associated task IDs.

---

1. On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.
   
   The CM/PILOT Main Menu is displayed.

2. Select **Scripts** and press **Enter**.
   
   The Script Action Menu is displayed.

3. Type the name of a **Script** or specify a wildcard pattern for **Script** and select **Delete a Script**. Then, press **Enter**.
   
   The Delete Script Confirmation Panel is displayed.

4. Select **Delete Script** and press **Enter**.

---

**TIP**

You can also browse a script by selecting **List Scripts** on the Script Action Menu panel. For more information, see “Listing scripts” on page 659.
Creating a task ID from a script

You can create a task ID from a script. Creating a task ID is the first step in using CM/PILOT to perform CHANGE MANAGER processes. All work in CM/PILOT is accomplished by processing a task ID.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2 Select Scripts and press Enter.

   The Script Action Menu is displayed.

3 Type the name of a Script or specify a wildcard pattern for Script and select Create a TASKID from a Script. Then, press Enter.

   The Create TASKID panel is displayed.

4 Specify the information for the task ID.

   A Type the name of the TASKID.

   A CM/PILOT uses the name of the task ID as the name for a new work ID. You can type a different name for WORKID or type the name of a work ID name template.

   B If you are creating a task ID that uses DML, type S to select Create or Edit DML. For more information, see “Migrating data structures by using DML” on page 123 or “Changing data structures by using DML” on page 415.

   C Press Enter.

5 Press END. The Script Action Menu is displayed.

   TIP

   You can also create a task ID from a script by selecting List TASKIDs for a Script on the Script Action Menu panel. For more information, see “Listing task IDs for a script” on page 665.
Listing task IDs for a script

You can list the task IDs that are associated with a script.

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.

2. Select Scripts and press Enter.
   The Script Action Menu is displayed.

3. Type the name of a Script or specify a wildcard pattern for Script and select List TASKIDs for a Script. Then, press Enter.
   The TASKID Selection List panel is displayed.

4. Type an action code adjacent to the task ID.

Copying a script supplied by BMC

You can copy a script that BMC supplies. You can view the steps in the script, then copy it, rename it, and modify it.

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.

2. Select Scripts and press Enter.
   The Script Action Menu is displayed.

3. Type the name of a Script and select Create a new Script based on a Script from BMC Software. Then, press Enter.
   The Script Selection List for Scripts Supplied by BMC Software panel is displayed.

4. Type an action in the Act column adjacent to a script.

   A. To view the steps in a script, type S and press Enter. The Step Selection List panel is displayed. Press END.

   B. To copy the script, type C and press Enter. The Create Script panel is displayed.
Copying a user-defined script

You can copy a script that you have created. You can view the steps in the script, and then copy it, rename it, and modify it.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

The CM/PILOT Main Menu is displayed.

2 Select Scripts and press Enter.

The Script Action Menu is displayed.

3 Specify the name of the Script that you want to copy or specify a wildcard pattern. Then, select Create a new Script based on a User Defined Script and press Enter.

The User-Defined Script Selection List panel is displayed.

4 Type S in the Act column adjacent to the script that you want to copy. Then, press Enter.

The Create Script panel is displayed.

5 Type the name of the new Script and then perform the steps in “Creating a script” on page 648 to specify the information to create a script.

Using applications

The CM/PILOT applications group a set of CHANGE MANAGER profiles. This feature enables you to define a standard set of profiles for the various change management tasks of a DB2 application. The number of task IDs that an application can use is unlimited.

You can also define a work ID name template in an application. A work ID name template lets you standardize naming conventions for new work IDs that CM/PILOT creates.
You create an application and perform other actions from the Application Action menu, which is displayed in Figure 193.

**Figure 193  Application Action Menu**

<table>
<thead>
<tr>
<th>Command</th>
<th>Application Action Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type a specific Application name or type a wildcard pattern for a selection list. Example: A.B to access a specific Application, A*.B* for all Applications like this pattern.</td>
<td></td>
</tr>
<tr>
<td>Application  RDACRJ.*</td>
<td></td>
</tr>
</tbody>
</table>

Select an action. Then Press Enter.
1. List Applications
2. Create an Application
3. Edit an Application
4. Browse an Application
5. Delete an Application
6. Create a TASKID from an existing Application
7. List TASKIDs for an Application

Commands: HELP END

**Table 50** describes the actions that you can perform with applications. The procedural steps that are needed to perform actions, along with the related information, are described.

**Table 50  Application tasks**

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating an application”</td>
<td>page 668</td>
</tr>
<tr>
<td>“Listing applications”</td>
<td>page 669</td>
</tr>
<tr>
<td>“Editing an application”</td>
<td>page 669</td>
</tr>
<tr>
<td>“Browsing an application”</td>
<td>page 670</td>
</tr>
<tr>
<td>“Deleting an application”</td>
<td>page 671</td>
</tr>
<tr>
<td>“Creating a task ID from an existing application”</td>
<td>page 671</td>
</tr>
<tr>
<td>“Listing task IDs for an application”</td>
<td>page 672</td>
</tr>
</tbody>
</table>

**NOTE**

The profiles must exist when you create an application.
Creating an application

1 On the CHANGE MANAGER Main Menu, select **CM/PILOT**, and press **Enter**.

   The CM/PILOT Main Menu is displayed.

2 Select **Applications** and press **Enter**.

   The Application Action Menu is displayed.

3 Type the name for the **Application** and select **Create an Application**. Then, press **Enter**.

   The Create Application panel is displayed.

4 Specify the information for an application.

   **A** Specify the name of a profile for the task IDs that use the application or specify a wildcard pattern. You can specify the following types of profiles:

   - inbound migrate
   - outbound migrate
   - DDL baseline
   - catalog baseline

   **NOTE**
   An explicit profile name is in owner.identifier form. If either the owner or identifier part is missing, the name is considered a wildcard pattern.

   **B** *(optional)* Specify the work ID owner in the **WORKID Name Template**. If you do not specify a work ID owner, it will default to the creator of the task ID. The name template will be used for new work IDs that are created by task IDs that use the application. For more information about work ID name templates, see page 629.

   **C** *(optional)* Specify a **Comment** to describe the application.

   **D** Press **Enter** to validate the data, and then press **END**.
Listing applications

You can perform various actions from a list of applications. The Application Selection List panel displays information about the application owner, name, date created, and comments.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.

2 Select Applications and press Enter.
   The Application Action Menu is displayed.

3 Specify a wildcard pattern for Application and select List Applications. Then, press Enter.
   The Application Selection List is displayed.

4 Type an action code in the Act column adjacent to the application.

5 Press END.
   The Application Action Menu is displayed.

TIP

Consider the following items when you list applications:

- You can also list applications when you create or edit a Task ID. Specify a wildcard pattern in the Application field of the Create TASKID or Edit TASKID panels.

- You can also list the task IDs that use an application by selecting List TASKIDs for an Application on the Application Action Menu panel.

Editing an application

You can change information about an application.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.
   The CM/PILOT Main Menu is displayed.
Browsing an application

2 Select Applications and press Enter.

The Application Action Menu is displayed.

3 Type the name for the Application or specify a wildcard pattern and select Edit an Application. Then, press Enter.

The Edit Application panel is displayed.

4 You can change the names of the profiles, the work ID name template, and comments. Then, press END.

TIP
You can also edit an application by selecting List Applications on the Application Action Menu panel. For more information, see “Listing applications” on page 669.

Browsing an application

Browsing an application enables you to view read-only information about the application.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

The CM/PILOT Main Menu is displayed.

2 Select Applications and press Enter.

The Application Action Menu is displayed.

3 Type the name for the Application or specify a wildcard pattern and select Browse an Application. Then, press Enter.

The Browse Application panel is displayed.

4 Press END.

TIP
You can also browse an application by selecting List Applications on the Application Action Menu panel. For more information, see “Listing applications” on page 669.
Deleting an application

You can delete applications when you no longer need them.

NOTE

You cannot delete an application that is used by a task ID. You must first either change or delete the associated task IDs.

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select Applications and press Enter.

   The Application Action Menu is displayed.

3. Type the name for the Application or specify a wildcard pattern and select Delete an Application. Then, press Enter.

   The Application Delete Confirmation panel is displayed.

4. Select Delete Application and press Enter.

   TIP

   You can also delete an application by selecting List Applications on the Application Action Menu panel. For more information, see “Listing applications” on page 669.

Creating a task ID from an existing application

You can create a task ID from an existing application. Creating a task ID is the first step in using CM/PILOT to perform CHANGE MANAGER processes. All work in CM/PILOT is accomplished by processing task IDs.

1. On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

   The CM/PILOT Main Menu is displayed.

2. Select Applications and press Enter.

   The Application Action Menu is displayed.
Listing task IDs for an application

3 Type the name for the Application or specify a wildcard pattern and select Create a TASKID from an existing Application. Then, press Enter.

The Script Selection List panel is displayed.

4 Select the script that you want to use and press Enter.

The Create TASKID panel is displayed. For more information about creating a task ID, see “Creating a task ID” on page 628.

---

**TIP**

You can also create a task ID from an existing application by selecting List TASKIDs for an Application on the Application Action Menu panel and then typing C in the Act column adjacent to the task ID in the TASKID Selection List panel. For more information, see “Listing task IDs for an application” on page 672.

---

### Listing task IDs for an application

You can list the task IDs that are associated with an application.

1 On the CHANGE MANAGER Main Menu, select CM/PILOT, and press Enter.

The CM/PILOT Main Menu is displayed.

2 Select Applications and press Enter.

The Application Action Menu is displayed.

3 Type the name for the Application or specify a wildcard pattern and select List TASKIDs for an Application. Then, press Enter.

The TASKID Selection List panel is displayed.

4 Type an action code in the Act column adjacent to the task ID.

---

**TIP**

You can also list task IDs for an application by selecting List Applications on the Application Action Menu panel and then typing L in the Act column adjacent to the application in the Application Selection List panel. For more information, see “Listing applications” on page 669.
Using utilities with ALTER and CHANGE MANAGER

This appendix presents the following topics:

Overview ........................................... 674
Using the unload utilities ................................ 674
  Using the BMC BASIC UNLOAD utility .................. 675
  Using the BMC UNLOAD PLUS utility ..................... 676
  Using the IBM UNLOAD utility .......................... 686
  Using the BMC RECOVER PLUS utility ................... 689
  Using the EXPORT and IMPORT commands ................. 691
  Unloading Unicode, LOB, and XML data .................. 694
Using the load utilities .................................. 694
  Using the BMC LOADPLUS utility ....................... 695
  Using the IBM LOAD utility ............................ 697
  Loading Unicode, LOB, and XML data .................... 697
  Migrating tables with ROWID columns .................... 698
Using the check utilities ................................ 698
  Using the BMC CHECK PLUS utility ..................... 698
  Using the IBM CHECK DATA utility ..................... 698
Using the reorg utilities ................................ 699
  Using the BMC REORG PLUS utility ..................... 700
  Using the IBM REORG utility ........................... 701
  Generating SHRLEVEL reorgs ............................ 701
  Dynamically allocating reorg data sets ................... 703
  Considerations for using the utilities .................. 703
Using the recover and rebuild utilities .................... 705
  Using the BMC RECOVER PLUS utility .................. 706
  Using the IBM REBUILD INDEX utility .................. 706
  Considerations for using the utilities .................. 706
Using the copy utilities .................................. 707
  Using the BMC COPY PLUS utility ....................... 708
  Using the IBM COPY utility ............................ 708
  Copying and registering the data sets .................... 708
Overview

The Front End of the ALTER and CHANGE MANAGER products generates keywords in the ALUIN input stream. Using the ALUIN input stream, the Analysis component generates utility commands in the worklist. These keywords and commands perform the following functions:

- unload and load data
- check referential constraint violations
- reorganize table spaces and indexes
- create image copies
- generate and update catalog statistics

You can use the BMC utilities or the IBM utilities to perform these functions. If you select a utility to use, that utility must be installed.

**NOTE**

If the DDL for a table space or an index sets DEFINE to NO, CHANGE MANAGER does not use the utilities to perform any of the preceding functions.

Using the unload utilities

When the Analysis component must modify a table or table space by dropping and rebuilding it, it uses an unload utility to unload the data. It then uses a load utility to reload the data.

You can select which unload utilities to use from the Analysis Utility Options panel (see Figure 194). You can choose from the following utilities:

- BMC BASIC UNLOAD
- BMC UNLOAD PLUS
- IBM UNLOAD
You can also choose the BMC RECOVER PLUS utility to migrate data from image copy data sets.

**NOTE**

To run a CHANGE MANAGER worklist in parallel, you must use the UNLOAD PLUS utility. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”

**Figure 194  Analysis Utility Options panel**

```
ALUFOAN4 ------------------ Analysis Utility Options --------------------------
Command ===> 
Type information. Then press Enter to continue or PF12 for previous panel.

Unload . . . 3 1. BASIC UNLOAD (included with ALTER or CHANGE MANAGER)
  2. BMC UNLOAD PLUS
     _ Use LOBDATAMOVER for LOBs
  3. IBM UNLOAD
  4. BMC UNLOAD PLUS from Image Copies (ICs)
  5. BMC RECOVER PLUS from ICs
     (included with ALTER or CHANGE MANAGER)
  6. BMC COPY PLUS EXPORT/ BMC RECOVER PLUS IMPORT
     (Database Administration or Recovery Management Solution)

Load . . . . 1 1. IBM LOAD LOG NO   2. IBM LOAD LOG YES   3. BMC LOADPLUS

Check . . . . 1 1. IBM CHECK       2. BMC CHECK PLUS

Rebuild . . . 1 1. IBM REBUILD INDEX 2. BMC RECOVER PLUS
            3. No rebuild index

Commands:  HELP END PREVIOUS
```

**Using the BMC BASIC UNLOAD utility**

The BASIC UNLOAD utility is a utility for unloading DB2 data. When you installed ALTER or CHANGE MANAGER, this utility was also installed. The BASIC UNLOAD utility uses a portion of the functionality of the BMC UNLOAD PLUS utility. For more information about the UNLOAD PLUS utility, see the UNLOAD PLUS for DB2 Reference Manual.

Selecting BASIC UNLOAD inserts the BASICUNLOAD keyword into the ALUIN input stream. When the work ID is analyzed, the Analysis component inserts the -BMCB command into the worklist.
Unloading data

The BASIC UNLOAD utility uses DB2 dynamic SQL programming techniques to process a SELECT statement and read data in a table. The utility retains the format of the data (EBCDIC, ASCII, or Unicode). You can use the BASIC UNLOAD utility and the BMC LOADPLUS or IBM LOAD utility to unload and load objects that are defined as Unicode.

Migrating tables with ROWID columns

You can modify or migrate tables that contain ROWID columns with data. If the ROWID column is defined as GENERATED ALWAYS, ALTER and CHANGE MANAGER omit the ROWID column in the UNLOAD and LOAD statements. If the ROWID column is defined as GENERATED BY DEFAULT, ALTER and CHANGE MANAGER include the ROWID column in the UNLOAD and LOAD statements. The column is unloaded last in the sequence of columns and is also loaded last, even though it might not be the last column in the table. When a table that contains a ROWID column is unloaded, Analysis lists all of the columns that are unloaded in the UNLOAD statement in the worklist.

Using the BMC UNLOAD PLUS utility

The BMC UNLOAD PLUS utility is a high-performance utility for unloading DB2 Unicode, LOB, and XML data. You can use the UNLOAD PLUS utility to perform the following functions:

- multitask unloading and loading data
- use fast unload
- use separate unload data sets
- migrate tables with ROWID columns
- unload data in XML and LOB columns
- dynamically allocate unload data sets
- create an explicit column list
- unload from image copies

Selecting BMC UNLOAD PLUS inserts the BMCUNLOAD keyword into the ALUIN input stream. When the work ID is analyzed, the Analysis component inserts the -BMCD command into the worklist and generates a control card that explicitly specifies the UNLOAD PLUS default options.

The -BMCD command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”
Multitasking unloading and loading data

When you specify to use UNLOAD PLUS and the BMC LOADPLUS utility, the utilities can use separate data sets for each table space partition and can multitask the unloading and the loading of data. The utilities can multitask the unloading and loading of data for the following actions:

- changing a partitioned table space from index controlled to table controlled
- changing a partitioned table space from table controlled to index controlled
- adding or dropping partitions in a table-controlled partitioned table space
- unloading and loading data in a partitioned table space

The MAXSYSREC keyword in the ALUIN input stream specifies the maximum number of dynamically allocated unload (SYSREC) data sets to use for each unload and load worklist command. If the number of partitions in the table space exceeds the value specified for the MAXSYSREC keyword, you must specify the DYNUNLD keyword in the ALUIN input stream to multitask the unloading and loading of data, as the product will create multiple unload and load commands for each table.

For multitasking to occur, the following conditions must exist:

- A partitioned table space must be specified for migration or for a drop and create action.

  Multitasking is required when a partitioned table space contains more than 256 partitions.

- The BMCUNLOAD and BMCLOAD keywords must be included in the ALUIN input stream.

- Data must be unloaded from a table, not an image copy.

When multitasking occurs, tape stacking for the unload (SYSREC) data sets is disabled. If tape stacking is required, you can add the NOMTASK keyword to the ALUIN input stream to disable multitasking.

**NOTE**

Analysis does not perform multitasking if you are taking or recovering to a full-recovery baseline. For information about establishing full-recovery baselines, see Chapter 2, “Taking a snapshot of a database environment.” For information about recovering to a full-recovery baseline, see Chapter 5, “Maintaining database environments.”
Using fast unload

When you specify to use the UNLOAD PLUS and LOADPLUS utilities, the utilities can quickly unload data from one table and load it into another table that has a similar structure. When you use a migrate-type work ID, you can copy data in one DB2 table or DB2 subsystem and move it to another table or DB2 subsystem. When you use an alter-type work ID, you can make changes to a table that do not affect the structure of the table, such as changing the table space name.

To enable this feature, the BMCFASTL installation option must be set to Y. If BMCFASTL is set to N, you must manually change the NOBMCFASTLOAD keyword to BMCFASTLOAD in the ALUIN input stream. When the Alter- or migrate-type work ID is analyzed, Analysis inserts the -BMCD command with the FORMAT BMCLOAD control card into the worklist.

NOTE
You can use the fast unload feature to unload and load objects that are defined as Unicode. The fast unload feature requires the CCSID definition of the table space to match the default CCSID definition of the subsystem that is specified in the DSNHDECP load module. If the definitions do not match, the product does not use the feature to unload and load data.

Unloading data in separate unload data sets

When you use the UNLOAD PLUS utility to unload data, the Analysis component can generate a single -BMCD worklist command for a table space that contains multiple tables. To unload data in each table into a separate unload (SYSREC) data set, you must manually add the MULTITBDS keyword to the ALUIN input stream. In addition, you must select the Use Dynamic SYSRE datasets option on the Analysis Utility Dataset Options panel (see Figure 196).

Migrating tables with ROWID columns

For information about migrating tables with ROWID columns, see “Migrating tables with ROWID columns” on page 676.

Unloading data in XML and LOB columns

When you use the UNLOAD PLUS utility to unload data contained in XML or LOB columns, the utility unloads the data to a file reference output data set that contains the data. The utility requires that the file reference partitioned data sets be dynamically allocated and on DASD.
Using the BMC UNLOAD PLUS utility

**NOTE**

If you specify NODYNUNLD, the utility statically allocates the unload (SYSREC) and discard data sets. However, the utility dynamically allocates the file reference data sets.

JCL Generation uses the symbolic variable that is specified in the JCL Generation Data Set Options for File Reference Sysrec panel to create OUTPUT descriptors for file references for Analysis. When UNLOAD PLUS dynamically allocates the unload and file reference data sets, it uses the OUTPUT descriptors listed in Table 51.

**Table 51  OUTPUT descriptors used by BMC UNLOAD PLUS**

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYRC</td>
<td>identifies the unload (SYSREC) data set</td>
</tr>
<tr>
<td>SYLBnn</td>
<td>references a file reference data set that contains LOB data</td>
</tr>
<tr>
<td></td>
<td><em>nn</em> represents a sequence number for each LOB column.</td>
</tr>
<tr>
<td>SYXCnn</td>
<td>references a file reference data set that contains XML data</td>
</tr>
<tr>
<td></td>
<td><em>nn</em> represents a sequence number for each XML column.</td>
</tr>
</tbody>
</table>

For the Database Administration solution, you can also use the LOB DATA MOVER program to unload and load data that is contained in LOB columns. To use the LOB DATA MOVER program, the following requirements must be met:

- You must have a password for the Database Administration solution.
- On the Analysis Worklist Command Options panel, you must select the **Data** option.
- On the Analysis Utility Options panel, you must select the BMC UNLOAD PLUS and BMC LOADPLUS utilities, and the option to use the LOB DATA MOVER.
- You must use dynamic unload (SYSREC) data sets.

The BMC UNLOAD PLUS utility unloads the base table, creating two separate unload (SYSREC) data sets:

- One data set contains all of the columns in the table, except the LOB columns.
- One data set contains the ROWID column.

Then the LOB DATA MOVER program unloads the LOB columns, using the ROWID column in the unload (SYSREC) data set. When the LOB DATA MOVER program is used to unload data, Analysis inserts the -UNLB command into the worklist.
The BMC LOADPLUS utility loads all of the columns in the base table, except the LOB columns. The utility loads the ROWID column in the base table. The LOB DATA MOVER program loads the LOB columns. When the LOB DATA MOVER program is used to load data, Analysis inserts the -LOLB command into the worklist.

Periodically, the LOB DATA MOVER program issues an SQL COMMIT statement. On the Analysis LOB DATA MOVER Commit Options panel (see Figure 195), you can specify whether the COMMIT statement is issued based on the number of rows that have been updated or by the amount of data that has been updated.

- The value that you specify for **Commit frequency** indicates the number of rows that can be loaded into a LOB column before an SQL COMMIT statement is issued. When you specify this value, the product inserts the COMMITFREQ keyword into the ALUIN input stream.

- The value that you specify for **Commit size** indicates the number of megabytes (MB) of data that can be loaded into a LOB column before the SQL COMMIT statement is issued. When you specify this value, the product inserts the COMMITSIZE keyword into the ALUIN input stream.

The LOB DATA MOVER program issues the SQL COMMIT statement when either the value for the frequency or the size is met.

**Figure 195  Analysis LOB DATA MOVER Commit Options panel**

```
ALUFOAN8 ----------- Analysis LOB DATA MOVER Commit Options -------------------
Command ===>
Type information. Then press Enter to continue or PF12 for previous panel.
These options are used only when the Database Administration solution uses BMC UNLOAD PLUS and BMC LOADPLUS to unload and load LOB data.
Commit frequency . . . 2500  (1-9999 Rows)
Commit size . . . . . 100   (1-1000 Megabytes)
```

Commands:  HELP END
Consider the following items when you choose to use the LOB DATA MOVER to unload and load LOB data:

■ The LOB DATA MOVER program can unload LOB data on tape.

■ If the length of a LOB column is defined as greater than 32 megabytes (MB), the LOB DATA MOVER can be used to unload and load LOB data only if the following conditions exist:
  — The LOBGT32M keyword is included in the ALUIN input stream.
  — The Use Dynamic SYSRE datasets option on the Analysis Utility Dataset Options panel is selected.

■ If the LOB column in the base table is defined as NOT NULL WITHOUT DEFAULT, data cannot be loaded into the table.

■ The ROWID SYSREC data set is used by the LOB DATA MOVER program to unload data contained in a ROWID column. BMC recommends that you specify SYSDA for the unit name for a ROWID SYSREC unload data set so that the performance of the worklist commands for unloading the ROWID data set can be improved when the worklist is executed in parallel. For more information about worklist parallelism, see Appendix D, “Using worklist parallelism.”

■ CHANGE MANAGER supports the conversion of a VARCHAR or LONG VARCHAR data type to a BLOB or CLOB data type. If a column is already defined as a LOB data type, it cannot be converted. For more information about converting the data type, see “Converting a VARCHAR data type to a LOB data type” on page 543.

■ You cannot use the LOB DATA MOVER to generate full-recovery baselines of tables that include a LOB column.

■ You cannot use the LOB DATA MOVER to unload and load data in a table if the CCSID is defined as UNICODE.

■ The BMC UNLOAD PLUS from Image Copies option on the Analysis Utility Options panel cannot be used to unload LOB data.

■ If you are performing a data-only migration and you have an update trigger on a base table, the trigger might execute when the LOB DATA MOVER program updates the value of the LOB column in the base table.

■ The LOB DATA MOVER program can process a column name that is delimited, provided that the name does not contain embedded blanks.
Dynamically allocating unload data sets

UNLOAD PLUS and LOADPLUS can dynamically allocate the data sets that are used by the utilities to unload and load data. On the Analysis Utility Dataset Options panel (see Figure 196), you can select to use the dynamically allocated data sets and to override any JCL Generation options that apply to dynamic allocation.

Figure 196 Analysis Utility Dataset Options panel

When you select to use UNLOAD PLUS, LOADPLUS, and Use Dynamic SYSRE datasets, the product uses the utilities and inserts the DYNUNLD keyword into the ALUIN input stream. When you select Override Defaults for Dynamic Allocation, the JCL Generation Main Menu is displayed. You can select Tape Options and Individual Data Set Options to override the options that relate to dynamic allocation. For information about setting these options, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

NOTE

Before you select Override Defaults for Dynamic Allocation, you must select Use Dynamic SYSRE datasets.
Considerations for dynamically allocating data sets

Consider the following items when you choose to dynamically allocate the unload data sets:

- The data set name that is used for an object should be the same for both UNLOAD PLUS and LOADPLUS. Do not modify the name of the UNLOAD data set for an object without modifying the name of the corresponding LOAD data set for the object in the worklist.

- You cannot stack the unload (SYSREC) data sets on a tape.

- To run a CHANGE MANAGER worklist in parallel, you must use dynamically allocated unload data sets. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”

- To multtask the unloading and loading of data in partitioned table spaces that contain more partitions than the value of the MAXSYSREC keyword, you must use dynamically allocated unload (SYSREC) data sets. The default value of the MAXSYSREC keyword is 256.

The UNLOAD PLUS utility performs the following actions when you choose to dynamically allocate the unload data sets:

- If the UNLOAD PLUS utility cannot unload a file, it deletes the dynamically allocated unload (SYSREC) data set.

- In the Database Administration solution, when you specify to use the UNLOAD PLUS and LOADPLUS utilities to unload and load LOB data, the UNLOAD PLUS utility dynamically allocates a ROWID SYSREC data set (ROWIDDSN). The LOB DATA MOVER program dynamically allocates the LOB SYSREC data set (LOBDSN). For more information about unloading and loading LOB data, see “Unloading data in XML and LOB columns” on page 678.

- When UNLOAD PLUS dynamically allocates the unload data sets, it uses OUTPUT descriptors in the worklist for the primary data sets. JCL Generation uses the symbolic variables that are specified in the JCL Generation Data Set Options for SYSRECS panels to create a fully qualified data set name.

**NOTE**

If statistics are unavailable for a table, or if the table spaces are compressed, you can modify the values for the PCTPRIM and NBRSECD installation options for the UNLOAD PLUS utility. For more information about dynamically allocated data sets, see the UNLOAD PLUS for DB2 Reference Manual.
The product performs the following actions when you choose to dynamically allocate the unload data sets and unload the data sets from image copies:

- If any tables in the table space do not have statistics, Analysis uses the default sizes for the primary quantity and secondary quantity that are specified in the POF for the unload (SYSREC) data sets. In the worklist, Analysis specifies these sizes in the SPACE parameter in the OUTPUT descriptor in the -BMCD worklist command.

- If all tables in the table space have statistics, Analysis includes the ESTROWS parameter in the -BMCD worklist command, and UNLOAD PLUS uses the ESTROWS estimate to calculate the size of the data sets.

**Creating an explicit column list**

You can specify whether to create a worklist with an explicit column list in the Analysis Unload / Copy Options panel (see Figure 197). When you select *Explicit column list with BMC UNLOAD PLUS*, Analysis lists all of the columns that are unloaded for a table in the UNLOAD statement in the worklist. The product inserts the UNLDCOLL keyword into the ALUIN input stream.

---

**Figure 197 Analysis Unload / Copy Options panel—UNLOAD PLUS**

ALUFOAN7 --------------- Analysis Unload / Copy Options -----------------------
Command ===>

Type information. Then press Enter to continue or PF12 for previous panel.

BMC UNLOAD PLUS from Image Copy (IC) Options

1. Relative IC dataset determined at runtime by BMC UNLOAD PLUS
2. Relative IC dataset determined during analysis
0 Relative version of IC to be used (0=latest, -1 to -9999)

1. Use Share Level CHANGE or REFERENCE Image Copies
2. Use Share Level REFERENCE Image Copies Only

Commands: HELP END PREVIOUS
Unloading data from image copies

When you select **BMC UNLOAD PLUS from Image Copies (ICs)**, Analysis uses the UNLOAD PLUS utility and unloads the data from image copies. Requirements for image copies are listed in Table 52.

**Table 52  Requirements for image copies for UNLOAD PLUS**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Value in SYSIBM.SYSCOPY catalog table</th>
</tr>
</thead>
<tbody>
<tr>
<td>cataloged to an integrated catalog facility (ICF) catalog</td>
<td>DSVOLSER = blank</td>
</tr>
<tr>
<td>full image copy</td>
<td>ICTYPE = F</td>
</tr>
<tr>
<td>local primary copy</td>
<td>ICBACKUP = blank</td>
</tr>
<tr>
<td>DB2 image copy (not a concurrent or an inline image copy)</td>
<td>STYPE = blank</td>
</tr>
<tr>
<td>image copies that are marked as SHRLEVEL REFERENCE or SHRLEVEL CHANGE, or SHRLEVEL REFERENCE only must exist</td>
<td>SHRLEVEL = R or SHRLEVEL = C</td>
</tr>
<tr>
<td>relative version number of the image copy must exist</td>
<td>not applicable</td>
</tr>
</tbody>
</table>

Using a specific image copy

You might want Analysis to choose the image copy if additional copies might be taken before the worklist is executed, and you want to ensure that a specific image copy that was taken at a point of consistency is used. You can specify when the image copy data set is determined at runtime or during analysis in the Analysis Unload / Copy Options panel (see Figure 197):

- Selecting **Relative IC dataset determined at runtime by BMC UNLOAD PLUS** inserts the UNLOADCOPY xx keyword (where xx is the relative image copy number) into the ALUIN input stream. UNLOAD PLUS uses the INFILE IMAGECOPY FULL-xx parameters and determines which data set to use. Select this option for image copies that are taken at the partition level (that is, DSNUM is not 0 in the SYSIBM.SYSCOPY catalog table) or the table space level (that is, DSNUM is 0).

- Selecting **Relative IC dataset determined during analysis** inserts the UNLOADCOPYDS xx keyword (where xx is the relative image copy number) into the ALUIN input stream. Analysis includes the name of the data set in the unload section of the worklist. UNLOAD PLUS uses the INFILE <ddname> syntax. Select this option for image copies that are taken at the table space level only (that is, DSNUM is 0 in the SYSIBM.SYSCOPY catalog table).
Specifying a relative version of the image copy

Analysis determines which image copies in the SYSIBM.SYSCOPY catalog table meet the requirements listed in Table 52. Analysis sorts the results in descending order by timestamp. Analysis then uses these results to determine the relative version number that you specify on the Analysis Unload / Copy Options panel. The relative image copy number of 0 is always the latest version.

**NOTE**

If you request a specific version of an image copy in the Relative version of IC to be used option and that version does not exist, Analysis performs the following actions:

- If you selected Relative IC dataset determined at runtime by BMC UNLOAD PLUS, Analysis issues a warning message and builds the worklist.
- If you selected Relative IC dataset determined during analysis, Analysis terminates with an error.

Using SHRLEVEL REFERENCE or SHRLEVEL CHANGE image copies

You can also specify whether to use image copies that were marked as SHRLEVEL CHANGE or SHRLEVEL REFERENCE. SHRLEVEL specifies the level of access that concurrently operating DB2 applications and utilities have to table spaces when the image copy was taken.

- Selecting Use Share Level CHANGE or REFERENCE Image Copies inserts the SHRLEVELCHG keyword into the ALUIN input stream. UNLOAD PLUS unloads from full image copies that are marked as either SHRLEVEL REFERENCE or SHRLEVEL CHANGE.
- Selecting Use Share Level REFERENCE Image Copies Only inserts the SHRLEVELREF keyword into the ALUIN input stream. UNLOAD PLUS unloads from full image copies that are marked as SHRLEVEL REFERENCE.

Using the IBM UNLOAD utility

You can use the IBM UNLOAD utility to perform the following functions:

- dynamically allocate unload data sets
- use multitasking
- unload data in LOB columns and XML columns
- unload Unicode data
- migrate tables with ROWID columns
Selecting **IBM UNLOAD** inserts the IBMUNLOAD keyword into the ALUIN input stream. When the work ID is analyzed, the Analysis component inserts the -UNLI command into the worklist.

**NOTE**
If you select IBM UNLOAD to perform an unload, you must select **IBM LOAD LOG NO** or **IBM LOAD LOG YES** to perform the load.

### Dynamically allocating unload data sets

IBM UNLOAD and IBM LOAD can dynamically allocate the data sets that are used by the utilities to unload and load data. On the Analysis Utility Dataset Options panel (see Figure 196), you can select to use the dynamically allocated data sets and to override any JCL Generation options that apply to dynamic allocation.

When you select to use UNLOAD, LOAD, and **Use Dynamic SYSRE datasets**, the product uses the utilities and inserts the DYNUNLD keyword into the ALUIN input stream. When you select **Override Defaults for Dynamic Allocation**, the JCL Generation Main Menu is displayed. You can select **Tape Options** and **Individual Data Set Options** to override the options that relate to dynamic allocation. For information about setting these options, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*.

**NOTE**
Before you select **Override Defaults for Dynamic Allocation**, you must select **Use Dynamic SYSRE datasets**.

### Considerations for dynamically allocating data sets

Consider the following items when you choose to dynamically allocate the unload data sets:

- You cannot stack the unload (SYSREC) data sets on a tape.

- If the IBM UNLOAD utility fails while unloading the data, you must manually delete the appropriate unload (SYSREC) data sets before restarting the worklist. IBM UNLOAD does not automatically delete the dynamically allocated data sets.
Multitasking the unloading and loading of data

When you specify to use the IBM UNLOAD and LOAD utilities, the utilities can multitask the unloading and loading of data. For multitasking to occur, the following conditions must exist:

- A partitioned table space must be specified for migration or for a drop and create action.
- The unload (SYSREC) data set must be dynamically allocated.
- The IBMUNLOAD and IBMLOAD keywords must be included in the ALUIN input stream.

When multitasking occurs, tape stacking for the unload (SYSREC) data sets is disabled. If tape stacking is required, you can add the NOMTASK keyword to the ALUIN input stream to disable multitasking.

Unloading data in LOB and XML columns

The IBM UNLOAD utility can be used to unload data that is contained in LOB and XML columns. The utility requires that the file reference partitioned data sets be dynamically allocated and on DASD.

**NOTE**

If you specify NODYNUNLD, the utility statically allocates the unload (SYSREC) and discard data sets. However, the utility dynamically allocates the file reference data sets.

JCL Generation uses the symbolic variable that is specified in the JCL Generation Data Set Options for File Reference Sysrec panel to create TEMPLATE descriptors for file references for Analysis. When IBM UNLOAD dynamically allocates the unload and file reference data sets, it uses the TEMPLATE descriptors listed in Table 53.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYRC</td>
<td>identifies the unload (SYSREC) data set</td>
</tr>
<tr>
<td>SYLBNn</td>
<td>references a file reference data set that contains LOB data</td>
</tr>
<tr>
<td></td>
<td>( nn ) represents a sequence number for each LOB column.</td>
</tr>
<tr>
<td>SYXCNn</td>
<td>references a file reference data set that contains XML data</td>
</tr>
<tr>
<td></td>
<td>( nn ) represents a sequence number for each XML column.</td>
</tr>
</tbody>
</table>
Migrating tables with ROWID columns

For information about migrating tables with ROWID columns, see “Migrating tables with ROWID columns” on page 676.

Using the BMC RECOVER PLUS utility

The BMC RECOVER PLUS utility enables you to migrate data from image copy data sets. When you installed ALTER or CHANGE MANAGER, this functionality of RECOVER PLUS was also installed. A RECOVER PLUS password is not needed to access the functionality; you can access it with an ALTER, CHANGE MANAGER, Administrative Assistant, or Database Administration password. To use any additional functionality of RECOVER PLUS, you must provide a valid RECOVER PLUS password or Recovery Management for DB2 solution password.

If you select the RECOVER PLUS utility, the Analysis component uses image copies of existing data sets as input to copy the data to the receiving subsystem. You can use image copies taken at a quiesce point to ensure data consistency and to avoid contention with an unload, such as avoiding locks on the VSAM data sets on the sending subsystem. The image copies must meet the requirements that are listed in Table 52.

When you select BMC RECOVER PLUS from ICs to analyze your migrate-type work ID (see Figure 194 on page 675), the product uses the RECOVER PLUS utility and inserts the RESMAXIC nnnnn keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCV command into the worklist and includes the name of the data set.

NOTE

You can select RECOVER PLUS only when no structural changes to the table or table space exist. You cannot specify scope rules at the table level, and the exclude-type change rules cannot exclude tables.

You cannot select the BMC RECOVER PLUS from ICs option to recover LOB data.

WARNING

If you select RECOVER PLUS for a data-only migration and structural differences exist between the sending and receiving subsystems, migrated data might become invalid or unusable.
Specifying a relative version of the image copy

Analysis determines which image copies in the SYSIBM.SYSCOPY catalog table meet the requirements listed in Table 52. Analysis sorts the results in descending order by timestamp. Analysis then uses these results to determine the relative version number that you specify on the Analysis Unload / Copy Options panel (see Figure 198). The relative image copy number of 0 is always the latest version.

Figure 198 Analysis Unload / Copy Options panel—RECOVER PLUS

Using SHRLEVEL CHANGE or SHRLEVEL REFERENCE copies

You also have the option to specify whether to use image copies that were marked as SHRLEVEL CHANGE or SHRLEVEL REFERENCE (see Figure 198 on page 690). SHRLEVEL specifies the level of access that concurrently operating DB2 applications and utilities had to table spaces when the image copy was taken.

- Selecting Use Share Level CHANGE or REFERENCE Image Copies inserts the SHRLEVELCHG keyword into the ALUIN input stream. RECOVER PLUS uses full image copies that are marked as either SHRLEVEL REFERENCE or SHRLEVEL CHANGE.

- Selecting Use Share Level REFERENCE Image Copies Only inserts the SHRLEVELREF keyword into the ALUIN input stream. RECOVER PLUS uses full image copies that are marked as SHRLEVEL REFERENCE.
Using the EXPORT and IMPORT commands

The BMC COPY PLUS EXPORT command and BMC RECOVER PLUS IMPORT command enable you to migrate an image copy or set of image copies within a DB2 subsystem or to another DB2 subsystem.

The EXPORT command runs on the sending subsystem during phase 1 of the migration to collect information on one or more table spaces. This information includes object identifiers, the names of image copies, and other information relating to the sending table spaces. The EXPORT command includes this information in a sequential migration file.

The IMPORT command runs on the receiving subsystem during phase 2 of the migration to migrate all the image copy data to the receiving table space that is identified by the command. The IMPORT command uses the object identifiers found in the EXPORT command’s migration file and translates them to the object identifiers found in the receiving catalog.

The EXPORT and IMPORT commands comprise the Copy Migration feature of the Recovery Management and Database Administration solutions. When you use the Copy Migration feature, the following requirements must be met:

- You must have a valid BMC Database Administration solution or BMC Recovery Management solution password.
- No structural changes to the table or table space exist.
- The table spaces cannot contain XML data.
- Objects must be migrated at the table space level.
- For DB2 Version 10 subsystems, the IBM IMPDSDEF subsystem parameter must be set to YES. (If the parameter is set to NO and the data set does not exist on the receiving subsystem, RECOVER PLUS fails.)
- An existing registered local primary copy data set must exist.

When you select **BMC COPY PLUS EXPORT/BMC RECOVER PLUS IMPORT** to analyze your migrate-type work ID (see Figure 194 on page 675), the product uses the COPY PLUS and RECOVER PLUS utilities and inserts the COPYEXPORT keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCC and -BMCV commands into the worklist.

Note the following items when you use the Copy Migration feature:

- The feature supports the migration of LOB data from the image copies of the LOB table spaces.
Using the EXPORT and IMPORT commands

- The EXPORT command always uses the most current version of either SHRLEVEL REFERENCE or SHRLEVEL CHANGE image copies.

- The feature will not re-create all indexes for all objects with the IMPORT command. CHANGE MANAGER generates the RECOVER PLUS REBUILD INDEX(ALL) command to re-create the indexes.

- When an image copy contains auxiliary table spaces, CHANGE MANAGER invokes the IBM CHECK LOB utility. The CHECK LOB utility identifies inconsistencies in the auxiliary table spaces and invalid LOB values.

- To migrate history table spaces, you must explicitly include them in the scope.

- When an image copy contains a clone table, CHANGE MANAGER generates separate EXPORT and IMPORT commands for the base table and the clone table.

- The feature supports objects that are defined in an implicit database or table space. The table space must not be defined with DEFINE NO.

- The feature supports the migration of a simple table space.

**Specifying the number of table spaces per file**

When you use the Copy Migration feature, you can specify the number of table spaces to include in a single migration file that the COPY PLUS EXPORT command creates. The valid range of values is 0 through 9999. The default value is 100. A value of 0 indicates that all table spaces in the scope will be included in a single file. You specify the value on the Analysis Unload / Copy Options panel (see Figure 199).
Using the EXPORT and IMPORT commands

Synchronizing the table spaces in the migration file

You can also specify whether the migration file should include all spaces in the IMPORT command or only those spaces that have changed since the last migration (see Figure 199).

- Selecting Use SYNC REPLACE option for RECOVER PLUS IMPORT replaces all table spaces regardless of whether they have changed or not. CHANGE MANAGER inserts the RPLUSYNCREPLACE keyword into the ALUIN input stream.

  Consider this option when migrating to a subsystem in which the data is manipulated in test environment.

- Selecting Use SYNC AUTO option for RECOVER PLUS IMPORT replaces only table spaces that have changed since the last time those table spaces were imported. CHANGE MANAGER inserts the RPLUSYNCAUTO keyword into the ALUIN input stream.

  Consider this option when migrating to a query-based environment when the table spaces are read only.

The SYNC option is useful in cases in which the image copy indicated by the current EXPORT file is identical to the spaces imported previously. The option that you select overrides the value of the RPLUSYNC installation option.
Unloading Unicode, LOB, and XML data

ALTER and CHANGE MANAGER can use the unload and load utilities shown in Table 54 to process Unicode, LOB, and XML data.

Table 54  Utilities used to process Unicode, LOB, and XML data

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Unload</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicode</td>
<td>BMC UNLOAD PLUS</td>
<td>BMC LOADPLUS</td>
</tr>
<tr>
<td></td>
<td>BMC UNLOAD PLUS</td>
<td>IBM LOAD</td>
</tr>
<tr>
<td></td>
<td>BMC BASIC UNLOAD</td>
<td>BMC LOADPLUS</td>
</tr>
<tr>
<td></td>
<td>BMC BASIC UNLOAD</td>
<td>IBM LOAD</td>
</tr>
<tr>
<td></td>
<td>IBM UNLOAD</td>
<td>IBM LOAD</td>
</tr>
<tr>
<td>LOB</td>
<td>BMC UNLOAD PLUS</td>
<td>BMC LOADPLUS</td>
</tr>
<tr>
<td></td>
<td>BMC UNLOAD PLUS with LOB DATA MOVER</td>
<td>BMC LOADPLUS with LOB DATA MOVER</td>
</tr>
<tr>
<td>XML</td>
<td>BMC UNLOAD PLUS</td>
<td>BMC LOADPLUS</td>
</tr>
<tr>
<td></td>
<td>IBM UNLOAD</td>
<td>IBM LOAD</td>
</tr>
</tbody>
</table>

Using the load utilities

After the Analysis component has unloaded data by using one of the unload utilities, it uses a load utility to reload the data. Analysis might use the load utilities to build image copies and gather statistics. You can select which load utility to use from the Analysis Utility Options panel (see Figure 194 on page 675). You can choose from the following utilities:

- BMC LOADPLUS
- IBM LOAD

**NOTE**

To run a CHANGE MANAGER worklist in parallel, you must use the LOADPLUS utility. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”
Using the BMC LOADPLUS utility

The BMC LOADPLUS utility is a high-performance utility that loads data from sequential files into DB2 tables. You can use the LOADPLUS utility to create inline image copies, produce BMCSTATS in a statistics report (if the BMC DASD MANAGER PLUS product is installed), and update DB2 catalog statistics. You can also use the LOADPLUS utility with the BASIC UNLOAD utility to unload and load objects that are defined as Unicode.

When you select BMC LOADPLUS, the product uses the LOADPLUS utility and inserts the BMCLOAD keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCL command into the worklist.

NOTE

If a table space is being created in the worklist, Analysis includes the REDEFINE NO parameter in the -BMCL command.

The -BMCL command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”

Converting data with the LOADPLUS utility

You can use the UNLOAD PLUS utility in conjunction with the LOADPLUS utility to convert data.

If you specify to use the UNLOAD PLUS utility with the LOADPLUS utility and data conversions are taking place, Analysis generates a SELECT INTO statement for the UNLOAD PLUS utility to ensure that conversions that will take place on the load are valid (supported). The data is written to a DUMMY DD; this data is not saved. If problems are detected, an error occurs and processing stops. If no errors occur, Analysis invokes the UNLOAD PLUS utility a second time to unload the data.

Using multiple work data sets

You can use multiple SYSUT temporary work data sets to reduce the amount of DASD required for loading data if the index key lengths vary a great deal. In addition, if enough sort work space and memory are available, using multiple SYSUT data sets enables LOADPLUS to build nonclustering indexes for a table concurrently and to improve performance.

You specify the maximum number of SYSUT data sets that you want the LOADPLUS utility to use to build indexes in the Analysis Utility Dataset Options panel (see Figure 196 on page 682). The number of data sets are specified in the MAXSYSUT keyword in the ALUIN input stream.
ALTER and CHANGE MANAGER take the following factors into consideration when MAXSYSUT is specified:

- If the number of indexes in a table is equivalent to or less than the maximum number of SYSUT data sets, the products create a SYSUT data set for each nonclustering index.

  When the work ID is analyzed, the Analysis component specifies the number of SYSUT data sets in the -JCLP command and generates a WORKDDN (SUT) parameter in the -BMCL command in the worklist. In addition, JCL Generation creates SUTnnnn0 DD statements for the tables in the JCL.

- If a table contains more indexes than the maximum number of SYSUT data sets or if the table contains only one index, the products create only one SYSUT data set.

  When the work ID is analyzed, the Analysis component generates a WORKDDN SYSUT001 parameter in the -BMCL command in the worklist. JCL Generation creates a SYSUT001 DD statement in the JCL.

### Using fast load

If you specify to use the UNLOAD PLUS and LOADPLUS utilities, the utilities can quickly unload data from one table and load it into another table that has a similar structure. When you use a migrate-type work ID, you can copy data in one DB2 table or DB2 subsystem and move it to another table or DB2 subsystem. When you use an alter-type work ID, you can make changes to a table that do not affect the structure of the table, such as changing the column names.

To enable this feature, the BMCFASTL installation option must be set to Y. If BMCFASTL is set to N, you must manually change the NOBMCFASTLOAD keyword to BMCFASTLOAD in the ALUIN input stream. When the alter-type or migrate-type work ID is analyzed, Analysis inserts the -BMCL command with the FORMAT BMCUNLOAD control card into the worklist.

**NOTE**

You can use the fast unload feature to unload and load objects that are defined as Unicode. The fast load feature requires the CCSID definition of the table space to match the default CCSID definition of the subsystem that is specified in the DSNHDECP load module. If the definitions do not match, the product does not use the feature to unload and load data.

### Loading data in XML and LOB columns

When you use the LOADPLUS utility to unload data contained in XML or LOB columns, the utility loads the data from a file reference input data set that contains the data.
Using the IBM LOAD utility

You can use the IBM LOAD utility to perform the following functions:

- build image copies and gather statistics
- with the BASIC UNLOAD, UNLOAD PLUS, or IBM UNLOAD utility, unload and load objects that are defined as Unicode
- with the IBM UNLOAD utility, unload and load data contained in LOB and XML columns

When you select **IBM LOAD LOG NO**, the product uses the IBM LOAD utility and inserts the IBMLOAD keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -LOAD command into the worklist.

---

**NOTE**

If a table space is being created in the worklist, Analysis includes the REUSE parameter in the -LOAD command.

---

When you select **IBM LOAD LOG YES** so that you can log the loaded records, the product also inserts the LOG keyword into the ALUIN input stream.

---

**WARNING**

If you are using a tape data set for SYSUT work data sets with IBM LOAD, you must manually edit the worklist and JCL to ensure that multiple instances of LOAD use unique work data sets for ddnames.

---

Loading Unicode, LOB, and XML data

ALTER and CHANGE MANAGER can use the unload and load utilities shown in Table 54 on page 694 to process Unicode, LOB, and XML data.
Migrating tables with ROWID columns

For information about migrating tables with ROWID columns, see “Migrating tables with ROWID columns” on page 676.

Using the check utilities

You can select which check data utilities to use from the Analysis Utility Options panel (see Figure 194 on page 675). You can choose from the following utilities:

- BMC CHECK PLUS
- IBM CHECK DATA

Using the BMC CHECK PLUS utility

When you select BMC CHECK PLUS, the product uses the CHECK PLUS utility and inserts the BMCCHECK keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCK command into the worklist.

The -BMCK command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”

Using the IBM CHECK DATA utility

When you select IBM CHECK, the product uses the CHECK DATA utility and inserts the IBMCHECK keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -CHEK command into the worklist.

The -CHEK command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”
Loading data

When Analysis builds a worklist that requires the loading of data, it uses the ENFORCE option:

- If a table has referential integrity, or if it has both referential integrity and check constraints, the Analysis component uses the ENFORCE NO option during a load. Analysis also generates a -CHEK command in the worklist.
- If a table has only check constraints, the Analysis component uses the ENFORCE CONSTRAINTS option during a load.
- If a table has neither referential integrity nor check constraints, the Analysis component uses the ENFORCE NO option during the load.

For information about load utilities and the ENFORCE option, see the LOADPLUS for DB2 Reference Manual and the IBM documentation.

Using the reorg utilities

Analysis determines whether a reorganization of a table space or index in a worklist is required or optional:

- A reorganization is required under the following circumstances:
  - The changes that you make to the object attributes cause DB2 to place the object in a REORG-pending (REORP) restrictive status (for example, a change to the limit key for a partition).
  - DB2 requires that you reorganize the objects before you can access the data.

When a reorganization is required, the product inserts the REORGPENDONLY keyword into the ALUIN input stream.

- A reorganization is optional or applicable under the following circumstances:
  - The changes that you make to the object attributes cause DB2 to place the object in a REORG-pending (AREO*) advisory status or advisory REORG-pending (AREOR) status (for example, changes to data types or column lengths).
  - DB2 suggests that the objects be reorganized for optimal performance.

When a reorganization is optional, the product inserts the REORGALL keyword into the ALUIN input stream.
You can select when to perform a reorganization and which reorganization utilities to use from the Analysis Reorg Options panel (see Figure 200). You can choose from the following utilities:

- BMC REORG PLUS
- IBM REORG

**Figure 200  Analysis Reorg Options panel**

Using the BMC REORG PLUS utility

When you select **BMC REORG PLUS**, the product uses the REORG PLUS utility and inserts the BMCREORG keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCR command into the worklist.

Using multiple work data sets

You can use multiple SYSUT temporary work data sets to improve I/O performance when more than one nonclustering index for a table exists. Using multiple data sets enables the products to write fewer data blocks and to overlap I/O operations.

You specify the maximum number of SYSUT data sets that you want the REORG PLUS utility to use to build indexes in the Analysis Utility Dataset Options panel (see Figure 196 on page 682). The number of data sets are specified in the MAXSYSUT keyword in the ALUIN input stream.
Using the IBM REORG utility

When you select IBM REORG, the product uses the REORG utility and inserts the IBMREORG keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -REOR command into the worklist.

Generating SHRLEVEL reorgs

In some cases, the only changes that you make in a work ID can be accomplished by using ALTER statements and online reorganizations (for example, online schema changes). In those cases, you can use the utilities to reorganize a table space by performing an online reorg (SHRLEVEL CHANGE).

The utilities cannot perform an online reorganization if any of the following conditions exist:

- An object will be dropped and created in the worklist.
- A VCAT-defined table space or index requires AMS commands.
- LOG NO or NOT LOGGED is specified for a table space.
- A limit key of a table space or index partition was changed.
- An identity column was added to a table.
- A table contains an XML column.
- A check constraint is defined for a table.
- A table space is a large object (LOB) table space, or an index is defined for an auxiliary object.

NOTE

If the number of nonclustering indexes in a table is equivalent to or less than the maximum number of SYSUT data sets, the products create a SYSUT data set for each nonclustering index.

When the work ID is analyzed, the Analysis component specifies the number of SYSUT data sets in the -JCLP command and generates a WORKDDN (SUT) parameter in the -BMCR command in the worklist. In addition, JCL Generation creates SUTnnnnn0 DD statements in the JCL.
Generating SHRLEVEL reorgs

If you select **Generate SHRLEVEL CHANGE (Online) Reorg when applicable**, the product inserts the REORGONLINE keyword into the ALUIN input stream. Whether or not you request an online reorg, you can select a SHRLEVEL REFERENCE or SHRLEVEL NONE reorganization. The product will perform the SHRLEVEL REFERENCE or SHRLEVEL NONE reorganization if an online reorg is not applicable:

- If you select **Generate SHRLEVEL REFERENCE Reorg**, the product inserts the REORGREF keyword.

- If you select **Generate SHRLEVEL NONE Reorg**, the product inserts the REORGNONE keyword.

You can use the IBM REORG utility to reorganize a large object (LOB) table space; in this case, use SHRLEVEL CHANGE to perform an online reorg or use SHRLEVEL REFERENCE. When reorganizing a table space that includes a table with one or more LOB columns, the product can generate the REORG command with the AUX YES parameter on the base table space. The reorganization occurs on the base table space and on all associated LOB table spaces at the same time.

For the objects shown in the shaded rows, **Table 55** summarizes the valid SHRLEVEL options for different versions of DB2.

**Table 55  SHRLEVEL options**

<table>
<thead>
<tr>
<th>DB2 version</th>
<th>Reorg utility</th>
<th>SHRLEVEL options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index and table space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8, 9, 10</td>
<td>IBM and BMC</td>
<td>CHANGE, REFERENCE, and NONE</td>
</tr>
<tr>
<td>LOB index</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8, 9, 10</td>
<td>IBM and BMC</td>
<td>CHANGE and REFERENCE</td>
</tr>
<tr>
<td>LOB table space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>not applicable</td>
<td>No reorganization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The product generates commands to drop and re-create the table space.</td>
</tr>
<tr>
<td>9, 10</td>
<td>IBM and BMC</td>
<td>CHANGE and REFERENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The product can generate the REORG command with the AUX YES parameter on the base table space.</td>
</tr>
</tbody>
</table>
Dynamically allocating reorg data sets

REORG PLUS and REORG can dynamically allocate the data sets that are used by the utilities. On the Analysis Utility Dataset Options panel (see Figure 196), you can select to use the dynamically allocated reorg data sets. When you select to use REORG PLUS or IBM REORG and select the Use Dynamic REORG datasets field, the products use the utilities and insert the DYNREORG keyword into the ALUIN input stream.

Table 56 lists the data sets that the utilities dynamically allocate.

<table>
<thead>
<tr>
<th>Data set</th>
<th>REORG PLUS</th>
<th>REORG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive (ARCHDDN)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Copy (COPYDDN, RECOVERYDDN)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Discard (DISCARDDN)</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Punch (PUNCHDDN)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sysrec (UNLDDN)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sortout or sysut (WORKDDN)</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Consider the following items when you choose to dynamically allocate the reorg data sets:

- The REORG PLUS utility invokes the IBM DSNUTILB utility control program to enable certain features. Because REORG PLUS requires dynamic allocation when it invokes DSNUTILB, ensure that you select Use Dynamic REORG datasets. For information about the features for which REORG PLUS invokes DSNUTILB, see the REORG PLUS for DB2 Reference Manual.

- When you use REORG, you cannot stack the reorg data sets on a tape.

Considerations for using the utilities

Consider the following items when you use the REORG PLUS utility or the IBM REORG utility:

- The products cannot use multiple work data sets for table spaces that are VCAT defined and are reorganized in two phases. Data overlay could occur if the SYSUT data set was reused and another object was loaded or reorged between the two phases of the reorg. The Analysis component generates a WORKDDN (WRK100n).
Considerations for using the utilities

parameter in the -BMCR or the -REOR command in the worklist. JCL Generation creates a WRK100 DD statement in the JCL. To specify information about the WRK100n or SYSUT data sets, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.

- If VCAT-defined table spaces or indexes have changes to the PRIQTY, SECQTY, or VOLUME attributes and a reorganization is necessary, the REORG UNLOAD PAUSE option is inserted before issuing -AMS worklist commands for deleting and defining VSAM data sets. The appropriate reorg worklist command and the REORG RESTART option complete the processing for the reorganization. The command and the option are specified after the ALTER INDEX section of the worklist.

- For table spaces and indexes that are storage-group defined, the reorganization worklist commands are also after the ALTER INDEX section of the worklist.

- If you changed the PRIQTY or SECQTY for a table space or index partition, Analysis generates reorganization commands in the worklist if you have IBM REORG or the BMC REORG PLUS utility designated in your installation options module. Analysis also generates these commands if you override your installation options through the Analysis Options panels.

Changes to the following attributes cause Analysis to place reorganization commands in the worklist:

- PRIQTY
- SECQTY
- PCTFREE
- FREEPAGE
- VOLUME (for VCAT defined partitions)
- changes to clustering index or new cluster index
- STOGROUP (changes when the VCAT of the STOGROUP remains the same)

If a worklist specifies any of the preceding changes to a table space or index but the object is also being dropped and then rebuilt, the table space or index is not reorganized.

- *(DB2 Version 8 only)* If you specify REORGALT, you cannot specify REORGPENDONLY. To reorganize the table space, specify NOREORGALT and REORGPENDONLY, or REORGALT and REORGALL.

- If you specify UTILCOPY, an image copy might be made during a reorganization. However, you cannot use the REORG PLUS utility to dynamically allocate image copy data sets. If you specify to use these utilities to build image copies, Analysis generates a separate COPY command.
If you do not specify the Data option on the Analysis Worklist Command Options panel, the utilities reorganize the table space only when the table space requires a reorganization.

The LOG NO and SORTDATA options are specified with the worklist reorganization commands. However, if LOG NO is specified for a table space, the utilities cannot reorganize the table space by using SHRLEVEL REFERENCE or SHRLEVEL CHANGE; the utilities must use SHRLEVEL NONE.

For more information about these options, see the REORG PLUS for DB2 Reference Manual or the IBM documentation.

When you perform an online REORG, DB2 uses data sets that are designated with “I” and “J” to store the reorganized data structures and data. The products define an I data set when it creates a VCAT-defined table space or index. If the data currently resides on a J data set, the products automatically delete the J data set. However, if an I data set with the same name already exists on your system, the worklist that is created by Analysis fails during Execution.

If you know that you have both I and J data sets before running Analysis, you can prevent the worklist from failing by manually inserting the new AMSDELETEI keyword into the ALUIN input stream. When this keyword is specified, the products insert a DELETE statement for the data set in the worklist. If you do not specify this keyword, or if you specify the new NOAMSDDELETEI keyword, the products issue a warning message and you must manually delete the I data set.

If you change the name of the VCAT-defined data set by changing the high-level qualifier (HLQ), database name, or table space or index name, and you specified the AMSDELETEI keyword, the products do not delete the I data set since no conflict exists.

**NOTE**

Performing an online REORG after Analysis builds the worklist and before Execution runs the worklist can make a worklist invalid. If you use VCAT-defined data sets, run the online REORG before the worklist is built or after it is executed.

### Using the recover and rebuild utilities

You can select which recover and rebuild utilities to use from the Analysis Utility Options panel (see Figure 194 on page 675). You can choose from the following utilities:

- BMC RECOVER PLUS
- IBM REBUILD INDEX
If you choose not to use a recover or rebuild utility (by selecting **No rebuild index**), the product inserts the NOREBUILD keyword into the ALUIN input stream.

### Using the BMC RECOVER PLUS utility

When you select **BMC RECOVER PLUS**, the product uses the RECOVER PLUS utility with the REBUILD command and inserts the BMCREBUILD keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCV command into the worklist.

If you specify to migrate data from a table using the BMC RECOVER PLUS for DB2 utility from image copies option, Analysis uses a stand-alone copy utility if all of the following conditions exist:

- The table space contains more than 254 partitions.
- You did not specify to use dynamic COPY data sets.
- You selected to use other utilities to build image copies.

To ensure that Analysis uses the RECOVER PLUS utility to make the copy of the table space, specify to use dynamic COPY data sets.

### Using the IBM REBUILD INDEX utility

When you select **IBM REBUILD INDEX**, the product uses the IBM REBUILD INDEX utility and inserts the IBMREBUILD keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -RBLD command into the worklist.

### Considerations for using the utilities

Consider the following items when you use the IBM REBUILD INDEX utility or the BMC RECOVER PLUS utility:

- If you are creating an index in the worklist and the parent table is not being created in the worklist, you can defer building the index. The **BMC RECOVER PLUS** or **IBM REBUILD INDEX** utility is run against the index if the index is created with DEFER YES.

- You cannot use RECOVER PLUS to recover objects from an image copy of a simple table space that contains tables that have been dropped and recreated. The image copy contains incorrect object identifiers (OBIDs) for all of the tables, including the tables that were dropped.
Using the copy utilities

With the copy utilities, you can create up to four image copies (primary and backup copies for the local and the remote sites). You can select which copy utilities to use from the Analysis Image Copy Options panel (see Figure 201). You can choose from the following utilities:

- BMC COPY PLUS
- IBM COPY

Figure 201 Analysis Image Copy Options panel

If you choose not to use a copy utility (by selecting No copy), the product inserts the NOCOPY keyword into the ALUIN input stream and leaves table spaces that were loaded in a copy-pending state. If you select No copy start force, the product inserts the NOCOPYFORCE keyword into the ALUIN input stream and forces access to table spaces that are left in a copy-pending state and resets them. When the work ID is analyzed, Analysis inserts the -CMD START DATABASE (databaseName) SPACENAM (databaseName.tableSpaceName) ACCESS(FORCE) command into the worklist.

NOTE

To run a CHANGE MANAGER worklist in parallel, you must use the COPY PLUS utility to create image copies or you must select not to use a copy utility (the No copy or No copy start force option). For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”
Using the BMC COPY PLUS utility

When you select **BMC COPY PLUS**, the product uses the COPY PLUS utility and inserts the BMCCOPY keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCC command into the worklist.

The -BMCC command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”

Using the IBM COPY utility

When you select **IBM COPY**, the product uses the IBM COPY utility and inserts the IBMCOPY keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -COPY command into the worklist.

Copying and registering the data sets

On the Analysis Image Copy Options panel (see Figure 201), you can select options for the primary and backup data sets at the local and remote sites.

The local sites are Copy DDN1 and Copy DDN2, and the remote sites are Recv DDN1 and Recv DDN2. You can select both Copy DDN1 and Copy DDN2 as local copies, and you can select both Recv DDN1 and Recv DDN2 as remote copies. You can also select the following options for the primary and backup data sets at the local and remote sites:

- Selecting **Copy** inserts the COPYDDN (COPY01 COPY02 RECV01 RECV02) keyword into the ALUIN input stream. The product creates and registers a copy of the data set.
  
  — If you specify **IBM COPY**, the copy is automatically registered in the DB2 catalog for recovery.
  
  — If you are creating a stand-alone copy of a DB2 object and you specify **BMC COPY PLUS**, the COPY PLUS REGALL default option determines whether the copy is registered. If REGALL=YES in the COPY PLUS installation options, the copy is registered.
  
  — If you are creating an inline copy of a DB2 object with COPY PLUS and you are using the LOADPLUS, REORG PLUS, or RECOVER PLUS utility, the product inserts the REGISTER (COPY01 COPY02 RECV01 RECV02) keyword into the ALUIN input stream.
Creating image copies

- Selecting Copy no register inserts the COPYDDN (COPY01 COPY02 RECV01 RECV02) keyword into the ALUIN input stream. The product creates a copy of the data set, but does not register it.

**NOTE**

You cannot select this option for IBM COPY or for a stand-alone copy of a DB2 object using the COPY PLUS utility; the copy must be registered.

- Selecting No copy results in a copy of the data set not being made.

## Creating image copies

When you create image copies, you can specify that Analysis performs the following functions:

- builds pre-drop image copy commands before drop and reorg commands in the worklist
- generates load, copy, or reorg commands to create the image copies
- generates partition-level image copy commands

**NOTE**

If an image copy is requested, a worklist will contain image copy commands for the specified indexes in the worklist and for any indexes that are dependents of the specified tables and table spaces, provided that the COPY YES parameter was specified in the CREATE INDEX statement when the index was created.

The CHANGE MANAGER component of the Database Administration solution can build stand-alone image copies of auxiliary objects by using the COPY PLUS or COPY utility. The CHANGE MANAGER product can build stand-alone image copies of base table spaces by using the COPY PLUS or COPY utility.

## Building image copies before DROPs or REORGs

If you select Build an image copy before drops or reorgs, an image copy is created before tables are dropped. Analysis inserts the PIC keyword into the ALUIN input stream.
Consider the following items when you choose to create a pre-drop or pre-reorg image copy:

- If either the NOCOPY or NOCOPYFORCE keywords are in the ALUIN input stream, the IBMCOPY or BMCCOPY keywords are ignored—except if the PIC keyword is also in the ALUIN stream or set in the installation options module.

- The IBM COPY utility is used if the PIC keyword and either the NOCOPY or NOCOPYFORCE keywords are specified in the ALUIN input stream, or if the PIC=Y and either the BMCCOPY=X or BMCCOPY=F keywords are specified in the installation options module.

- If you want to dynamically allocate the image copy data sets that are created before the tables are dropped and either the NOCOPY or NOCOPYFORCE keywords are the in ALUIN input stream, you must manually insert the DYNCOPY keyword into the ALUIN input stream.

**Specifying the utility to create the image copy**

You can select one of the following options to indicate which utility should be the default for building image copies:

- When you select **Use Copy selection specified above to build image copies**, separate copy commands generate all of the copies that are taken by the specific copy utility. The product inserts the NOUTILCOPY keyword into the ALUIN input stream.

- If you select **Other utilities should build image copies when possible**, image copies are created by utilities other than the copy utilities whenever possible. These other utilities include the LOAD, LOADPLUS, RECOVER PLUS, REORG, or REORG PLUS utilities. If the utilities cannot create a copy, a separate copy command is generated. The product inserts the UTILCOPY keyword into the ALUIN input stream.

Consider the following items when you specify the utility to create image copies:

- If you select **BMC LOADPLUS**, the UTILCOPY keyword creates an image copy during the load. If it is a multi-table table space, the image copy is taken when the last table is loaded.

- If you select **IBM LOAD**, the UTILCOPY keyword creates an image copy if no other tables exist in the table space. If it is a multi-table table space, the copy utility takes the copies.

- If you select **BMC REORG PLUS** or **IBM REORG**, the UTILCOPY keyword creates an image copy if the entire table space is reorged and no tables are being loaded. If the table space is partitioned and REORG is being executed on a single partition, the image copies are deferred and are created later by the load or copy utilities.
Creating image copies

NOTE
If the NOCOPY, NOCOPYFORCE, DYNCOPY, or PARALLEL keywords are in the ALUIN input stream, the UTILCOPY and NOUTILCOPY keywords are ignored—except if you have selected the BMC RECOVER PLUS utility.

Creating partition-level image copies

You can specify whether Analysis generates partition-level image copies for any partitioned table spaces or indexes. If you select Make partition level copies using BMC utilities, the products use the COPY PLUS, LOADPLUS, RECOVER PLUS, or REORG PLUS utility to make the copy. The products insert the PARTCOPY keyword into the ALUIN input stream. (The products insert the NOPARTCOPY keyword into the ALUIN input stream if you choose not to make a partition-level copy.)

NOTE
When you use COPY PLUS to create partition-level copies for any partitioned table spaces or indexes, COPY PLUS dynamically allocates the image copy data sets.

For partitioned and nonpartitioned objects, the products use different ddnames for the following copies:

- local primary copy
- local backup copy
- recovery primary copy
- recovery backup copy

Table 57 lists the naming conventions for the primary and backup copies.

Table 57 Primary and backup copy naming conventions for objects

<table>
<thead>
<tr>
<th>Type of copy</th>
<th>Nonpartitioned object ddname</th>
<th>Partitioned object ddname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local primary</td>
<td>SYCLnnnn</td>
<td>Cxxxxyyyy</td>
</tr>
<tr>
<td>Local backup</td>
<td>SYCRnnnn</td>
<td>Dxxxxyyyy</td>
</tr>
<tr>
<td>Recovery primary</td>
<td>RECLnnnn</td>
<td>Exxxxyyyy</td>
</tr>
<tr>
<td>Recovery backup</td>
<td>RECRnnnn</td>
<td>Fxxxxyyyy</td>
</tr>
</tbody>
</table>
In Table 57, \textit{nnnn} represents a sequential number for the table space or index. The variable \textit{xxx} represents a sequential number that pertains to the table space or index that is copied, and \textit{yyyy} represents the partition number for copies made by COPY PLUS.

When you select \textbf{Make partition level copies using BMC utilities}, the products use the following ddnames:

- If the object is partitioned, the products use the partitioned object ddname.
- If the object is not partitioned, the products use the nonpartitioned object name.

\textbf{NOTE}\footnote{If you do not select \textbf{Make partition level copies using BMC utilities}, the products use the nonpartitioned object ddname.}

\section*{Dynamically allocating copy data sets}

COPY PLUS, RECOVER PLUS, and IBM COPY can dynamically allocate the data sets that are used by utilities to create valid image copies. On the Analysis Utility Dataset Options panel (see Figure 196 on page 682), you can select to use the dynamically allocated copy data sets and to override any JCL Generation options that apply to dynamic allocation.

When you select \textbf{Use Dynamic COPY datasets}, the product uses the utility that you selected and inserts the DYNCOPY keyword into the ALUIN input stream. When you select \textbf{Override Defaults for Dynamic Allocation}, the JCL Generation Main Menu is displayed. You can select \textbf{Tape Options} and \textbf{Individual Data Set Options} to override the options that relate to dynamic allocation. For information about setting these options, see the \textit{ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1}.

\textbf{NOTE}\footnote{Before you select \textbf{Override Defaults for Dynamic Allocation}, you must select \textbf{Use Dynamic COPY datasets}.}
Considerations for using dynamic allocation

Consider the following items when you choose to dynamically allocate the copy data sets:

- You cannot use the LOADPLUS and REORG PLUS utilities to dynamically allocate image copy data sets. If you specify to use these utilities to build image copies, Analysis generates a separate COPY command.

- If you want to generate partition-level image copies and dynamically allocate the data sets that are used for those copies, you must append the \$\&PART symbolic variable (where \$a represents an alphabetic character) to the data set prefix for the copies. You can modify the prefix in the JCL Generation Data Set Options for Copies panels.

- When you use COPY PLUS to create partition-level copies for any partitioned table spaces or indexes, COPY PLUS dynamically allocates the image copy data sets.

- If you are rebuilding an index and you want to use RECOVER PLUS to build image copies and to dynamically allocate the image copies, you must select the Use Dynamic COPY datasets option from the Analysis Utility Dataset Options panel and the Other utilities should build image copies when possible option from the Analysis Image Copy Options panel.

- The following JCL Generation options are not applicable:

  - maximum number of cylinders for static data sets

  - parity, data conversion, translation, and compression value for 7-track tape drives

  - primary and secondary space for copy and unload data sets

  - tape stacking (except when using COPY PLUS to copy explicitly created table spaces)

- JCL Generation uses the symbolic variable that is specified in the JCL Generation Data Set Options for Copies panels to create an OUTPUT or a TEMPLATE descriptor for Analysis. Analysis uses this descriptor variable in the data set name.

- When COPY PLUS and RECOVER PLUS dynamically allocate the copy data sets, the utilities use different OUTPUT descriptors in the worklist for local and recovery primary and backup copies. Table 58 lists the OUTPUT descriptors that the products use for dynamic allocation.
Dynamically allocating copy data sets

IBM COPY uses TEMPLATE descriptors and provides templates for a primary set of data set attributes and an alternate set of data set attributes. The utility uses the template with the alternate set of attributes when the size of the data set exceeds a threshold value for the number of units. You can specify the threshold value, the alternate unit name, and alternate SMS classes in the JCL Generation Data Set Options for Copies panels. Table 59 lists the TEMPLATE descriptors that are used for dynamic allocation.

Table 58 Naming conventions for OUTPUT descriptors

<table>
<thead>
<tr>
<th>Type of copy</th>
<th>OUTPUT descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local primary</td>
<td>DCPYLOCP</td>
</tr>
<tr>
<td>Local backup</td>
<td>DCPYLOCB</td>
</tr>
<tr>
<td>Recovery primary</td>
<td>DCPYREMP</td>
</tr>
<tr>
<td>Recovery backup</td>
<td>DCPYREMB</td>
</tr>
<tr>
<td>EXPORT</td>
<td>EXPFILE</td>
</tr>
</tbody>
</table>

Table 59 Naming conventions for TEMPLATE descriptors

<table>
<thead>
<tr>
<th>Type of copy</th>
<th>TEMPLATE descriptor for primary data set</th>
<th>TEMPLATE descriptor for alternate data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local primary</td>
<td>DCPYLOCP</td>
<td>SCPYLOCP</td>
</tr>
<tr>
<td>Local backup</td>
<td>DCPYLOCB</td>
<td>SCPYLOCB</td>
</tr>
<tr>
<td>Recovery primary</td>
<td>DCPYREMP</td>
<td>SCPYREMP</td>
</tr>
<tr>
<td>Recovery backup</td>
<td>DCPYREMB</td>
<td>SCPYREMB</td>
</tr>
</tbody>
</table>

- If you need to start a worklist over, you do not need to delete copy data sets before you start the worklist over if the data sets were dynamically allocated and used GDGs. However, if GDGs were not used for the copy data sets, you must change the name of the dynamically allocated data set in each of the -BMCC worklist commands. Then, you can start the worklist over.

- JCL Generation cannot create the base of the GDG when IBM COPY is used to generate image copies for implicitly created objects.

- To run a CHANGE MANAGER worklist in parallel, you must use dynamically allocated copy data sets. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”

- If you select the No copy or No copy start force Analysis image copy option, dynamically allocated copy data sets cannot be used.

- If the NOCOPY, NOCOPYFORCE, DYNCOPY, or PARALLEL keywords are in the ALUIN input stream, the UTILCOPY and NOUTILCOPY keywords are ignored—except if you have selected the BMC RECOVER PLUS utility.
If statistics are unavailable, or if the table spaces are compressed, modify the values for the PCTPRIM and NBRSECD installation options for the UNLOAD PLUS utility. For more information about dynamically allocated data sets, see the UNLOAD PLUS for DB2 Reference Manual.

You must use dynamically allocated image copy data sets for image copies of partition-by-growth table spaces.

Using the statistics utilities

You can use BMC and IBM utilities to collect and update statistics. You can select your options from the Analysis Statistics Options panel (see Figure 202).

Figure 202  Analysis Statistics Options panel

Collecting statistics

To collect statistics, you can choose from the following utilities:

- BMC BMCSTATS

  The DASD MANAGER PLUS product must be installed to use BMCSTATS.
Collecting statistics

- other BMC utilities (such as BMC REORG PLUS, BMC LOADPLUS, BMC RECOVER PLUS, and BMC COPY PLUS)
- IBM RUNSTATS
- other IBM utilities (such as IBM REORG, IBM LOAD, or IBM REBUILD INDEX)

You also have the option not to collect statistics. When you choose this option, the product inserts the NOSTATS keyword into the ALUIN input stream.

**Using the BMC BMCSTATS utility**

To use the BMCSTATS utility to collect statistics, select *Generate a statistics step in the worklist*. When you select this option, Analysis generates a separate worklist command to collect the statistics. This option is referred to as *stand-alone statistics*.

The product inserts the STANDALONESTATS keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -BMCS command into the worklist. (The -BMCS command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”)

When you use the BMCSTATS utility to collect statistics, you can also select whether to include the TABLE ALL parameter in the worklist command. Choosing Y for the *Include TABLE(ALL) to gather column statistics* option enables the utility to gather column statistics on all of the columns in all of the tables in a table space. The product inserts the TABLEALL keyword into the ALUIN input stream. If you do not choose this option, the product inserts the NOTABLEALL keyword into the ALUIN input stream. When the work ID is analyzed, Analysis includes the TABLE ALL parameter in the worklist command.

**NOTE**

Choosing the *Include TABLE(ALL) to gather column statistics* option can affect execution performance.

**Using the IBM RUNSTATS utility**

To use the IBM RUNSTATS utility to collect statistics, select *Generate a statistics step in the worklist*. When you select this option, Analysis generates a separate worklist command to collect the statistics. This option is referred to as *stand-alone statistics*. 
The product inserts the STANDALONESTATS keyword into the ALUIN input stream. When the work ID is analyzed, Analysis inserts the -RNST command into the worklist. (The -RNST command can be processed in parallel. For more information about the worklist parallelism feature of the Database Administration solution, see Appendix D, “Using worklist parallelism.”)

When you use the RUNSTATS utility to collect statistics, the product provides you with the following options:

- You can choose to enable the utility to gather column statistics on all of the columns in all of the tables in a table space.

  To gather column statistics, choose **Y** for the Include **TABLE(ALL)** to gather column statistics option.

- You can choose to update the DB2 catalog history tables with the current statistics that are being collected.

  If you choose **Y** for the Include IBM RUNSTATS History option, the utility updates the history tables. Analysis inserts the HISTORYALL keyword into the ALUIN input stream. When the work ID is analyzed, Analysis includes the HISTORY ALL parameter in the -RNST command in the worklist.

  If you choose **N** for the option, the utility does not update the history tables. Analysis inserts the NOHISTORY keyword into the ALUIN input stream. When the work ID is analyzed, Analysis includes the HISTORY NONE parameter in the -RNST command in the worklist.

- You can choose to take cardinality statistics for the key columns of an index.

  To take cardinality statistics, choose **Y** for the Include IBM RUNSTATS KEYCARD when gathering Index statistics option. Analysis inserts the KEYCARD keyword into the ALUIN input stream. When analyzing the work ID, Analysis includes the KEYCARD parameter in the -RNST command in the worklist.

  If you choose **N** for the option, the utility does not take cardinality statistics. Analysis inserts the NOKEYCARD keyword into the ALUIN input stream. When analyzing the work ID, Analysis omits the KEYCARD parameter from the -RNST command in the worklist.

### Using other utilities

To use other BMC or IBM utilities to collect statistics, select **Collect statistics with other utilities whenever possible**. When you select this option, Analysis does not generate a separate worklist command to collect statistics. The utility that you use collects the statistics. This option is referred to as **inline statistics**. The product inserts the UTILSTATS keyword into the ALUIN input stream.
Updating statistics

To update statistics, you can choose BMCSTATS or other BMC utilities, or RUNSTATS or other IBM utilities.

Using BMCSTATS or BMC utility statistics

The BMCSTATS and other BMC utilities collect statistics and populate the DASD MANAGER PLUS database, which is maintained in DB2 tables. The utilities can also update statistics in the DB2 catalog.

To update statistics using the BMCSTATS or BMC utilities, you can select one of the following options:

- When you select Use BMCSTATS or BMC utility statistics to update DASD MANAGER PLUS tables only, the worklist updates the statistics in only the DASD MANAGER PLUS tables. The product inserts the BMCSTATSUPD keyword into the ALUIN input stream.

- When you select Use BMCSTATS or BMC utility statistics to update DASD MANAGER PLUS tables and the DB2 Catalog, the worklist updates the statistics in the DASD MANAGER PLUS tables and the DB2 catalog. The product inserts the ALLSTATSUPD keyword into the ALUIN input stream.

**NOTE**
The BMCSTATS utility does not collect statistics on auxiliary table spaces or indexes. If you select Use BMCSTATS or BMC utility statistics to update BMC DASD MANAGER PLUS tables and the DB2 Catalog, Analysis uses the IBM RUNSTATS utility to collect statistics on auxiliary objects and to update the DB2 catalog.

Using IBM RUNSTATS or IBM utility statistics

To update statistics with the IBM RUNSTATS or other IBM utilities, select Use IBM RUNSTATS or IBM utility statistics to update the DB2 Catalog only. The worklist updates the statistics in the DB2 catalog. The product inserts the DB2STATSUPD keyword into the ALUIN input stream.
Using the skeleton library compiler

This appendix presents the following topics:

Overview ................................................................. 719
Compiling SLIBs ......................................................... 720
Changing SLIBs .......................................................... 721
   Testing changes using ISPF file tailoring ....................... 722
   Compiling changed SLIBs ........................................... 723
Processing SLIBs ......................................................... 723
   Generating the SLIB report ......................................... 723

Overview

BMC has improved the performance of JCL construction by using the BMC skeleton library (SLIB) compiler. The SLIB compiler is a tool that is supplied with the Administrative Products for DB2. The use of compiled SLIBs with JCL Generation’s runtime unit, which was designed to handle the compiled SLIBs, can eliminate the ISPF file tailoring process. Figure 203 illustrates the processing flow of the SLIB compiler.
Compiling SLIBs

This appendix describes the compiler, how to test SLIBs before compiling them, and the compiler’s associated runtime unit.

The SLIB compiler is written in REXX and runs as a batch TSO job. Each SLIB is compiled, assembled, and linked into a separately executable load module. The separation of load modules provides maximum flexibility and ease of use because no other dependencies between SLIBs or other object modules exist. You can change one SLIB without having to recompile or relink any other SLIBs or modules.
The SLIB compiler examines each line of the interpretive SLIB language and translates it into assembler source code with commands and instructions that directly interact with the runtime unit. The runtime unit processes the requests and builds the JCL. Like other compilers, the SLIB compiler translates a higher-level language into assembler instructions. Like compilers for C, C++, COBOL, and other languages, the SLIB compiler has its own runtime unit.

**NOTE**
The SLIB compiler does not support all options provided with ISPF file tailoring, but it does support all features that the Administrative products currently use.

The SLIB compiler assumes that the SLIB adheres to standard ISPF file tailoring rules and constructs. Because the compiler does little verification of SLIB syntax, BMC recommends that you verify SLIB changes by using standard ISPF file tailoring before you compile the SLIBs. For more information, see “Testing changes using ISPF file tailoring” on page 722.

## Changing SLIBs

To change an SLIB, code the changes to your SLIB source. Before you compile the SLIB, you should use JCL Generation to test the changes by using ISPF standard file tailoring. Testing the changes ensures that the SLIB is coded correctly and that no ISPF-related errors exist.

You must recompile an SLIB each time that you change its source. Sample JCL for the SLIB compiler is in member AJXCOMPS in the HLQ.DBCNTL data set that BMC supplies at installation. To customize the JCL to your shop’s standards, follow the directions provided in this member.

**WARNING**
BMC uses SMP/E to package and deliver the SLIB members. If you must make a change to an SLIB, consider copying the SLIB member or members to be changed into a separate library. You can then make your changes to the SLIB members in that library without the risk of applying SMP/E maintenance that overlays your changes. Note that PTFs or GA releases that BMC distributes might change the SLIB source. You will need to determine whether the SLIB source was changed. If the SLIB source was changed, you will need to copy the new version of the SLIB source to your separate library, and reapply any changes that you made.
Testing changes using ISPF file tailoring

The runtime unit attempts to process compiled SLIBs first. If it cannot process a compiled SLIB, it reverts to standard ISPF file tailoring. (If standard file tailoring is required, the SLIB that is being processed must exist in the ISPSLIB data set.)

Any of the following criteria force the runtime unit to use standard ISPF file tailoring for an SLIB:

- A compiled SLIB cannot be loaded from either STEPLIB or ISPLLIB.
  
  Remove or rename the compiled version of the SLIB in the STEPLIB or ISPLLIB. Removing the compiled version forces the runtime unit to process the uncompiled SLIB, using standard ISPF file tailoring.

- The first four bytes of the compiled SLIB contain binary zeros.
  
  Insert a new line 1 in the SLIB. Beginning in column 1, enter:

  )CM NO-COMPILE

  Then, compile the SLIB. The compiler recognizes this SLIB as noncompilable and builds a load module with binary zeros in the first four bytes. The binary zeros force the runtime unit to process this SLIB by using standard ISPF file tailoring.

- The ddname, $USESTFT, is allocated as DUMMY to either the TSO session or the batch job that is being executed.
  
  Add the ddname $USESTFT to the JCL stream for batch jobs, or allocate it to your TSO session with the TSO ALLOC command. This option forces all SLIBs to be processed by standard ISPF file tailoring.

  For example, in Batch Execution JCL Generation, add the following JCL:

  //$USESTFT DD DUMMY

  The option remains in effect for the duration of the batch job. In foreground processing, issue the following command before entering the product:

  TSO ALLOC FI($USESTFT) DA('NULLFILE') SHR

  The option remains in effect until you log off or you issue the following command:

  TSO FREE FI($USESTFT)
Compiling changed SLIBs

After you successfully test the SLIB using standard file tailoring, compile the SLIB into your production &HLQ,UDBLINK library. Ensure that you remove the )CM NO-COMPILE option if you used it. Before you test the compiled SLIB, turn off any other options that you used, such as $USESTFT.

BMC strongly recommends that you process all SLIBs as compiled SLIBs, because the runtime performance can be adversely affected by processing noncompiled SLIBs. You can use the report that the runtime unit generates to verify that you are running with compiled SLIBs and that the compile date on the SLIB is what you expect. For more information about the runtime report, see “Generating the SLIB report” on page 723.

Processing SLIBs

The runtime unit replaces the ISPF file tailoring interface. In most cases, interfaces to the runtime unit are the same as interfaces to ISPF file tailoring. To improve runtime performance, the runtime unit offers special logic that is designed specifically for JCL Generation.

The runtime unit handles mixed mode processing of compiled and noncompiled SLIBs. However, mixed mode processing is not recommended because it compromises the improved performance that the runtime unit was designed to provide.

All JCL processing flows through the runtime unit. The runtime unit

- resolves all variables
- provides numeric data padding
- handles I/O
- processes standard file tailoring requests, if necessary
- builds a report about the JCL generation process

Generating the SLIB report

The reporting feature of the runtime unit helps you determine which SLIBs were processed, how they were processed, and when they were last assembled. To use the SLIB reporting feature in Batch Execution JCL Generation, perform the following steps:
1. Add the ddname JGENSRPT to your batch JCL stream in the step that executes AJXBMAIN with a DCB of the following parameters.

   LRECL=80,RECFM=FB, BLKSIZE=6160,DSORG=PS

2. Resubmit your job.

   An example follows:

   ```
   //JGENSRPT DD SYSOUT=*,
   // DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB,DSORG=PS)
   ```

   If you need to produce this report in the foreground, you can use the TSO ALLOC command to allocate the ddname to any data set with RECFM=FB and LRECL=80. An example follows:

   ```
   TSO ALLOC FI(JGENSRPT) DA('dataSet.name') SHR
   ```

   In this example, dataSet.name is an existing sequential data set of RECFM=FB and LRECL=80.

   Figure 204 shows a sample runtime report.

---

**Figure 204  Sample runtime report (part 1 of 2)**

<table>
<thead>
<tr>
<th>Skelname</th>
<th>Usage</th>
<th>Compile Type</th>
<th>Date</th>
<th>Time</th>
<th>Usage Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJX$ACMX</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.07</td>
<td>1</td>
</tr>
<tr>
<td>AJXJOB0</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.42</td>
<td>1</td>
</tr>
<tr>
<td>AJX#USRV</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.39</td>
<td>1</td>
</tr>
<tr>
<td>AJXJOB5</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.37</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEP1</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>09.50</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEP7</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.00</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEPU</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>13.50</td>
<td>1</td>
</tr>
<tr>
<td>AJXSYSX$</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>17.09</td>
<td>1</td>
</tr>
<tr>
<td>AJXSYSMD</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>11.05</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTWKO</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>11.01</td>
<td>1</td>
</tr>
<tr>
<td>AJXSYSTS</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.47</td>
<td>1</td>
</tr>
<tr>
<td>AJXISPFM</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>09.51</td>
<td>1</td>
</tr>
<tr>
<td>AJXCLIBU</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>17.09</td>
<td>1</td>
</tr>
<tr>
<td>AJXMLIBU</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>17.09</td>
<td>2</td>
</tr>
<tr>
<td>AJXISPF$</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.09</td>
<td>1</td>
</tr>
<tr>
<td>AJXSLIBU</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>17.09</td>
<td>1</td>
</tr>
<tr>
<td>AJXTLIBU</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>17.09</td>
<td>1</td>
</tr>
<tr>
<td>AJXPLIBU</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/29/2013</td>
<td>17.09</td>
<td>1</td>
</tr>
<tr>
<td>AJX#PRNT</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>11.13</td>
<td>1</td>
</tr>
<tr>
<td>AJXNOSTS</td>
<td>Usage</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.56</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 204  Sample runtime report (part 2 of 2)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AJXSORT0</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.06</td>
<td>1</td>
</tr>
<tr>
<td>AJXSORT1</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.50</td>
<td>5</td>
</tr>
<tr>
<td>AJX#SORT</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>09.49</td>
<td>5</td>
</tr>
<tr>
<td>AJX#MTAP</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.01</td>
<td>5</td>
</tr>
<tr>
<td>AJXESTIM</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>11.01</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEP9</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>10.50</td>
<td>1</td>
</tr>
<tr>
<td>AJXJOB9</td>
<td>Compiled</td>
<td>01/19/2013</td>
<td>09.55</td>
<td>1</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>40</strong></td>
</tr>
<tr>
<td>Number of FTINCLs</td>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Number of )IMs</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>SLIBs processed</td>
<td></td>
<td></td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Number of JCLRECs</td>
<td></td>
<td></td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>Runtime units lastcc</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Runtime units maxrc</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

The report summary at the end of Figure 204 provides the information shown in Table 60.

Table 60  Runtime report statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of FTINCLs</td>
<td>the number of file tailoring FTINCL requests</td>
</tr>
<tr>
<td>Number of )IMs</td>
<td>the number of imbeds that are encountered when FTINCLs are processed</td>
</tr>
<tr>
<td>SLIBs processed</td>
<td>the number of SLIBs</td>
</tr>
<tr>
<td>Number of JCLRECs</td>
<td>the number of JCL records</td>
</tr>
<tr>
<td>Runtime units lastcc</td>
<td>the last condition code encountered</td>
</tr>
<tr>
<td>Runtime units maxrc</td>
<td>the highest return code encountered</td>
</tr>
</tbody>
</table>
Generating the SLIB report
Using worklist parallelism

This appendix presents the following topics:

Overview of worklist parallelism .................................................. 727
Considerations for enabling worklist parallelism ............................. 729
Controlling XIM execution ............................................................. 733
  Determining the status of XIM ................................................... 734
  Starting XIM ................................................................. 735
  Inactivating XIM initiators .................................................... 736
  Shutting down XIM ............................................................. 736
  Activating XIM initiators ...................................................... 737
  Modifying MVS image variables ............................................. 737
  Troubleshooting the execution of XIM ..................................... 739
Processing a worklist in parallel .................................................. 740

Overview of worklist parallelism

The worklist parallelism feature of the Database Administration for DB2 solution reduces the elapsed time required for executing a worklist that is generated by the BMC CHANGE MANAGER for DB2 component of the solution by executing portions of the worklist concurrently. The CHANGE MANAGER component uses the BMC Cross-System Image Manager (XIM) technology to provide sysplex performance improvements by enabling the distribution and management of discrete units of work (UOW) across one or more IBM OS/390 and z/OS images in a data sharing environment. By doing so, the CHANGE MANAGER component can divide single, long-running tasks into multiple parallel tasks that can be run across multiple computers.

By distributing the work to multiple address spaces, CHANGE MANAGER avoids any memory constraints for a single address space and allocates the work to images that have adequate CPU capacity. As a result, work is distributed dynamically to processors that are under utilized, workloads are balanced, and elapsed time for processing is improved. For example, in a PeopleSoft environment, where thousands
of tables exist, worklist parallelism can be used to unload, load, and copy structures and data to migrate one subsystem to another quickly and efficiently. Tasks can also be run on a single OS/390 or z/OS image to achieve the same results that are realized in a data sharing environment.

Two functions in the CHANGE MANAGER component control the worklist parallelism feature:

- **Analysis**—creates the appropriate worklist commands to run a worklist in parallel
- **Execution**—executes the worklist in parallel

You can include the appropriate worklist parallelism commands in a worklist without actually running the worklist in parallel. The performance of the worklist is the same, regardless of whether the commands are included and the worklist is not run in parallel, or the commands are not included.

*Figure 205* illustrates the parallel processing of a worklist. Each UOW is processed by an XIM initiator on an OS/390 or z/OS image on which XIM is installed. When processing is complete on an XIM initiator, the next UOW is assigned to that initiator. For example, after XIMINIT 2 completes the processing of UOW 2, it begins processing UOW 4.
In a UOW, commands for unload, load, check, copy, and statistics utilities can be run in parallel. Data definition language (DDL) statements are run sequentially.

**Considerations for enabling worklist parallelism**

You should consider the following items before you enable worklist parallelism:

- Start XIM on every OS/390 or z/OS image on which you want to run a worklist in parallel.
- You cannot use stacked tapes.
- Allocate the `/AEXPRT DD` to SYSOUT.
Allocate the /SYSUDUMP DD to a permanent data set.

If the /SYSUDUMP DD is allocated to SYSOUT=*, XIM will not replicate the allocation for each of the initiators. As a result, if an abend occurs, a dump is not created. To ensure that the allocation for each of the initiators is replicated, allocate the /SYSUDUMP DD to a permanent data set.

Ensure that you have enough space to allocate permanent work data sets (such as SYSUT, SUT, SORT, SORTP, and SYSER) for each initiator, in addition to the base data sets. The Execution JCL will include a preallocation step in which permanent work data sets are created for parallel processing. You must specify a DASD unit, not tape, for these work data sets.

More DASD is required when you run a worklist in parallel. To help estimate the amount of DASD, consider the following items:

— Work data sets are sized for the largest object in the scope of the worklist.

— Numerous data sets are allocated for each initiator. Thus, as you increase the number of initiators that you use, you also increase the amount of DASD that is required.

— The DASD must be shared across all of the images on which a parallel worklist is run.

— For new data sets that are dynamically allocated (such as unload, copy, and discard data sets):
  
  Ensure that the statistics information for the DB2 objects is current.

  If statistics are unavailable, or if the table spaces are compressed, modify the values for the PCTPRIM and NBRSECD installation options for the UNLOAD PLUS utility. For more information about dynamically allocated data sets, see the UNLOAD PLUS for DB2 Reference Manual.

  Configure your IBM Storage Management Subsystem (SMS) environment properly so that UNLOAD PLUS can dynamically adjust allocation parameters.

Data set prefixes exist for the following permanent work and image copy data sets:

— baseline recovery
— discard
— local primary and backup copy
— recovery primary and backup copy
— primary SYSREC
These prefixes ensure the uniqueness of the name of the data set when the data set is dynamically allocated. The product options file (POF) that is generated in the HLQ.UDBCNTL data set includes the new prefixes. If you copy an existing POF during installation to the HLQ.UDBCNTL data set, you might need to modify the data set prefixes.

— For data sets that are not dynamically allocated (static), follow the procedures for setting the JCL debugging, display, and execution options in the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1. Choose to include comments that show the statistics that are used to determine the sizes of the data sets in the generated JCL.

- Consider creating a user POF to set the options for processing a worklist in parallel.

- In the Execution JCL for the worklist, you can specify a percentage of the available computing resources or system capacity that should be reserved for running a worklist in parallel.

- Enable the worklist parallelism feature only for worklists that contain a large number of objects or that need to be processed quickly.

You benefit more by running a worklist that contains thousands of objects than you do by running a worklist that contains a single large table space or table. Only one UOW is used for each table space or table. To process a partitioned table space or table more efficiently, the UNLOAD PLUS and LOADPLUS utilities can use separate data sets for each table space partition and can multitask the unloading and loading of data.

- Determine the minimum and maximum number of XIM initiators that you want to use for each job that is to be run in parallel. BMC recommends that you use the default values of 2 as the minimum number and 3 as the maximum number.

The maximum number of initiators influences the number of permanent work data sets that the product allocates in the Execution JCL. If the maximum number of XIM initiators that you specify is greater than the number of objects in the scope of the worklist that you want to run in parallel, your DASD might not be used efficiently.
Considerations for enabling worklist parallelism

- If you use data sharing, consider the following requirements:
  
  — All members in the data sharing group must be at the same version level of DB2.
  
  — The CHANGE MANAGER product libraries must be APF authorized on all images in the data sharing group.
  
  — Cross-System Image Manager (XIM) initiators for any members in the data sharing group must use the same DSNLOAD and DSNEXIT concatenation.
  
  — If you want to use all members in the data sharing environment, invoke CHANGE MANAGER by using the group attachment name for the SSID.
  
- Observe the following restrictions for enabling worklist parallelism in a JES3 environment:
  
  — You must submit the job that executes the worklist on the LPAR on which the JES3 global processor is running.
  
  — The target DB2 subsystem (that is, the subsystem on which you run Analysis in CHANGE MANAGER) must be one of the following items:
    
    ■ a data sharing group name or an active member in a data sharing group
    ■ resident in the same LPAR as the JES3 global processor
  
  — In a DB2 data sharing environment, the following conditions must be met:
    
    ■ A member of the data sharing group must be active on the JES3 global processor.
    
    ■ Before you submit the job for execution, you must start XIM on those LPARs on which active DB2 members reside.

For information about installing and customizing the solution and the worklist parallelism feature, see the Installation System User Guide and the BMC Products and Solutions for DB2 Configuration Guide. For information about restricting access to the feature, see the ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.
Controlling XIM execution

The Database Administration solution uses Cross-System Image Manager (XIM) to manage units of work (UOWs). XIM executes as a separate OS/390 or z/OS started task. You must start XIM on each image where CHANGE MANAGER will use XIM as a distribution point for UOWs. XIM uses the services of the IBM Cross-System Coupling Facility (XCF) to locate and connect to other instances of itself within the OS/390 or z/OS parallel sysplex.

If your site uses the Resource Access Control Facility (RACF) or CA-Top Secret, you can authorize the procedures for the XIM subsystem as started tasks in the started procedures table. If your site uses CA-ACF2, you can authorize the procedures for the XIM subsystem as started tasks under the started task control. Table 61 describes authorization for XIM.

Table 61  Authorizing XIM procedures

<table>
<thead>
<tr>
<th>Product</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACF or CA-Top Secret</td>
<td>Authorize the procedures for the following subsystems as started tasks in the started procedures table:</td>
</tr>
<tr>
<td></td>
<td>▪ XIM performance subsystem</td>
</tr>
<tr>
<td></td>
<td>▪ XIM extended job entry subsystem</td>
</tr>
<tr>
<td></td>
<td>If you are running RACF version 2.1 or later, you can use the STARTED class to add or modify RACF security definitions for started procedures without having to perform an IPL of the system. The STARTED class allows you to modify the security definitions dynamically through the RDEFINE, RALTER, and RLIST commands. For more information about using the STARTED class, see the appropriate RACF publication.</td>
</tr>
<tr>
<td>CA-ACF2</td>
<td>Authorize the procedures for the following subsystems as started tasks under the started task control:</td>
</tr>
<tr>
<td></td>
<td>▪ XIM performance subsystem</td>
</tr>
<tr>
<td></td>
<td>▪ XIM extended job entry subsystem</td>
</tr>
<tr>
<td></td>
<td>For more information, see the appropriate CA-ACF2 publication.</td>
</tr>
</tbody>
</table>
Determining the status of XIM

Your systems programmer can issue XIM console commands from an OS/390 or z/OS console to start, stop, and modify XIM. This section describes the commands to use and the procedures to follow to perform the following tasks:

- determine the status of XIM
- start XIM
- inactivate XIM initiators
- shut down XIM
- activate XIM initiators
- modify MVS image variables
- troubleshoot the execution of XIM

For more information about XIM, see the Cross-System Image Manager (XIM) User Guide.

Determining the status of XIM

You can use the STATUS command to display information about XIM instances in the sysplex or jobs connected to an XIM initiator. To determine whether XIM is running, issue the following command:

/F XIMACM, STATUS

An excerpt from the JES log (see Figure 206) shows the result of issuing the STATUS command where XIM is active.

<table>
<thead>
<tr>
<th>Jobname</th>
<th>Jobid</th>
<th>SmfId</th>
<th>Cvtsname</th>
<th>Status</th>
<th>inits</th>
<th>active</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC01000</td>
<td>DB2A</td>
<td>DB2A</td>
<td>Active</td>
<td>16</td>
<td>0 active</td>
<td></td>
</tr>
<tr>
<td>STC08798</td>
<td>SYSN</td>
<td>SYSN</td>
<td>Active</td>
<td>16</td>
<td>0 active</td>
<td></td>
</tr>
<tr>
<td>STC08638</td>
<td>DB2B</td>
<td>DB2B</td>
<td>Active</td>
<td>16</td>
<td>0 active</td>
<td></td>
</tr>
</tbody>
</table>
Starting XIM

**NOTE**
Before you start XIM, ensure that the STEPLIB library is APF authorized.

Start XIM on each OS/390 or z/OS image that processes work for CHANGE MANAGER. To start XIM, issue the following command:

```
/S XIMACM
```

XIMACM is the name of the started task. The XIMACM procedure is located in a system PROCLIB data set. (When the product was installed, the procedure should have been copied to this data set.)

**NOTE**
You must ensure that the XIM started task name is unique for each version of the solution if both of the following conditions exist:

- You have installed the worklist parallelism feature of the Database Administration solution in one environment (for example, production).
- You later install a new version of the solution in a different environment (for example, test).

In addition, BMC recommends that the XIM started task procedure name that is specified on the Execution Worklist Parallelism Options panel match the name of the started task.

For information about specifying the procedure name, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1.*

Under the following conditions, the Execution function of the solution attempts to start XIM automatically on the image on which Execution is running:

- XIM is not started.
- You attempt to execute a worklist that has worklist parallelism enabled.
- The XIMSTART YES parameter is specified in the AEXPIN input stream in the execution JCL for a worklist.

XIM is not started on any image other than the image on which you submitted a job.
Inactivating XIM initiators

You can use the QUIESCE command to prevent additional work from being accepted. Work that is in progress is allowed to finish. Typically, you would issue this command before shutting down XIM. To inactivate the XIM initiators, issue the following command:

/F XIMACM,QUIESCE

An excerpt from the system log (Figure 207) shows the result of issuing the QUIESCE command.

Figure 207  Result of QUIESCE command

<table>
<thead>
<tr>
<th>BMC951001</th>
<th>XIM QUIESCE Command Accepted, XIM QUIESCE in progress XIMACM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC951000I</td>
<td>XIM STOP Command Accepted, Initiator termination in progress</td>
</tr>
<tr>
<td>BMC985221</td>
<td>Initiator shutdown request received in ASID(01F6). XJS1</td>
</tr>
<tr>
<td>BMC982121</td>
<td>XJS initiator ended in ASID(01F6). XJS1</td>
</tr>
</tbody>
</table>

**NOTE**

The XIM initiators are inactivated only on the image on which you issued the QUIESCE command. If more than one image is participating in a group, issue the QUIESCE command on each image.

Shutting down XIM

You can use the SHUTDOWN command to terminate inactive XIM initiators and XIM. To terminate the XIMACM address space completely, first issue the QUIESCE command and then issue the following command:

/F XIMACM,SHUTDOWN

**NOTE**

If any XIM initiators are active, the SHUTDOWN command fails. You must issue this command on each image.
Activating XIM initiators

You can use the ACTIVATE command to allow initiators to be scheduled again after you issue a QUIESCE command. To restart the XIM initiators, issue the following command:

/F XIMACM,ACTIVATE

Modifying MVS image variables

To modify variables that are specific to an OS/390 or z/OS image, you can modify the member from which active parameters are loaded. Before you can modify the variables, you must determine the location from which the parameters are loaded.

To determine the location from which the parameters are loaded

1. Using your normal method to review SYSOUT, review the active XIMACM started task.

   Alternatively, you can review the XIMACM procedure in your system PROCLIB library.

2. Locate the partitioned data set (PDS) that is allocated to the XIMPARM ddname.

3. On the //EXEC PGM=XIMMAIN statement, locate the PARM option.

   A keyword specifies SUFFIX=xxxx.

4. To determine the member name, append the SUFFIX to XIM.

   For example, if SUFFIX=PARM, the active parameters are loaded from the XIMPARM member, as shown in the following line of JCL:

   //XIMPARM DD DSN=RCDTJP.XIM.CNTL(XIMPARM)

To modify the variables

1. Edit the XIMxxxx member in the data set that is referenced by the //XIMPARM DD statement.

   In the example shown in Figure 208, the name of the member is XIMACMI.
Modify the INITIATORS variable.

In the example shown in Figure 208, the member contains global variables and MVS image variables. The variables in the MVS image variables section override the same variables in the global variable section. For example, the default value for the global number of initiators is 8. However, for the DB2A subsystem ID, the number of initiators is 16.

**NOTE**

Typically, you should not modify other variables unless BMC Customer Support directs you to do so. However, you must ensure that the values for the XIM_GROUP and XCF_GROUP variables are unique for each version if both of the following conditions exist:

- You have installed the worklist parallelism feature of the Database Administration solution in one environment (for example, production).
- You later install a new version of the solution in a different environment (for example, test).

In addition, the XIM group name that is specified on the Execution Worklist Parallelism Options panel must match the name of the group.

For information about specifying the group name, see the *ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1*.

---

**Figure 208  XIMACMI member (part 1 of 2)**

```plaintext
* XIM STARTUP PARM FOR CHANGE MANAGER FOR DB2
* *
* SYNTAX RULES:
* USE COL 1 - 71
* USE ONE PARAMETER PER STATEMENT
* DO NOT CONTINUE A PARM ONTO A SECOND LINE
* ANYTHING FOLLOWING A PARM AND ITS VALUE IS A COMMENT
* THE EQUAL SIGN IS THE REQUIRED DELIMITER
* SPACES TO THE LEFT AND RIGHT OF THE = ARE PERMITTED
* BLANK LINES AND LINES BEGINNING WITH * ARE IGNORED
* ***************************************************************
* GLOBAL VARIABLE SECTION
* XIM_GROUP=XIMACM
* XCF_GROUP=XIMACMCF
* INITIATORS=8
* INIT_PROC=XIMACMI  * PROC FOR TARGET INITIATORS
* *
* RESPONSE TIMEOUT INTERVAL (SECONDS)
* RESPONSE_TIMEOUT=90  * RESPONSE TIME OUT (SECONDS)
* WORKLOAD_REFRESH=1  * WORKLOAD REFRESH INTERVAL (MINUTES)
* ENVIRONMENT_TIMER=60  * ENVIRONMENT TIMER INTERVAL (SECONDS)
* *
* LOCAL MVS IMAGE VARIABLES (COMMENTED TO SHOW AS AN EXAMPLE)
```
3 Save the changes to the member.

4 Inactivate XIM by issuing the QUIESCE command:

/F XIMACM,QUIESCE

5 Verify the status of XIM by issuing the STATUS command:

/F XIMACM,STATUS

6 Shut down XIM by issuing the SHUTDOWN command:

/F XIMACM,SHUTDOWN

7 Start XIM by issuing the start command:

/S XIMACM

The new instance of XIMACM uses the new parameters.

8 To verify the new parameters, issue the STATUS command:

/F XIMACM,STATUS

9 Review the values that XIM displays in the system log.

10 If you use data sharing, repeat step 4 through step 9 for each OS/390 or z/OS image.

Troubleshooting the execution of XIM

If your job could not connect with XIM, use any of the following methods to determine the cause:

- Issue the STATUS command to verify whether XIM was started:

  /F XIMACM,STATUS
If you are using a data sharing environment, ensure that XIM was started on all of the images.

Ensure that the STEPLIB library was APF authorized.

Review the output from the XIMACM started task procedure.

Review the XIM job or the system log for error messages that were issued by the XIM started task or by the CHANGE MANAGER batch job. Using your job name, search the log for enqueue-type messages for the IBM Global Resource Serialization (GRS) or Unicenter CA-MIM products. If you are using a data sharing environment with multiple OS/390 or z/OS images and you previously canceled a parallel job, an initiator might still be running and holding data sets. This initiator might be preventing another initiator from starting.

If necessary, specify the TRACE YES keyword in the AEXPIN input stream and run the job again. For more information, see the *ALTER and CHANGE MANAGER for DB2 Reference Manual*.

**Processing a worklist in parallel**

By default, a worklist is not processed in parallel. The following conditions must exist for CHANGE MANAGER to execute the worklist concurrently:

- The BMC UNLOAD PLUS and LOADPLUS utilities are used. (That is, the BMCUNLOAD and BMCLoad keywords are specified in the ALUIN input stream.)

- Either the BMC COPY PLUS utility is used to create image copies or no copy utility is used. (That is, the BMCCOPY, NOCOPY, or NOCOPYFORCE keyword is specified in the ALUIN input stream.)

Other utilities cannot be used to create image copies. If the UTILCOPY keyword is in the ALUIN input stream, it is ignored.

- Dynamically allocated SYSRE, SYSDISC, and COPY data sets are used. (That is, the DYNCOPY and DYNUNLDS parameters in the installation options module are set to Y, or the DYNCOPY and DYNUNLDS keywords are specified in the ALUIN input stream.)
Commands to unload and load data are included. (That is, the DATA parameter is specified with the INCLUDE keyword in the ALUIN input stream.) Thus, CHANGE MANAGER will not run the worklist in parallel if any of the following conditions exist:

- The BMC REORG PLUS for DB2 utility is used.
- The IBM LOAD, REORG, or COPY utility is used.
- BMC UNLOAD PLUS from image copies or RECOVER PLUS from image copies is selected as the method to unload data.
- Commands to unload and load data are excluded.

Worklist commands for the following utilities can be processed in parallel:

- BMC BMCSTATS utility (-BMCS)
- BMC CHECK PLUS utility (-BMCK)
- BMC COPY PLUS utility (-BMCC)
- BMC LOADPLUS utility (-BMCL)
- BMC UNLOAD PLUS utility (-BMCD)
- IBM CHECK DATA utility (-CHEK)
- IBM RUNSTATS utility (-RNST)

In addition, worklist commands for the LOB DATA MOVER program (-UNLB and -LOLB) can be processed in parallel. However, because the LOB DATA MOVER loads LOB data into the base table by using SQL UPDATE statements, the base table must be locked during an update. As a result, multiple LOB columns in the same base table cannot be loaded at the same time. Instead, the worklist is processed in parallel for each base table (not for each LOB column). For more information about the LOB DATA MOVER program, see “Unloading data in XML and LOB columns” on page 678.

To enable the worklist parallelism feature, you must specify options in the Analysis and Execution functions of CHANGE MANAGER. The Analysis override options (as well as the PARALLEL installation options parameter) create the appropriate worklist commands to run a worklist in parallel. The Execution override options actually enable the worklist to run in parallel.
For more information about using the worklist parallelism feature, see the topics listed in Table 62.

### Table 62  Worklist parallelism topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifying prefixes for permanent data sets</td>
<td><em>ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1</em></td>
</tr>
<tr>
<td>Specifying Analysis options</td>
<td>“Overriding the default processing options” on page 61</td>
</tr>
<tr>
<td>Reviewing parallelism worklist commands</td>
<td><em>ALTER and CHANGE MANAGER for DB2 Reference Manual</em></td>
</tr>
<tr>
<td>Specifying Execution options</td>
<td><em>ALTER and CHANGE MANAGER for DB2 User Guide, Volume 1</em></td>
</tr>
<tr>
<td>Reviewing Execution JCL</td>
<td>“Execution JCL” on page 91</td>
</tr>
<tr>
<td>Reviewing the output from a parallel worklist</td>
<td>“Output for parallel worklists” on page 103</td>
</tr>
<tr>
<td>Restarting a worklist</td>
<td>“Restart methods” on page 106</td>
</tr>
</tbody>
</table>
Glossary

A

ACM
The product code that BMC uses to identify the CHANGE MANAGER for DB2 product.

ACT
The product code that BMC uses to identify the CATALOG MANAGER for DB2 product.

action code
A one-character or two-character command that you can enter on one of the lines of a list panel. Although you can only enter one command per line, you can enter multiple commands on a single panel.

Administrative Products for DB2
A collection of products from BMC that includes ALTER, CATALOG MANAGER, CHANGE MANAGER, and DASD MANAGER PLUS. These integrated products are designed to help database administrators, system programmers, and application developers automate the tasks associated with the implementation and administration of a DB2 Universal Database for z/OS system.

When all Administrative products for DB2 are installed properly, they can access some of each other’s functionality. In addition, these products can access the IBM and BMC utilities.

AEXIN
The ddname of the input stream that the Execution component uses.

AEXPIN
The ddname of the input stream that the Execution component uses to control parallelism processing.

AEXPRINT
The ddname of diagnostic output of the Execution component. This diagnostic output data set contains all output from the Execution process, including DB2 for OS/390 utility messages, BMC utility messages, dynamic SQL messages, IDCAMS messages, and any other messages that are generated by the actions of the worklist. AEXPRINT is frequently referred to as the worklist execution log.
AEXPRnnn
The ddname of one of the initiator outputs of the Execution component. When the worklist parallelism feature of the BMC Database Administration for DB2 solution is used, this output data set contains AEXPRINT output from each BMC Cross-System Image Manager (XIM) initiator.

AEXPTRAC
The ddname of the trace output of the Execution component. When the worklist parallelism feature of the BMC Database Administration for DB2 solution is used, this output data set contains tracing records.

AEXSYnnn
The ddname of one of the initiator outputs of the Execution component. When the worklist parallelism feature of the BMC Database Administration for DB2 solution is used, this output data set contains system messages and job information for each BMC Cross-System Image Manager (XIM) initiator.

AJX variables
A group of variables (JOB, STEP, and DD) that are used during Job Control Language (JCL) generation. JOB global variables are set once per JCL creation session. STEP global variables are set at the beginning of a JCL creation session and are updated when a new JOBSTEP is detected. Data Definitions (DD) variables are local variables that are set for each creation of a JCL DD entry. Descriptions of these variables are provided in HLQ.DBSLIB($AJXDOCV).

AJXIN
The ddname of the input stream that the Batch Execution JCL Generation component uses.

AJXPOFIN
The ddname of the override input stream that the Batch Execution JCL Generation component uses.

AJXPOFVL
The ddname of the product options file (POF) validation report that the Batch Execution JCL Generation component uses.

AJXPRINT
The ddname of the diagnostic output of the Batch Execution JCL Generation component.

alloc unit
The allocation unit that is used for space estimation calculations, volume placement, and primary and secondary quantities. Possible values are K (kilobytes), T (tracks), or C (cylinders). The default value comes from the installation options modules.
ALTER for DB2
A BMC product that provides advanced database administration and manipulation within a single DB2 subsystem. ALTER streamlines the process of changing and migrating database objects, handles analysis for both changes and migrations, and automatically generates SQL, DB2 commands, and utilities.

alter-type work ID
A work ID that ALTER and CHANGE MANAGER use to perform modifications to the local DB2 subsystem.

ALU
The product code that BMC uses to identify the ALTER for DB2 product.

ALUIN
The ddname of the input stream that the Import, Baseline, Baseline Report, Compare, Analysis, and CM/PILOT components use.

ALUPRINT
The ddname of the diagnostic output of the Import, Baseline, Baseline Report, Compare, and Analysis components.

Analysis
A component of ALTER and CHANGE MANAGER that reads the change and migration requests for a particular work ID and generates a worklist to implement the requests. Analysis checks for consistency with the DB2 catalog before it builds the worklist.

Application
In the CM/PILOT component, the association of a group of CHANGE MANAGER profiles that are used repeatedly for the change management tasks of a specific DB2 application.

ASU
The product code that BMC uses to identify the DASD MANAGER PLUS for DB2 product.

attribute
A value that defines certain properties of an object. Each attribute of an object can occur only once and has a single value from a finite list of possible values. For example, some of the attributes of the TABLE object are database, table space, and EDITPROC. Attributes differ from sub-elements because sub-elements can occur multiple times and can have attributes of their own. For example, columns are sub-elements of tables.

AUTHID
See authorization ID.

authorization ID
An identifier that is allowed a set of privileges. An example of the authorization ID is the owner of a table space, database, storage group, or synonym. An authorization ID is the implicit qualifier of a table, view, alias, or index name.
auxiliary index
An index on an auxiliary table. Each index refers to a large object (LOB) column.

auxiliary list
A list of several auxiliary objects (such as auxiliary table spaces, tables, or indexes) of only one type.

auxiliary table
A table that contains a single large object (LOB) column. An auxiliary table resides in an auxiliary (or LOB) table space.

auxiliary table space
A nonpartitioned table space that contains the data for a large object (LOB) column in an auxiliary table.

B

base table
A table that contains a ROWID column and the definition for a large object (LOB) column. A base table is incomplete if it does not contain any auxiliary objects, such as auxiliary table spaces, tables, and indexes.

base table space
A table space that contains base tables.

baseline
A set of data structures that are captured at a specific point in time.

Baseline
A component of CHANGE MANAGER that captures a set of DB2 structure definitions from either the DB2 catalog or a DDL file at a specific point in time.

baseline name template
A template that a baseline profile can contain. This template is used to create the names of the baselines that are established with the profile. The template can include arbitrary text and the special character sequences #### and @@@@@@@ or @@@@@@@@@, or a combination of both types of sequences. When the baseline is created, ascending numbers replace the #### sequence and the current date replaces the @@@@@@ or @@@@@@@@ sequence. For @@@@@@, the date is in the form YYMMDD. For @@@@@@@@@, the date is in the form YYYYMMDD.

baseline profile
A BMC object that contains the information that is necessary to establish a baseline. The two types of baseline profiles are catalog and DDL.

batch component
A component that can run in batch mode. Batch components include Analysis, Baseline, Baseline Report, Compare, Execution, and Import.
BMC object
A logical entity that contains data that is necessary for performing tasks. BMC objects have a two-part name \((a,b)\). The BMC objects include work IDs, worklists, unload data sets, baselines, profiles, CDL files, DDL files, internal tables, task IDs, CM/PILOT worklists, applications, and script tables.

BMCCOPY
The BMC COPY PLUS utility that is used to create an image copy. The short form of the command is BMCI.

BMCLOAD
The BMC LOADPLUS utility that is used to load DB2 tables. The short form of the command is BMCL.

BMCREORG
The BMC REORG PLUS utility that is used to perform a reorganization. The short form of the command is BMCR.

BMCSTATS
A command that invokes the statistics collecting function of the DASD MANAGER PLUS product. The short form of the command is BMCS. BMCSTATS is similar to the IBM RUNSTATS utility. It provides the same statistics plus many additional statistics.

BMCUNLOAD
The BMC UNLOAD PLUS utility that is used to unload data from a full image copy of one or more tables in a table space. The short form of the command is BMCD.

C
catalog baseline
A baseline that is established on part of the DB2 catalog. A catalog baseline must include a scope, which might be defined directly in a baseline profile or by reference to a scope in a migrate profile.

catalog indirection
An optional method of implementing the BMC Administrative products for DB2 that allows them to access the DB2 catalog indirectly when making information queries. Catalog indirection is accomplished by using synonyms that point either to a copy of the DB2 catalog or to user-created views of the catalog. Benefits include reducing catalog contention and providing an additional level of security for sensitive catalog data.

CATALOG MANAGER for DB2
A tool that is designed to automate the day-to-day tasks associated with administering DB2. This product provides an interactive, intuitive, and easy-to-use interface for submitting DB2 commands and retrieving catalog information using qualified lists, wildcard searches, and dependency lists. CATALOG MANAGER provides the ability to create and drop DB2 objects, re-create dropped structures and data, and browse and edit table data.
CDL
    See Change Definition Language.

Change Definition Language (CDL)
    A BMC proprietary language that is used to specify changes to DB2 data structures.

CHANGE MANAGER for DB2
    A BMC product that enables database administrators, system administrators, and developers to manage user applications and individual database objects globally. As a robust extension of the ALTER product, CHANGE MANAGER automates data structure changes across multiple DB2 subsystems and between DB2 and data modeling tools by providing a way to implement, migrate, synchronize, and back out data structure changes while preserving structure modifications that might be unique to a specific subsystem.

change rule
    An element of a migrate-type work ID or a migrate profile. Change rules define the automatic object attribute changes that are made during a migration or change migration process. These rules can include changes, or sub-element inclusion or exclusion. Change rules can also force VCAT definitions for table spaces and indexes.

checkpoint
    A point at which information about the status of a job and the system can be recorded so that the job step can later be restarted.

    See also sync point.

child
    A DB2 object that contains the foreign keys which reference the primary key in a parent table.

    See also parent.

CLIST
    Command list.

CM/PILOT for DB2
    A component of CHANGE MANAGER that automates the DB2 change management processes that you perform using CHANGE MANAGER.

CM/PILOT worklist
    A data set that contains the ordered commands, keywords, and parameters that CHANGE MANAGER needs to process a task ID.

command
    A token that you can enter at the command prompt on a panel.

    See also action code.
commit
An operation that terminates a unit of work. A commit releases all locks. Data that was changed is now consistent.

Compare
A component of CHANGE MANAGER that identifies the differences between two sets of data structures and then generates a CDL file. You can review the differences and decide how to apply the differences to implement the necessary changes. You can compare data structures that are stored in a DDL file, baseline, worklist, or DB2 catalog.

Compare1
The primary input to the compare process.

See also source.

Compare2
The secondary input to the compare process.

See also target.

component
A major functional unit of ALTER or CHANGE MANAGER, such as Analysis, Execution, Specification, or Import.

costant
See referential constraint.

Cross-System Image Manager (XIM)
A BMC technology that provides sysplex performance improvements by enabling the distribution and management of discrete units of work (UOW) across one or more OS/390 and z/OS systems. The BMC products that use XIM can divide single, long-running tasks into multiple parallel tasks that can be run across multiple computers in the sysplex, thus decreasing the overall elapsed time. The products can also be used with XIM in a data sharing environment on a single OS/390 or z/OS image.

D

DASD MANAGER PLUS for DB2
A BMC product that automates utility generation, gathers comprehensive statistics, monitors changes in the database, and enables you to perform maintenance based on the condition of the data instead of a rigid schedule.

Data Control Language (DCL)
A category of SQL statements that control data security.

data definition language (DDL)
A category of SQL statements that create, modify, or delete database objects.
data definition name (ddname)  
The name of a data definition (DD) statement in job control language (JCL) that corresponds to a data control block that contains the same name.

Data Manipulation Language (DML)  
In the CM/PILOT component of CHANGE MANAGER, SQL-like statements that can be used to update, delete, and migrate data structures.

data set sizing  
The process of determining data set allocations, especially as used by the JCL Generation component. Data set sizing is distinct from space estimation.

data structure  
An object that is defined in the DB2 catalog. Objects include storage groups, databases, table spaces, tables, indexes, foreign keys, views, synonyms, aliases, and triggers.

database administrator (DBA)  
An individual who is responsible for the design, development, operation, security, maintenance, and use of databases.

database request module (DBRM)  
A module that contains SQL statements which the DB2 precompiler has extracted from a source program.

DBA  
See database administrator.

DB2 catalog  
System tables, maintained and used by DB2, that contain descriptions of DB2 objects such as tables, views, and indexes.

DB2 command  
An instruction to the DB2 subsystem. Some example command processes enable you to start or stop DB2, display information on current users, start or stop databases, and display information about databases. DB2 commands always begin with a hyphen (–).

DBCS  
See double-byte character set.

DD statement  
Data Definition statement.

DDL  
See data definition language.

DDL baseline  
A baseline that is established on a file that contains DDL.
**ddname**

See data definition name.

**default options module**

See installation options module.

**default value**

A predetermined value, attribute, or option that is assumed when no other is explicitly specified.

**delimited identifier**

An SQL identifier that is enclosed within escape characters.

**dependencies**

The name or values of objects which another object uses as part of its definition or as a hierarchical subordinate.

**dependent**

A child object (row or table) that has at least one parent.

**dependent object**

An object whose definition relies on the name or the values of another object. The dependent object references the other object.

**destination**

The intended receiving location for CDL or a worklist.

**device type**

The type of disk device used for DB2 data set allocation, such as 3380 and 3390, or generic types TAPE and CART.

**double-byte character set (DBCS)**

A delimited set of characters in which each character is represented by two bytes. Katakana and other lowercase characters are nonstandard characters and must be contained within double quotes.

**DOPTS**

See installation options module.

**duplicate**

An action that you can make on a database object. You can duplicate single or multiple objects within the same database system. If you want to create an object that is like an existing object, you can use the Create Like command to duplicate the existing object. You can then change the object name and make any other necessary changes.
E

effect procedure
   See EDITPROC.

EDITPROC
   An edit procedure that defines an editing routine to be invoked just after a record that
   corresponds to a table row is retrieved and just before that record is stored. Editing routines
   allow for data compression, decompression, and encrypting.

END
   An ISPF command, similar to Enter, that validates and processes the information on a panel and
   returns you to the previous panel, but does not execute commands. This command is typically
   programmed on your keyboard as a function key, such as PF3.

Enter key
   The key that executes any commands that have been specified. For a sequence of panels, the
   Enter key displays the next panel.

Execution
   A component of ALTER and CHANGE MANAGER that carries out the commands in a
   worklist.

exit routine
   A program (BMC, IBM, or user-written) that receives control from DB2 to perform specific
   functions. Exit routines run as extensions of DB2 (for example, authorization checking).

F

field procedure
   See FIELDPROC.

FIELDPROC
   A user-written exit routine that is designed to receive a single value and transform (encode or
   decode) it in any way that the user specifies.

foreground component
   A component that can run in foreground mode. Foreground components include Front End,
   Specification, Analysis, Compare, JCL Generation, and Import.

Front End
   A component of ALTER and CHANGE MANAGER that acts as the interface between the user
   and the other components. Front End is an interactive ISPF dialog that is responsible for
   creating and maintaining BMC objects and facilitating the generation of JCL.

full-recovery baseline
   A baseline that captures data and the data structure definitions at a specific point in time.
GDG

See generation data group.

generation data group (GDG)
A finite number of data sets that are kept in chronological order. Each data set is a generation data set.

group ID
In an outbound migrate profile, a four-character identifier that links locations (or application instances) together. If a migrate profile defines one or more Group IDs, Analysis and Compare generate a single output file (worklist or CDL) for each group. If groups are not defined, Analysis and Compare generate one output file for each location. Locations within a group must reside on the same physical DB2 subsystem.

hash value
A number that appears at the end of commands in worklists or CDL files. The hash value is generated based on the contents of the command line and allows the products to determine whether the line has been manually modified since the file was generated.

HLQ
High-level qualifier of a data set.

image copy
An exact reproduction of all or part of the data in a table space. IBM provides utility programs to make full image copies (copy the entire table space) or incremental image copies (copy only the pages that have been modified since the last image copy). The BMC COPY PLUS utility can perform the same function. You can make an image copy of an index.

import
The process of obtaining an object or an object-set definition from a file or an external database and applying it to an alter-type work ID.

Import
A component of CHANGE MANAGER that converts statements that are stored in a CDL or DDL file into change requests in a work ID. In ALTER, Import converts statements that are stored in a DDL file.

inbound migrate profile
A profile that can be used with the Import component to automate changes to object attributes. An inbound migrate profile can contain only change rules (no locations or scope).
incremental DDL
   The DDL that changes the data structures that exist in the DB2 catalog, by using either an alter strategy or a drop-then-rebuild strategy.

installation options module
   An assembler module that contains keywords whose global values determine the operating environment for a BMC product.

Interactive System Productivity Facility (ISPF)
   An IBM product that provides a full-screen editor and dialog manager.

internal table
   A table that ALTER or CHANGE MANAGER use to store information.

ISPF
   See Interactive System Productivity Facility.

ISPF skeletons
   Data definition statement templates that JCL Generation uses. The skeletons are described in HLQ.DBSLIB($AJXDOC).

J

JCL
   Job Control Language that is used to execute processes in batch mode.

JCL DSN
   The name of the data set that contains job control language (JCL). The data set must exist and can be partitioned or sequential. You must specify a member name for partitioned data sets. You can use symbolic variables.

JCL Generation (JCLGEN)
   A component of ALTER and CHANGE MANAGER that constructs a job control language (JCL) file for running the components in batch mode. When you choose to build JCL, JCLGEN is passed to the worklist that contains the control statements. ALTER and CHANGE MANAGER resolve all data set names that are entered with symbolic variables on the interface panels. JCLGEN resolves all data sets that are passed from the option panels and the unload data sets that are used by the Execution facility. The generated JCL includes data definition statements (ddnames) for all data sets that are needed by Execution, as well as the EXEC statement for the program and any necessary control parameters.

JCL variable display
   The resulting output of a user option that includes debugging comments within any generated JCL. All AJX-prefixed variables are displayed as //* comments in the JCL to assist in diagnosing JCL Generation problems.
JCLGEN
   See JCL Generation.

job
   A batch unit of work that is defined by JCL, a work ID, and a worklist to perform tasks. ALTER
   and CHANGE MANAGER use the information that you supply for a work ID to generate the
   worklist, which provides the BMC JCL Generation component (JCLGEN) with the information
   that is necessary to build the JCL to run a job.

L

large object (LOB) column
   A type of column that is used to store large objects (LOBs), such as images, audio, video, text, or
   graphics, as strings. The data type of the column is defined as LOB (such as a binary large object,
   or BLOB; character large object, or CLOB; or double-byte character large object, or DBCLOB) or
   as a distinct or user-defined type (UDT) that is based on a LOB data type. A LOB column resides
   in an auxiliary table.

large object (LOB) table space
   See auxiliary table space.

LINK library
   A partitioned, cataloged data set that is used to store and retrieve all or part of a program in a
   form that is suitable for loading into main storage for execution. The LINK library contains
   executable modules that perform a product’s processes.

LOAD
   A command that invokes the IBM LOAD utility. The short form of the command is LOAD. The
   utility loads data from a file to a table space.

location
   An arbitrary identifier that groups change rules for a particular destination. An outbound
   migrate profile can define one or more locations. A location is not a DB2 location identifier.

   See also application instance.

LOB DATA MOVER
   In the Database Administration solution, a program that is used to unload and load data that is
   contained in large object (LOB) columns.

M

menu
   A list of action options. You select an action by typing its corresponding number in the option
   input field and pressing Enter. A menu panel might contain other fields that you can use to
   qualify the action.
**migrate**

The process of moving DB2 data structures, data structures and data, or data structure changes, from an origin to a destination subsystem.

**migrate profile**

A profile that can be reused to select a set of objects, to customize changes to objects which are migrated to different locations or used in a comparison, or both.

**migrate-type work ID**

A work ID that is used to migrate data structures, data, or both. Worklists that are generated from a migrate-type work ID do not contain SQL DROP statements. These work IDs can have migrate options and change rules.

**mixed list**

A panel in the Specification component that can display multiple DB2 object types.

**N**

**name propagation**

The process of extending to dependent objects the changes that you make in a referenced object. For example, if you change the name of a table that is referenced in a view, name propagation replicates the new name in the view definition.

**name template**

See baseline name template and work ID name template.

**null**

A special value that indicates the absence of information.

**O**

**object**

A DB2 entity that can be created or dropped. Objects are storage groups, databases, table spaces, tables, indexes, foreign keys, views, synonyms, aliases, or triggers. Unlike DB2, which treats foreign keys and check constraints as table attributes, ALTER and CHANGE MANAGER treat foreign keys and check constraints as independent objects.

**object list**

A list of one or more objects of different types.

**option**

A named value that is used to control one or more components. Global options are defined in the installation options module. The user can override the installation options by specifying the user options or by specifying keywords in the component’s ALUIN, AEXIN, or AJXIN input stream.
origin
   The sending location or source of a migration or a change migration.

orphaned auxiliary index
   An auxiliary index that is not associated with an existing auxiliary table.

orphaned auxiliary table space
   An auxiliary table space that does not contain an auxiliary table.

outbound migrate profile
   A profile that is used in performing a change migration. An outbound migrate profile might
define one or more locations that contain change rules that are defined for any or all of those
locations. Optionally, an outbound migrate profile can have a scope to select the DB2 objects on
which to operate.

P

page
   A unit of storage within a table space (4K or 32K) or index space (4K). A page in a table space
contains one or more rows of a table. 8K and 16K pages can be used.

parent
   A DB2 object that contains the primary key which might be referenced by one or more foreign
keys in the child table.

   See also child.

partitioned data set (PDS)
   A data set in direct access storage that is divided into partitions, called members, each of which
can contain a program, part of a program, or data. Synonymous with program library.

partitioned table space
   A table space that is subdivided into parts (based upon index key range), each of which can be
processed independently by utilities.

pattern
   A rule that is applied to the naming of objects of a specified type. You can use wildcard
characters (% and *) when you define a pattern.

PDS
   See partitioned data set.

piecesize
   For a nonpartitioned Type 2 index, the maximum size of a data set storage piece in kilobytes.
   Valid sizes are 0, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288,
   1048576, 2097152, and 4194304.
privilege
The capability of performing a specific function (authorization) on an object. Privileges might be explicitly or implicitly granted.

profile
A collection of scope rules, change rules, and locations that enables you to define and control a change, migrate, or baseline process.

See also baseline profile, migrate profile, inbound migrate profile, and outbound migrate profile.

protected baseline
A baseline that is designated as protected from deletion. A protected baseline cannot be deleted until the protected designation is removed.

See also unprotected baseline.

R
receive-type work ID
A work ID that is used to create new data structures and load migrated data on a different subsystem.

recovery
The process of restoring a set of data structure definitions to their state at a particular point in time. Recovery involves comparing the DB2 catalog to a baseline, importing the CDL, generating a worklist with Analysis, and executing the worklist. If the baseline is a full-recovery baseline, you can recover data and the data structure definitions.

reference location
A location in a profile whose rules are used by another location. For example, a location called Houston might have that rules that are explicitly defined, while locations called Austin and Dallas might use Houston as a reference location.

reference profile
A profile whose scope is used by another profile. A baseline profile can reference a migrate profile, and a migrate profile can reference a baseline profile. Using a reference profile enables you to define the scope only once and thus eliminates errors that might arise from redesigning the scope.

referenced object
An object that a dependent object references. If you change the definition of a referenced object, dependent objects might not continue to function properly.

referential constraint
The requirement that nonnull values of a designated foreign key are valid only if they equal values of the primary key of a designated table. The relationship between the primary key in the parent table and a foreign key in a dependent table is used to establish referential integrity in a
database. A referential constraint is always assigned a name to distinguish it from other constraints.

**referential integrity**
The condition that exists when all intended references from data in one column of a table to data in another column of the same or a different table are valid. Maintaining referential integrity requires enforcing referential constraints on all LOAD, RECOVER, INSERT, UPDATE, and DELETE operations.

**REORG**
A command that invokes the IBM REORG utility. The short form of the command is REOR.

**S**

**SBCS**
See single-byte character set.

**scope**
The final set of DB2 objects that are selected from the catalog on which the product operates. A scope includes objects that are explicitly selected and any of their object dependencies. For the Baseline component, the scope selects the objects that are captured in the baseline. For the Compare and Analysis components, the scope selects the objects that are included in the comparison or analysis process. A scope consists of one or more scope rules.

**scope rule**
A specification for selecting DB2 objects from the catalog by object type and name.

**Script table**
In the CM/PILOT component of CHANGE MANAGER, ordered steps that prompt you for the information that is required to perform a change management task.

**selection list**
A list of related items from which you can select one for further action. The actions (line commands) that you can specify in the Act field are typically displayed across the top of the panel.

**single-byte character set (SBCS)**
A character set in which each character is represented by a one-byte code.

**source**
In CHANGE MANAGER, the original object of a migration or the original object of a Compare process. When migrating objects or databases, the source database is the database from which you are migrating. The Compare process compares a source data structure with a target data structure. This process synchronizes two data structures, and the source is the data structure that needs to change.

See also Compare1.
Space Estimation
A feature of ALTER and CHANGE MANAGER that enables you to determine the amount of space that a table space or index will require, based on the object definitions and their estimated usages.

Specification
A component of ALTER and CHANGE MANAGER that enables you to create or edit data structure change or migration requests. Specification stores its change or migration requests in a work ID.

SQL
See Structured Query Language.

SQLID
The authorization ID that is used as the implicit qualifier of table, view, synonym, and index names in dynamic SQL statements. The SQL ID, along with the other authorization IDs of a process, is used for authorization checking of dynamic SQL statements.

SSID
A DB2 subsystem identifier.

sequence number
A six-digit, zero-filled number that identifies a statement in a worklist. The sequence number appears in columns 7 through 12 of the first line of each worklist command.

Structured Query Language (SQL)
An ANSI-standard language for database definition, manipulation, and query.

structure-only baseline
A baseline that contains only data structure definitions. No data from those data structures are included.

sub-element
A component of an object. For example, a column is a sub-element of a table, and a volume is a sub-element of a storage group.

subtype
See data subtype.

symbolic variable
A user interface variable that has its value set interactively for the current user and session at the time of job control language (JCL) generation. The BMC JCL Generation component (JCLGEN) uses symbolic variables to perform ISPF file tailoring services. A symbolic variable should be preceded with an ampersand (&). In the installation options modules, a symbolic variable
should be preceded with two ampersands (&&). Symbolic variables should not be confused with global job variables (AJX-type), which have their values set for all users and all sessions.

See also JCL Generation.

SYNC
A worklist command that invokes a checkpoint to use for restart processing and commits data to DB2.

sync point
A completion flag that is set during the execution of a worklist. The Execution program writes sync points to the SYNC table whenever it encounters –SYNC or –STOP commands in the worklist input stream. All SQL statements between sync points are executed as a single DB2 transaction. If a worklist is halted before completion for any reason, sync points enable you to begin processing the worklist from the last sync point.

synchronization
The process of identifying structural differences between two copies of the same data structure and then making the data structures identical. For example, separate groups of developers might be independently modifying several copies of a set of data structures. At various times, the copies need to be synchronized to ensure that all of the developers are using the same structure definitions.

table space
A page set that is used to store the rows of one or more tables. A table space represents one or more physical data sets.

target
The object of a migration or the object of a Compare process. When migrating objects or databases, the target database is the database to which you are migrating. The Compare process compares a source data structure with a target data structure. This process synchronizes two data structures, and the target is the data structure that contains the wanted changes.

See also Compare2.

task ID
A unit of work in the CM/PILOT component.

template
A method of obtaining specifications for the definition of an auxiliary table space, table, or index. This definition can be replicated for each of the partitions in the base table space.

See also baseline name template and work ID name template.
U

unit
A specific device, device type, or group of devices that are used in data set allocation.

unload data set
An object that is used to store data while DB2 objects are dropped and rebuilt. The unload data set is also referred to as a SYSREC data set.

UOW
Unit of Work. A unit of work consists of the worklist commands that are bounded by the -BEGU and -ENDU commands and that are run in a BMC Cross-System Image Manager (XIM) initiator.

UOWTRnnn
The ddname of the unit of work (UOW) output of the Execution component. When the worklist parallelism feature of the BMC Database Administration for DB2 solution is used, this output data set contains tracing records for each BMC Cross-System Image Manager (XIM) initiator.

unprotected baseline
A baseline that is not designated as protected from deletion.

See also protected baseline.

user options
A set of options that are stored in the user’s profile and that are used by Front End and JCL Generation for running ALTER or CHANGE MANAGER components. The user options are initially set from the installation options module the first time that the user runs ALTER or CHANGE MANAGER.

V

variable
See symbolic variable.

VCAT allocation
A volume placement parameter that specifies the data set high-level qualifier that is appropriate for the DB2 subsystem. Use this field for nonstorage group allocation only.

versioning
The process of comparing baselines that represent the same set of structures at different points in time in order to change a version of a data structure.
W

wildcard
A symbol that you can use to represent a value in SQL statements, filters, and name patterns. Valid wildcards for SQL statements and filters include the following symbols:

- the % and * represent any character string
- the _ and ? represent a single character

work ID
A unit of work with a two-part name (owner.name) that contains change or migration requests in change definition (CD) tables. The change or migration requests can be either imported or created manually through the Specification component.

work ID name
A work ID name is a string of up to eighteen alphanumeric characters excluding percent (%), asterisk (*), underscore (_), and space. When you specify a work ID name, you can use a wildcard pattern to display a group of similar names.

work ID name template
A template from which the name of a work ID is created. You can use this template to create or replace a work ID in batch mode. The name template contains the special character sequences #### and @@@@@@ or @@@@@@@@, in addition to text characters. An ascending sequence of numbers replaces the #### sequence, and the current date replaces the @@@@@@ or @@@@@@@@@ sequence when the work ID is generated. You can create and replace receive-type work IDs (using name templates) with the Import and Execution components.

work ID owner
The Authorization ID of the creator of the work ID.

worklist
A data set that contains commands for implementing a data structure change or migration.

worklist execution log
See AEXPRINT.

worklist parallelism
A feature in the BMC Database Administration for DB2 solution that reduces the elapsed time for executing a worklist that is generated by the BMC CHANGE MANAGER for DB2 product.

X

XIM
See Cross-System Image Manager.

XIM initiator
A program that executes one or more units of work (UOW).
Index

Symbols

##### 176, 629

$ change-level indicator 317

; (semicolon) as a delimiter 181, 243, 330

<BLANK> keyword 52

<CURRENT> keyword 50

<DEFLT> keyword 51

@ change-level indicator 317

@@@@@ 176, 629

Numerics

-1, used as value for PRIQTY attribute 50

-1, used as value for SECQTY attribute 50

A

A change-level indicator 316

ACM_PARALLEL_WORKLST AJXPOFIN keyword 71

ACMDMLD1 member of HLQ.DBCNTL data set 423

ACMDMLD2 member of HLQ.DBCNTL data set 375

ACMDMLD3 member of HLQ.DBCNTL data set 377

ACMDMLL1 member of HLQ.DBCNTL data set 551

ACMDMLL2 member of HLQ.DBCNTL data set 553, 621

ACMDMLM1 member of HLQ.DBCNTL data set 133

ACMDMLM2 member of HLQ.DBCNTL data set 138

ACMDMLP1 member of HLQ.DBCNTL data set 140

ACMDMLP2 member of HLQ.DBCNTL data set 451

ACMDMLP3 member of HLQ.DBCNTL data set 453

ACMDMLT1 member of HLQ.DBCNTL data set 477

ACMDMLT2 member of HLQ.DBCNTL data set 473

ACMDMLU1 member of HLQ.DBCNTL data set 555, 615

ACMDMLU2 member of HLQ.DBCNTL data set 548, 620

ACMDMLU3 member of HLQ.DBCNTL data set 428

ACMDMLU4 member of HLQ.DBCNTL data set 556, 615

ACMDMLU5 member of HLQ.DBCNTL data set 471

ACMDMLU6 member of HLQ.DBCNTL data set 421

ACMDMLU7 member of HLQ.DBCNTL data set 430

ACMDMLU8 member of HLQ.DBCNTL data set 401, 402

ACMDMLUA member of HLQ.DBCNTL data set 504

ACMDMLUB member of HLQ.DBCNTL data set 506

ACMDMLUC member of HLQ.DBCNTL data set 498

ACMDMLUD member of HLQ.DBCNTL data set 517

ACMDMLUE member of HLQ.DBCNTL data set 427

ACMDMLUF member of HLQ.DBCNTL data set 518

ACTIVATE command 737

activating XIM initiators 737

adding a column to a table 402

additional job steps in JCL, creating 636

advisory REORG-pending (AREOR) status 699

AEXIN input stream, overview 156

AEXPR

CHANGE MANAGER worklist 637

CM/PILOT worklist 637

data set 103, 729

used by Execution 103, 353

AEXPRnnn data set 105

AEXPTRAC data set 105

AEXSTLOG data set 104

AEXSTnnn data set 105

AESXnnn data set 105

AJXIN input stream 83

AJXPOFIN input stream 61, 83

aliases, deleting 422

Allocation Unit, Space Estimation parameter 340

ALLSTATSUPD keyword 718

ALTER

importing DDL 329, 381

using to merge subsystems 585

ALTER parameter of INCLUDE keyword 255

alter process, workflow 311

alternate delimiter 181, 243, 330

alternate terminator 181, 243, 330

alter-type work ID

converting to CDL 297, 329

creating 357

alter-type worklist

executing 562

generating 560

overview 347

ALUEUSX1 member 95

ALUIN input stream

building 61

overriding 61

used by Analysis 674

ALUIN keywords

baseline profiles 239

baselines 242
ALUIN keywords (continued)
displaying change rules 257
excluding objects 249
including CDL statements 255
including dependent objects 256
object attributes 251
object change flags 249
object owners 245
outbound migrate profiles 239
overriding change rules 246
partition attributes 252
resolving names 246
scope types 240

ALUPRINT
used by Analysis 348
used by Baseline 190
used by Baseline Report 192
used by Compare 262
used by Import 355
ALUXGRNT REXX executable 80
-AMS command 64, 102
AMSDELETEI keyword 705

Analysis
alter-type worklist 347
ALUPRINT diagnostic output file 348
analyzing change requests 345
changing installation options 61
including command types in a worklist 63
inputs 58, 99
locations, generating worklists for multiple 60
migrate-type worklist 78
override options 61, 346
primary key propagation 67
rebuilding a worklist 59
requesting a full-recovery baseline 184
using in change process 345

analyzing
alter-type work ID 560
change requests 560
migrate-type work ID 59

application tables, modifying 610

application-period temporal table, creating 433
applications
browsing 670
comments, adding 668
comments, changing 661
creating 668
deleting 671
dating 669
listing 669
listing task IDs for an application 672
maintaining 577
modifying 669
overview 666
planning to establish and maintain 578
profiles, displaying in a task ID 634

profiles, specifying 668
specifying work ID name template 668
strategies used in maintaining 577
task ID, specifying 630
archive data sets
dynamically allocating 703
assigning a unique name 128
ASUTIME attribute 51
attributes
ASUTIME 51
BUFFERPOOL 252
changing 328, 681
CHECK 252
COMPRESS 253
DEFINE 152, 251, 554, 557
DROPRESTRICT 252
DSSIZE 49
FREEPAGE 49, 253
IDENTITY 252
LOCKMAX 51, 251
LOCKSIZE 251
NUMPARTS 252
PATH 252
PCTFREE 49, 253
PIECESIZE 51, 244
PRIQTY 49, 253
SECQTY 49, 253
SEGSIZE 49, 251
specifying a range of values 49
specifying blank values 52
specifying default values 51
specifying new values 50
specifying nonnumeric values 51
specifying numeric values 49
specifying special numeric values 51
specifying values 49
START 252
STOGROUP 252
TRACKMOD 253

AU (Allocation Unit), Space Estimation parameter 344
-AUTH command 94, 102, 182
authorization switching
AUTHSW keyword 94
delimited identifiers 97
Global Authorization ID (GLID) 96
modes 94
used by Execution 94

authorizations
browsing in a baseline 187
commands 80
DB2 and data set, XIM 733
DB2 objects 186
restoring in baselines 180
shown in a baseline report 330
shown in a DDL file 192
AUTHSW installation option 94
AUTHSW keyword 94
AUTHSWGLID keyword 94, 97
AUTHSWOFF keyword 94
auxiliary indexes
  creating 318
  dropping 319
auxiliary objects
  building stand-alone image copies 709
  creating 536
  creating for a base table 519
  creating for a table space 318
  creating in a nonpartitioned table space 538
  creating in a single partition 531
  creating in partitioned table spaces 525
  creating scope rules 41
  defining attributes 318
  displayed in Specification 318
  dropping 44
  editing 44
  liking 44
  migrating 41, 44, 57, 121
  naming 319
  propagating name changes 56
  reviewing 530, 534, 541
  undoing changes 531, 535, 542
auxiliary table spaces
  creating 318
  dropping 319
auxiliary tables
  creating 318
  dropping 319
Avg Len, Space Estimation parameter 342
Avg Row Length, Space Estimation parameter 339
AX (Auxiliary Object LOB) action code 315
application milestone 173
browsing 187
checking contents 170
combining pre-execution baselines 351
complete 194
creating a profile 198
creating a step in a script 650
creating a structure-only baseline 201
data sets 189
definition 170
deleteage 177
deleting 193
deleting, automatically 213
deleting, explicitly 196, 215
deleting, using a baseline profile 214
deleting, using DELETEAGE 194
deleting, using RETAINMAX 195
establishing 173, 573
exception baselines 174
full recovery, creating 203
full recovery, overview 175
full recovery, script to create 207
full recovery, using 183
improving performance 197
incomplete 188, 194
inputs 170
managing 192
migrate profile reference 177
name template 176, 350, 574
NoBaseline option 193
object counts 211
objects, browsing 187
output 186
overview 170
periodic baselines 174
post-Execution 636
pre-Execution 636
profile 175, 198, 238
protecting baselines from deletion 192
recovery point 171, 174, 186
removing protection from baselines for deletion 192
report 190, 212
scope rules 178
script to create a full recovery baseline 207
specifying 634
structure and data recovery points 180
structure-only, creating 180, 201
structure-only, overview 174
structure-only, using 180
tables 187
template 176
types 180
unload data sets 184, 189
unprotecting baselines for deletion 192
using as the scope of a comparison 241
using in change migration process 573
using in compare process 230
backup copy, DD name 711
-BASE command 184, 188
base table spaces, building stand-alone image copies 709
base tables
  creating 519
  creating a clustering, partitioning index 482
  creating auxiliary objects 519, 536
  liking 536
  migrating 121
  undoing changes 320
baseline report
  authorizations 330
  creating a step in a script 657
  DDL 191
  generating 212
  overview 190
baseline tables 187
baseline to DDL file comparison 294
baselines
  ALUPRINT diagnostic output 190
baselines (continued)
  using in data structure recovery process 173
  using in old data recovery process 173
  using in structure recovery process 173
  viewing information 213
  work ID 188
  workflow 171
  worklist 182
baselines, deleting
  batch mode 216
baseline-to-baseline comparison 230, 283
baseline-to-catalog comparison 232, 287
BASICUNLOAD keyword 675
Batch Execution JCL Generation 86
-BIND command 65
BINDAUTH option 65
<BLANK> keyword 52
BLNAME ALUIN keyword 217
BLNAME keyword 183
BLPROFILE ALUIN keyword 217
BLPROFILE keyword 183
BLRECOVER keyword 186
BLRECOVERPOINT keyword 183, 186
BLRP data set 189
BLWORKID keyword 189, 350
BMC Software utilities
  BASIC UNLOAD 675
  BMCSTATS 109, 716
  BMCTRG 109
  CHECK PLUS 109, 698
  COPY PLUS 109, 708
  LOADPLUS 109, 695
  RECOVER PLUS 109, 689, 706
  REORG PLUS 109, 700
  restarting 109
  starting over 111
  UNLOAD PLUS 109, 695
BMC Software, contacting 2
-BMCC command
  processing in parallel 708
  using in a parallel worklist 741
BMCCHECK keyword 698
BMCCOPY keyword 708, 710
-BMCD command
  processing in parallel 676
  using fast unload 678
  using in a full-recovery baseline 184, 189
  using in a parallel worklist 741
  using with -UNLB command 679
BMCFASTL installation option 678, 696
BMCFASTLOAD keyword 678, 696
-BMCK command
  processing in parallel 698
  using in a parallel worklist 741
-BMCL command
  FORMAT BMCUNLOAD 696
  processing in parallel 695
  REDEFINE NO parameter 695
  specifying multiple SYSUT data sets 696
  using in a parallel worklist 741
  using with -LOLB command 680
-BMCR command 701
BMCREBUILD keyword 706
BMCREORG keyword 700
-BMCS command
  generating statistics 716
  processing in parallel 716
  using in a parallel worklist 741
BMCSPACE keyword 471
BMCSSTATS utility
  collecting statistics 716
  performing space estimation 337
  restarting 109
  using in a parallel worklist 741
BMCSSTATSUPD keyword 718
BMCTRIG utility 109
BMCCUNLOAD keyword 676
BMIDB2X9 member of the HLQ.UDBCNTL data set 614
BMIDB2XA member of the HLQ.UDBCNTL data set 614
BROWSE (browse diagnostics) command 636
browsing
  applications 670
  baselines 187
  DB2 authorizations 187
  DML statements 641
  scripts 662
  task IDs 640
  BUFFERPOOL attribute 252

C

CA-ACF2, authorizing procedures for XIM 733
calculating data set size, dynamically allocated unload
data sets 684
catalog
  comparing 223
  maintaining catalog information for data modeling
tools 583
  merging subsystems 585
  catalog baseline 180
catalog baseline profiles
  creating 198
  creating scope rules 178, 199
  used as scope 238
catalog baseline-to-catalog baseline comparison 230, 283
catalog baseline-to-catalog comparison 232, 287
catalog baseline-to-DDL file comparison 234
catalog baseline-to-worklist comparison 234
catalog indirection, overview 59
  CATALOG MANAGER Drop Recovery 89
catalog statistics, suppress updating 718
cataloged procedures 93
catalog-to-baseline comparison 290
catalog-to-catalog baseline comparison 233
catalog-to-catalog comparison
  considerations 225
defining scope 243
defining scope for a remote location 266
handling unresolved owner or object 247
improving performance 250
input 243
performing 272
requirements 225
resolving changes to names 245
using a script 300
using an outbound migrate profile 247
catalog-to-DDL file comparison 226, 276
catalog-to-worklist comparison 226
CATAUDIT keyword 99
CA-Top Secret, authorizing procedures for XIM 733
CATRECOVER keyword 89, 99
CD tables, used by Import 328
CDL (Change Definition Language)
catalog information 255
CDLDEP parameter 256
comparison report 263
evaluating 577
excluding command types and catalog information 254
generating changes 574
generating from a comparison 258
generating from a work ID 297
generating multiple CDL files using locations 254
handling unresolved owner or object name in CDL 247
importing in CHANGE MANAGER 379
including command types and catalog information 254
override options 256
overview 329
recording changes to objects 262
used by Compare 262
used by Import 329
using in change feedback process 581
using in change migration process 574
-CDL command 255
CDLCHANGERULES keyword 253, 263
CDLDEP parameter of INCLUDE keyword 256
Change Definition Language. See CDL.
change feedback process 580
CHANGE MANAGER
  importing CDL 379
  importing DDL 379
  importing DML 379
  override options 634
change migration process
  acting as receiver 575
  acting as sender 573
  establishing control points 577
  implementing changes 577
  receiving DDL 576
  using inbound migrate profiles 577
change propagation
dropped table columns in indexes 77
primary keys 67
table columns in base views 68
text 258
text dependencies 76
change requests
  analyzing 560
  executing 562
change rules
  application order 53
  applying to aliases and synonyms 48
  applying to trigger text 335
  change-type 46
  comparisons 46
  exclude 46
  force 46
  generating a migrate worklist using change rules 155
  import 46
  include 46
  migration 46
  name resolution in a catalog to catalog comparison 247
  object types 46
  specifying values 49
  suppress 46, 253
  tailoring worklists 48
  types 46
  using in migrate-type work IDs 46
  using in outbound migrate profiles 38, 48
  using to set objects to DEFINE NO 152
change validation of text 334
change-level indicators, list of 316
CHANGERULESIN2 keyword
  avoiding use of automatic change rules 257
  determining when objects are renamed 257
  displaying change rules 247
  improving performance of comparison 265
  resolving names of objects and creators 227
CHANGERULESIN2 NULL keyword 247
change-type change rules 46
characters, nonprintable 53
CHECK attribute 252
check constraint text validation
  by Analysis 75
  by Compare 258, 334
check constraint text, change propagation 76
CHECK DATA utility
  checking table spaces 698
  using in a parallel worklist 741
CHECK PLUS utility
  checking table spaces 698
  restarting 109
  using in a parallel worklist 741
CHEK command
  processing in parallel 698
  using in a parallel worklist 741
child tables 65
CHKSQNUM macro 81
choosing a script 646
CISIZE4K ALUIN keyword 70
cleanup, data set 93
cloning 90
clustering, changing for an index 397
clustering, partitioning index, creating for a base table 482
CM/PILOT scripts 354
CM/PILOT worklist
  creating 634
  executing 632
  failure, determining point of 639
  hash verification numbers 635
  processing 632
  processing status, viewing 639
  restarting execution over 642
  starting execution over 633
CMP_STRUCTURE_CHG script 300
CMPI keyword 242
CMPTYPE keyword 242, 245
columns
  adding to a table 402
  changing the data type 405
  changing the length 405
  copying 423
  creating a LOB 521
  creating a ROWID 520, 544
dropping from a table 404
  editing in a unique constraint 390
  identity 327
combining pre-execution baselines 351
commands
  BROWSE (browse diagnostics) 636
  DROPALL 44, 316
  MIGALL 44
  PE (propagate estimates) 339, 343
  UNDOALL 44, 316
  VIEW (view task ID execution status) 636
  ZOOM 43
commands, worklist
  -AMS 64, 102
  -AUTH 94, 102, 182
  -BASE 184, 188
  -BEGU 102
  -BIND 65
  -BMCD 189
  -BMCL 696
  -BMCR 701
  -BMCS 716
  -CDL 255
  -DATA 64
  -GLID 94
  -ISMT 69, 185
-JCLP 189, 696, 701
-MIGR 78
-REXC 80
-RNST 717
-SETP 102
-SETS 94, 102
-SQL 63, 102
-STOP 102
-SYNC 102
-UNRC 184, 185, 189
-UOWC 102
-WKID 98
comments 181, 243, 330
COMMIT keyword 197
COMMITFREQ keyword 680
COMMITFSIZE keyword 680
comparisons
  ALUPRINT diagnostic output 262
  application versioning 229
  CDL 258
  CDLDEP parameter 256
creating a step in a script 651
data modeling tool, using 226
data structure, post-execution 636
data structure, pre-execution 636
displaying a list of objects 244
displaying old and new values 263
generating CDL from a work ID 263, 329
  generating multiple CDL files using locations 254
  handling unresolved owner or object in CDL 247
  improving performance 250, 264
inputs 223
name resolution 247
name template 350
ordering inputs 223
outbound migrate profile 258
outputs 261
overriding options 248
profiles used 247, 253
propagating dependent object changes 256
rename 242
rename table 576
renamed object resolution 259, 261
report 263
renaming object resolution 259, 261
scope 236, 238
setting options 248
specifying a table owner 244
specifying information included in a report 263
specifying locations 245
specifying what to include in the CDL file 255
synchronizing objects 223
text validation 258
using a CM/PILOT script 300
using a DB2 catalog 224, 243
using a name template in a multi-step batch job 350
using a profile to scope 272
comparisons (continued)
  using baselines 230
  using DDL files 242
  using in change migration process 574
  using worklists 240
  work ID, name template 350
  workflow 221
comparisons, baseline
  baseline to baseline 230
  baseline to catalog 232
  baseline to DDL file 294
  catalog baseline to a catalog 287
  catalog baseline to catalog 232, 287
  catalog baseline to catalog baseline 283
  catalog baseline to DDL file 234
  catalog baseline to worklist 234
  DDL baseline to DDL file 234
  DDL baseline to worklist 234
  local baseline to remote baseline 283
comparisons, catalog
  catalog to baseline comparison 290
  catalog to catalog 223
  catalog to catalog baseline 233
  catalog to DDL file 226, 227, 229, 276
  catalog to worklist 226
  local catalog to a remote catalog 266
comparisons, DDL file
  baseline to DDL file 294
  DDL file to DDL file 228, 280
  DDL file to worklist 228, 280
comparisons, worklist
  baseline to DDL file or migrate-type worklist 294
  worklist to DDL file 228, 280
  worklist to worklist 228, 280
compiling SLIBs 723
COMPRESS attribute 253
Compress, Space Estimation parameter 340, 343
constraints
  creating a primary 384
  creating a unique constraint 386
  editing 390
control characters
  using in change rules 53
  using in DDL files 181, 241
  using in limit keys 485
control interval size, used in VSAM data set creation 70
control system, feeding changes back 580
controlling access to XIM 733
conventions, documentation 24
converting data
  changing attributes 328
  VARCHAR to BLOB or CLOB data types 681
converting table spaces 320
converting to LOB data types 328, 681
converting VCAT-defined to STOGROUP-defined partitions 429
copy data sets
  dynamically allocating 703
copy data sets, dynamically allocating 712
Copy Migration feature 691
COPY PLUS EXPORT command 691
COPY PLUS utility
  creating partition-level image copies 711
  restarting 109
  specifying 708
  using in a parallel worklist 741
copy utilities 707
COPY utility 708
COPYDDN keyword 708, 709
COPYEXPORT ALUIN keyword 691
copying
  BMC-supplied scripts 665
  columns 423
  user-defined scripts 666
CREATE parameter of INCLUDE keyword 255
creating
  alter-type work ID 357
  applications 668
  auxiliary objects 318
  baseline (strong receiver) 590
  data structure for a new application 411
  data structure, using specification 408
  database 548
  DDL baseline 411
  inbound migrate profile 361
  JCL for processing CM/PILOT worklist 635
  migrate-type work ID 114
  MQT from a view 431
  primary constraint 384
  receive-type work ID 155
  scripts 648
  synonyms 428
table space 548
task IDs 628
task IDs, from a script 664
task IDs, from an existing application 671
unique constraint for a table 386
worklists 634
<CURRENT> keyword 50
CURRENT RULES register 319
customer support 3
Cxxxxxxxxx DD name 711

D

D (Drop) action code 315
D change-level indicator 316
DASD MANAGER PLUS, updating with BMCSTATS 718
DASDMAN installation option 337
data
  migrating to a different subsystem 161
  migrating within the same subsystem 161
  unloading and loading LOB and XML 64
-DATA command 64
data conversions
  changing a column’s data type 405
  converting to LOB data types 328, 681
  using the UNLOAD PLUS utility with the
  LOADPLUS utility 695
Data Manipulation Language (DML), importing in
  CHANGE MANAGER 379
data modeling tools
  importing DDL 582
  maintaining catalog information 583
data set disposition
  restart processing 108
  startover processing 111
data set sizing 83
data sets
  AEXPR103
  AEXPRnnn 105
  AEXPRAC 105
  AEXSTLOG 104
  AEXSTnnn 105
  AEXSYnnn 105
  allocating 255
  baseline, unload 189
  CM/PILOT worklist 632
  copying 708
  JCL for processing CM/PILOT worklist 633
  local 708
  registering 708
  remote 708
  sizing 86, 88
  SYSnnnnn 106
  SYSUDUMP 730
  tape 697
  UOWTRnnn 105
  WRK100n 703
data sharing environment
  invoking CHANGE MANAGER 732
  requirements 732
  using DSNEXIT concatenation 732
  using DSNLOAD concatenation 732
data structures
  adding WITH RESTRICT ON DROP 426
  altering 310
  analyzing changes 345
  changing 408
  changing, using DML 415
  copying a column 423
  creating using specification 408
  deleting aliases 422
  deleting using specification 408
  falling back with current data 595
  falling back with old data 601
  importing 329
  importing for a new application 411
  migrating 57
  migrating changes 572
  migrating to a different subsystem 160
  migrating within the same subsystem 160
  receiving changes 587, 590
  receiving DDL 411
  recovering structures and data 174
  recovering with current data 596
  recovering with old data 602
  recovery process 173
  recovery process, using full-recovery baselines 175
  restoring with current data 595
  restoring with old data 601
  synchronizing 223
  updating index parts 424
  date formats
  baselinel template 176
  work ID name template 629
  databases
  creating 548
  creating implicitly 325
  specifying <DEFLT> as the name 325
  updating index parts 424
  Database Administration
  password 679
  worklist parallelism 727
  DB2 catalog
  used by Analysis 59
  used by Baseline 180
  used by Compare 243
  used by Import 334
  DB2 utilities
  CHECK DATA 698
  COPY 708
  LOAD 109, 697
  REBUILD INDEX 706
  REORG 109, 701
  REPAIR 70
  RUNSTATS 716
  UNLOAD 686
  DB2STATSUPD keyword 718
  DBRMLIB keyword 65
  DCPYLOCB OUTPUT descriptor 714
  DCPYLOCB TEMPLATE descriptor 714
  DCPYLOCP OUTPUT descriptor 714
  DCPYLOCP TEMPLATE descriptor 714
  DCPYREMB OUTPUT descriptor 714
  DCPYREMB TEMPLATE descriptor 714
  DD name, copies 711
Index 773

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

DDL
  baseline 181
  comment 181, 243, 330
  importing 329, 381, 411
  importing in ALTER 381
  importing in CHANGE MANAGER 379
  incremental DDL 330
  receiving 411
  scope 240
  setting columns to NOT NULL 370
  used by Baseline 181
  used by Compare 242
  used by Import 329
  using a script to receive 411
  using in a baseline report 191
  using in change migration process 576
  using in data modeling tools 582

DDL baseline to DDL file comparison 234
DDL baseline to worklist comparison 234
DDL file to DDL file comparison 228, 280
DDL file to worklist comparison 228, 280

<DEFLT> keyword 51
DEFAULTOFF keyword 257
DEFER YES parameter 70
DEFERUNIQUEIX keyword 70
DEFINE attribute
  setting with a change rule 152
  setting with a script 554
  updating with a script 557
  using in comparisons 251

DEFINE NO
  generation of the -ISMT command 69
  used in PeopleSoft environment 615

DELETE_WORKID script 373, 376
DELETEAGE keyword
  deleting baselines 194
  overriding 195

delimiters 181, 243, 330
delimited identifiers, authorization switching 97
detail panels, used by Specification 326
DETAIL parameter of REPORT keyword 263
Device Type, Space Estimation parameter for table space 338

DevIC, Space Estimation parameter 342
diagnostic output
  for Analysis 348
  for Baseline 190

DML
  adding columns to PLAN_TABLE 420
  browsing statements 641
  changing data structures 415
  changing statements 125
  errors 638
  evaluating changes before execution 637
  importing 379
  overview 39
  trial report 637

DML report 637
DML section 638
DML_MIGRATE script 123
DML_STRUCTURE_CHG script 415
documentation information 24
DROP parameter of INCLUDE keyword 255
Drop Recovery option 89
DROP RESTRICT clause 77, 426
DROPALL command 44, 316
dropping a column from a table 404
DROPRESTRICT attribute 252
DSNICOPY utility, using to perform data-only migrations 587
DSSIZE attribute 49
Dssize, Space Estimation parameter 340
Dxxxyyyy DD name 711

dynamic allocation
  archive data sets 703
  copy data sets 703, 712
data sets 92
discard data sets 703
punch data sets 703
reorg data sets 703
requirement for worklist parallelism 740
sortout data sets 703
SYSCOPY data sets 712
SYSREC data sets 682, 687
sysrec data sets 703
sysut data sets 703
unload data sets 682, 687
using the BMC REORG PLUS utility 703
using the COPY PLUS utility 712
using the IBM COPY utility 712
using the IBM REORG utility 703
using the IBM UNLOAD utility 687
using the RECOVER PLUS utility 712
using the UNLOAD PLUS utility 682

DYNCOPY installation option 740
DYNCOPY keyword  
  dynamically allocated image copy data sets 710, 712  
  using with UTILCOPY and NOUTILCOPY 711, 714  
  worklist parallelism 740  
DYNREORG keyword 703  
DYNUNLD installation option 740  
DYNUNLD keyword 682, 687, 740  
DYNWORKUNIT keyword 93  

E  
E (Edit) action code 315  
edit macros  
  CHKSQNUM 81  
  FIXSQNUM 81  
editing  
  applications 669  
  CM/PILOT worklists 635  
  DML statements 631  
  task IDs 631  
electronic documentation 23  
encoding scheme, changing 518  
ENFORCE CONSTRAINTS option 699  
ENFORCE NO option 699  
ENFORCE option 699  
enforcing referential constraints on a LOAD 699  
ENV keyword 67  
environments, PeopleSoft 613  
EOJ (end of job) 102  
ERP applications  
  challenges 610  
  changing multi-table table spaces to single-table table spaces 550  
  changing table spaces and indexes to DEFINE NO 132  
  CM/PILOT solutions 550  
  creating a new database and table spaces for tables 548  
  creating copies of environments 613  
  creating environments 611  
  managing 548  
  moving empty tables to one table space 552  
  optimizing the design 618  
  taking baselines 611  
estimating space  
  index 341, 460  
  table space 337, 453  
ESTROWS parameter, -BMCD worklist command 684  
evaluating CDL 577  
extception baseline 174  
Exclude option, scope rules 148  
exclude-type change rules 46  
Exec Corn (execution completed) 100  
Exec Srt (execution started) 100  
Execution return codes 98  

execution, worklist  
  additional job steps, to specify 564  
  AEXPRINT diagnostic output file 103, 353  
  alter-type worklist 562  
  authorization switching 94  
  Batch Execution JCL Generation 86  
  cleanup job set for data sets 93  
  creating a step in a script 656  
  creating a work ID in batch 157  
  data set sizing for 86  
  deleting sync records 102  
  editing JCL 91  
  generating JCL to create execution JCL in batch 86  
  initial 84  
  inputs 89, 349  
  LOAD utility 109  
  migrate-type worklist 82  
  optional job steps 111  
  options 349  
  outputs 103, 353  
  override options 349  
  overview 89, 349  
  preallocation step 92  
  REORG utility 109  
  replacing a work ID in batch 157  
  restart, considerations for multistep jobs 352  
  restart, overview 84  
  restarting 84  
  specifying a multi-step job 350  
  start over 85  
  start over, overview 85  
  status 99  
  sync records, deleting 102  
  sync table 101  
  unload data sets 99  
  utilities 98  
  View Execution Status 101  
  work ID, creating in batch 157  
  work ID, replacing in batch 157  
  work ID, status 100  
EXPFIELD OUTPUT descriptor 714  
eXCyyy DD name 711  

F  
falling back to a previous data structure  
  with current data 595  
  with old data 601  
feeding back changes 580  
file tailoring 722
INDEX 775

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

FILEREF parameter 185
FIND command 43
FIXSQNUM macro 81
Force-type change rules 46
foreign keys
  child table 65
  primary key propagation 67
FORMAT BMCLOAD 678
FORMAT BMUNLOAD 696
Free Page, Space Estimation parameter 344
FREEPAGE attribute 49, 253
Freepage, Space Estimation parameter 340
full recovery, load requirements 185
full-recovery baseline
  authorizations for dropped objects 175
  -BASE command 184, 188
  -BMCD command 184
  creating 183, 203, 204
  falling back 595, 601
  generating a worklist 203
  -ISMT command 185
  overview 175
  requesting 204
  -UNLI command 184
  using 183
Fxxyyyy DD name 711

G
GDG (generation data group), using dynamic allocation 714
GENERATED ALWAYS, defined for identity column 327
GENERATED ALWAYS, ROWID column 676
GENERATED BY DEFAULT, ROWID column 676
generating CDL for changes 574
generating JCL
  for generation data groups 82
  to create Execution JCL in batch 86
generating multiple CDL files 254
generating statistics 718
GLID (global authorization ID)
  override option 62
  overview 96
  -GLID command 94, 182
grants, migrating 80
group ID 254

H
hash verification numbers 78, 347, 635
Help, online 23
hexadecimal string 181
High Speed Structure Change (HSSC)
  creating objects for SHRLEVEL CHANGE 473
  modifying and migrating objects for SHRLEVEL REFERENCE 476
HISTORY ALL parameter 717
HISTORY NONE parameter 717
history table
  creating for a system-period temporal table 437
  creating from a base table 443
  creating from an existing history table 440
HISTORYALL keyword 717
HLQ.DBCNTL data set
  ACMDMLD1 423
  ACMDMLD2 375
  ACMDMLD3 377
  ACMDMLL1 551
  ACMDMLL2 553, 621
  ACMDMLM1 133
  ACMDMLM2 138
  ACMDMLP1 140
  ACMDMLP2 451
  ACMDMLP3 453
  ACMDMLT1 477
  ACMDMLT2 473
  ACMDMLU1 555, 615
  ACMDMLU2 548, 620
  ACMDMLU3 428
  ACMDMLU4 556, 615
  ACMDMLU5 471
  ACMDMLU6 421
  ACMDMLU7 430
  ACMDMLU8 401, 402
  ACMDMLU9 504
  ACMDMLUA 506
  ACMDMLUB 498
  ACMDMLUC 479
  ACMDMLUD 517
  ACMDMLUF 427
  ACMDMLUF 518
IBM DSNUTILB utility control program 703
IBM RUNSTATS utility 337
IBM utilities. See DB2 utilities
IBMCHECK keyword 698
IBMCOPY keyword 708, 710
IBMLOAD keyword 697
IBMREBUILD keyword 706
IBMREORG keyword 701
IBMUNLOAD keyword 687
IDENTITY attribute 252
identity column
  generating new values 328
  preserving existing values 327
  identity columns 327
IDENTITYOVERRIDE YES parameter, BMC LOADPLUS 327
image copies
   building 709
   registering 708
   SHRLEVEL 686, 690
implicit databases, converting from explicit 517
implicit table spaces
   auxiliary objects 326
   considerations for changing data structures 327
   ROWID GENERATED BY DEFAULT column 326
IMPORT command, synchronizing table spaces 693
Import component, CD tables 328
importing
   ALUPRINT diagnostic output 335
   CDL 329, 379
   creating a step in a script 653
   data structure for a new application 411
   DB2 catalog 334
   DDL 329, 379, 411
   DML 379
   external files 328
   implementing changes 577
   inbound migrate profiles 332
   incremental DDL 330
   new structures 329
   outputs 328
   overview 328
   text validation 334
inactivating XIM initiators 736
inbound migrate profiles
   creating 361
   creating change rules 152
   described 333
   using in change feedback process 581
   using in change migration process 577
INCLUDE keyword 63, 255, 741
include option, scope rules 40
include-type change rules 46
index on expression text validation
   by Analysis 75
   by Compare 258, 334
   change propagation 76
index partitions
   setting primary quantities 470
   setting secondary quantities 470, 471
   updating 424
index-controlled partitioned table spaces
   converting to partition-by-growth table spaces 479
   converting to range-partitioned table spaces 505, 506
   converting to table-controlled partitioned table spaces 500, 504
index-controlled partitioning, migrating to range partitioning 134
indexes
   auxiliary 318
   changing the clustering 397
   changing the key columns 397
   changing the key sequence 397
   changing the padding 397
   changing the uniqueness 397
   changing to DEFINE NO 132
   creating a primary constraint 384
   improving performance 197, 264
   propagating dropped table columns 77
   reorganizing 699
   specified with DEFINE NO 557
   unique 70
   updating 427
   updating to DEFINE NO 554
inline LOB columns
   migrating 138
   updating length 558
installation options
   AUTHSW 94
   BMCFASTL 678, 696
   DASDMAN 337
   DYNCOPY 740
   DYNUNLD 740
   REGALL 708
installation, space requirements 730
internal format 53
   -ISMT command 69, 185
ISPF interface
   CHKSQNUM macro 81
   file tailoring 722
   file tailoring for JCL generation 84
   FIXSQNUM macro 81
   REXX macro 80
J
JCL
   additional job steps, creating 636
   creating for CM/PILOT worklist processing 633
   editing Execution JCL 91
   editing for CM/PILOT worklist processing 635
   submitting for CHANGE MANAGER worklist processing 642
   submitting for CM/PILOT worklist processing 635
   symbolic variable use 633
JCL generation
   batch 86
   disposition 108
   input 82
   ISPF file tailoring 84
   output 82
   receive-side JCL 89
-JCLF command 189, 696, 701
JES3 restrictions for worklist parallelism 732
**K**

- katakana characters 53
- KEEP-ALL-PRIMARY parameter, OVERRIDE ALUIN keyword 257
- key columns, changing for an index 397
- key sequence, changing for an index 397
- KEYCARD ALUIN keyword 717
- KEYCARD parameter 717
- keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BLANK&gt;</td>
<td>52</td>
</tr>
<tr>
<td>&lt;CURRENT&gt;</td>
<td>50</td>
</tr>
<tr>
<td>&lt;DEFLT&gt;</td>
<td>51, 325</td>
</tr>
<tr>
<td>ACM_PARALLEL_WORKLST</td>
<td>71</td>
</tr>
<tr>
<td>ALLSTATSUPD</td>
<td>718</td>
</tr>
<tr>
<td>AMSDELETEI</td>
<td>705</td>
</tr>
<tr>
<td>AUTHSW</td>
<td>94</td>
</tr>
<tr>
<td>AUTHSWGLID</td>
<td>94, 97</td>
</tr>
<tr>
<td>AUTHSWOFF</td>
<td>94</td>
</tr>
<tr>
<td>BASICUNLOAD</td>
<td>675</td>
</tr>
<tr>
<td>BLNAME</td>
<td>183</td>
</tr>
<tr>
<td>BLPROFILE</td>
<td>183</td>
</tr>
<tr>
<td>BLRECOVER</td>
<td>186</td>
</tr>
<tr>
<td>BLRECOVERPOINT</td>
<td>183</td>
</tr>
<tr>
<td>BLWORKID</td>
<td>189, 350</td>
</tr>
<tr>
<td>BMCCHECK</td>
<td>698</td>
</tr>
<tr>
<td>BMCCOPY</td>
<td>708, 710</td>
</tr>
<tr>
<td>BMCFASTLOAD</td>
<td>678, 696</td>
</tr>
<tr>
<td>BMCREBUILD</td>
<td>706</td>
</tr>
<tr>
<td>BMCREORG</td>
<td>700</td>
</tr>
<tr>
<td>BMCSPACE</td>
<td>471</td>
</tr>
<tr>
<td>BMCSUPD</td>
<td>718</td>
</tr>
<tr>
<td>BMLOAD</td>
<td>676</td>
</tr>
<tr>
<td>CATAUDIT</td>
<td>99</td>
</tr>
<tr>
<td>CATRECOVER</td>
<td>89, 99</td>
</tr>
<tr>
<td>CDLCHANGERULES</td>
<td>253, 263</td>
</tr>
<tr>
<td>CHANGERULESIN2</td>
<td>227, 247, 257, 265</td>
</tr>
<tr>
<td>CHANGERULESIN2 NULL</td>
<td>247</td>
</tr>
<tr>
<td>CISIZE4K</td>
<td>70</td>
</tr>
<tr>
<td>CMPIN</td>
<td>242</td>
</tr>
<tr>
<td>CMPTYPE</td>
<td>242, 245</td>
</tr>
<tr>
<td>COMMIT</td>
<td>197</td>
</tr>
<tr>
<td>COMMITFREQ</td>
<td>680</td>
</tr>
<tr>
<td>COMMITSIZE</td>
<td>680</td>
</tr>
<tr>
<td>COPYDDN</td>
<td>708, 709</td>
</tr>
<tr>
<td>DB2STATSUPD</td>
<td>718</td>
</tr>
<tr>
<td>DEFAULTOFF</td>
<td>257</td>
</tr>
<tr>
<td>DEFERUNIQUEIX</td>
<td>70</td>
</tr>
<tr>
<td>DELETEAGE</td>
<td>194</td>
</tr>
<tr>
<td>DYNCOPY</td>
<td>710, 711, 712, 714, 740</td>
</tr>
<tr>
<td>DYNREORG</td>
<td>703</td>
</tr>
<tr>
<td>DYNUNLD</td>
<td>682, 687, 740</td>
</tr>
<tr>
<td>DYNWORKUNIT</td>
<td>93</td>
</tr>
<tr>
<td>ENV</td>
<td>67</td>
</tr>
<tr>
<td>HISTORYALL</td>
<td>717</td>
</tr>
<tr>
<td>IBMCHECK</td>
<td>698</td>
</tr>
<tr>
<td>IBMCOPY</td>
<td>708, 710</td>
</tr>
<tr>
<td>IBMLOAD</td>
<td>697</td>
</tr>
<tr>
<td>IBMREBUILD</td>
<td>706</td>
</tr>
<tr>
<td>IBMREORG</td>
<td>701</td>
</tr>
<tr>
<td>IBMUNLOAD</td>
<td>687</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>63, 255, 741</td>
</tr>
<tr>
<td>LOGBT32M</td>
<td>681</td>
</tr>
<tr>
<td>LOCATION</td>
<td>242, 245</td>
</tr>
<tr>
<td>LOG</td>
<td>697</td>
</tr>
<tr>
<td>MAXSYSUT</td>
<td>695, 700</td>
</tr>
<tr>
<td>MIGLOCATIONS</td>
<td>254</td>
</tr>
<tr>
<td>MIGSCOPE</td>
<td>60</td>
</tr>
<tr>
<td>MULTITBDS</td>
<td>678</td>
</tr>
<tr>
<td>NOALIAS</td>
<td>249</td>
</tr>
<tr>
<td>NOAMSDELETEI</td>
<td>705</td>
</tr>
<tr>
<td>NOAUTHSW</td>
<td>94</td>
</tr>
<tr>
<td>NOAUXILIARY</td>
<td>249</td>
</tr>
<tr>
<td>NOBASELINE</td>
<td>196</td>
</tr>
<tr>
<td>NOBMCFASTLOAD</td>
<td>678, 696</td>
</tr>
<tr>
<td>NOBUFFERPOOL</td>
<td>252</td>
</tr>
<tr>
<td>NOCHECKCNST</td>
<td>249</td>
</tr>
<tr>
<td>NOCOMPRESS</td>
<td>253</td>
</tr>
<tr>
<td>NOCOPY</td>
<td>707, 710, 711, 714</td>
</tr>
<tr>
<td>NOCOPYFORCE</td>
<td>707, 710, 711, 714</td>
</tr>
<tr>
<td>NODATABASE</td>
<td>247, 249</td>
</tr>
<tr>
<td>NODEREAD</td>
<td>525</td>
</tr>
<tr>
<td>NODROPRESTRICT</td>
<td>252</td>
</tr>
<tr>
<td>NOFOREIGNKEY</td>
<td>249</td>
</tr>
<tr>
<td>NOFREEPAGE</td>
<td>253</td>
</tr>
<tr>
<td>NOHISTORY</td>
<td>717</td>
</tr>
<tr>
<td>NOIDENTITY</td>
<td>252</td>
</tr>
<tr>
<td>NOINDEX</td>
<td>249</td>
</tr>
<tr>
<td>NOLOCKSIZE</td>
<td>251</td>
</tr>
<tr>
<td>NOORDERBY</td>
<td>69</td>
</tr>
<tr>
<td>NOORDERBY</td>
<td>69</td>
</tr>
<tr>
<td>NOPARTCOPY</td>
<td>711</td>
</tr>
<tr>
<td>NOPARTITION</td>
<td>252</td>
</tr>
<tr>
<td>NOPATH</td>
<td>252</td>
</tr>
<tr>
<td>NOPCTFREE</td>
<td>253</td>
</tr>
<tr>
<td>NOPRIQTY</td>
<td>253</td>
</tr>
<tr>
<td>NOREBUILD</td>
<td>706</td>
</tr>
<tr>
<td>NOREGIDENTITY</td>
<td>328</td>
</tr>
<tr>
<td>NOREMARKS</td>
<td>251</td>
</tr>
<tr>
<td>NOSECQTY</td>
<td>253</td>
</tr>
<tr>
<td>NOSEG.SIZE</td>
<td>251</td>
</tr>
<tr>
<td>NOSTARTVALUE</td>
<td>252</td>
</tr>
<tr>
<td>NOSTATS</td>
<td>716</td>
</tr>
<tr>
<td>NOSTOGROUP</td>
<td>249</td>
</tr>
<tr>
<td>NOSYNONYM</td>
<td>249</td>
</tr>
<tr>
<td>NOTABLE</td>
<td>249</td>
</tr>
<tr>
<td>NOTABLEACCESS</td>
<td>72</td>
</tr>
<tr>
<td>NOTABLEALL</td>
<td>716</td>
</tr>
<tr>
<td>NOTABLESPACE</td>
<td>249</td>
</tr>
<tr>
<td>NOTRACKMOD</td>
<td>253</td>
</tr>
<tr>
<td>NOTRIGGER</td>
<td>249</td>
</tr>
<tr>
<td>NOUNLOADEMPTY</td>
<td>69, 185</td>
</tr>
<tr>
<td>NOUTILCOPY</td>
<td>710, 711, 714</td>
</tr>
<tr>
<td>NOVIEW</td>
<td>249</td>
</tr>
</tbody>
</table>
keywords (continued)
NOVIEWCHECK 252
NOVVALPROP 69
OBJECTS 190
PARALLEL 71, 711, 714
PARTCOPY 711
PIC 709
PKEYPROP 67
PROTECT 193
REBUILD 59
REGENIDENTITY 328
REGISTER 708
REORGALT 72
REORGONLINE 702
REPLACEWORKID 157, 334
REPORTTYPE 190
RESMAXIC 689
RESTARTPARM 109
RETNMAX 194
SCOPE 237, 239
SCOPERULE 237, 245
SCOPETYPE 237, 239, 240
SENDONLY 89
SEQI 60, 81
SHOWRULES 257
SHRLEVELCHG 686, 690
SHRLEVELREF 686, 690
SINGLEPHASE 71
STANDALONESTATS 716, 717
STARTOVER 111
STOPCOMMIT 70
SYNCDELETE 102
TABLEALL 716
THRESHOLD 197, 265
UNLDCOLL 684
UNLOADCOPY 685
UNLOADCOPYDS 685
UTILCOPY 704, 710, 711, 714
UTILSTATS 717
VALWARN 68
VVALPROP 68
WORKID 188, 333
WORKIDMERGE 334
XIMPCTSURMN 93
XIMRETRYCOUNT 93
XIMRETRYMINS 92
XIMRETRYSECS 92

L
L (Like) action code 315
L change-level indicator 316
length, changing for a column 405
limit key 181

M
M (Migrate) action code 44
macros
CHKSQNUM 81
FIXSQNUM 81
REXX 80
maintaining
applications 577, 666
scripts 646
sync tables 360, 373, 641
task IDs 627
managing baselines 192
matching
  nonnumeric values 51
  numeric values 49
materialized query table (MQT) text validation
  by Analysis 75
  by Compare 258, 334
Maxpartitions, Space Estimation parameter 340
Maxrows, Space Estimation parameter 340
MAXSYSREC ALUIN keyword 677, 683
MAXSYSUT keyword 695, 700
merging subsystems 585
MIGALL command 44
MIGLOCATION5 keyword 254
-MIGR command 78
migrate-type work ID
  change rules 59
  migrate options 59
  specifying change rules 47
  used by Analysis 59
  used by Execution 89
  used in migrate process 37
migrate-type worklist
  generating 153
  overview 78
  used by Execution 82
  work ID 90
migrating
databases 128
  single-table table space to a partition-by-growth table space 136
  tables in explicit databases to implicit databases 137
migrating inline LOB columns 138
migration
  application changes to other subsystems 584
  authorizations 80
  auxiliary objects 41, 44, 57, 121
  change rules 46
  changes to data structures 572
  data only 57, 161
  data structures 57
  data structures and data 161
  data structures only 160
  establishing a baseline 573
  generating a migrate-type worklist using change rules 155
  generating a worklist using a migrate profile with CHANGE MANAGER 155
  generating CDL 574
  grants 80
  multiple locations 39
  outbound migrate profiles 38, 146
  overview 33
  performing a change migration process 584
  process 33
  receiving DDL or a worklist 576
  receiving subsystem 575
renaming tables 576
sending subsystem 573
stored procedures 45
stored procedures, by importing DML 140
tables with ROWID columns 676
to different subsystems 34
using an inbound migrate profile 577
using an outbound migrate profile 579
using DML 123
workflow 34
MIGSCOPE keyword 60
Mixed List action codes 315
MO (Migrate Options) action code 44
modifying
  applications 669
  DML statements 125, 418, 631
  MVS image variables 737
  task IDs 631
  worklists 80
  worklists and input streams 103
MQT (materialized query table), creating 431
multiple SYSUT data sets 695, 700
multiple tables, moving 406
multitasking, unloading and loading of data 677, 688
MULTITBDS keyword 678
MVS image variables 737

N
name prefixing 319
name template
  generating baseline names 574
  using in baseline profile 350
  using in multistep job 176
naming conventions, DD names 711
native SQL stored procedure, creating 447
native SQL stored procedure, creating a new version 449
NbrRows, Space Estimation parameter 339, 342
NBRSECD UNLOAD PLUS installation option 683
NEWWORKID ALUIN keyword 115, 359
NOALIAS keyword 249
NOAMSDELETEI keyword 705
NOAUTHSW keyword 94
NOAUUXILIARY keyword 249
NOBASELINE ALUIN keyword 217
NOBASELINE keyword 196
NoBaseline option 193
NOBMCFASTLOAD keyword 678, 696
NOBUFFERPOOL keyword 252
NOCHECKCNST keyword 249
NOCOMPRESS keyword 253
NOCOPY keyword 707, 710, 711, 714
NOCOPYFORCE keyword 707, 710, 711, 714
NODATABASE keyword 247, 249
NODEFINE keyword 251
NODROPRESTRICT keyword 252
NOFOREIGNKEY keyword 249
NOFREEPAGE keyword 253
NOHISTORY keyword 717
NOIDENTITY keyword 252
NOINDEX keyword 249
NOKEYCARD ALUIN keyword 717
NOLOCKSIZE keyword 251
NOMTASK keyword 677, 688
NonLf KeyLn, Space Estimation parameter 344
nonpartitioned indexes, PIECESIZE attribute 244
nonpartitioned table spaces
  converting to partition-by-growth table spaces 479, 491
  converting to range-partitioned table spaces 490
  converting to table-controlled partitioned table spaces 486
creating auxiliary objects 538
nonprintable characters 181, 241, 485
nonstandard characters 53
nonviewable characters 53, 181, 241, 485
NOORDERBY keyword 69
NOPARTCOPY keyword 711
NOPARTITION keyword 252
NOPATH keyword 252
NOPCTFREE keyword 253
NOPRIQTY keyword 253
NOREBUILD keyword 706
NOREGENIDENTITY ALUIN keyword 327
NOREGENIDENTITY keyword 328
NOREMARKS keyword 251
NOREORGALT ALUIN keyword 72, 704
NOSECQTY keyword 253
NOSEGSIZE keyword 251
NOSTARTVALUE keyword 252
NOSTATS keyword 716
NOSTOGROUP keyword 249
NOSTOPLIST ALUIN keyword 144
NOSYNONYM keyword 249
NOTABLE keyword 249
NOTABLEACCESS keyword 72
NOTABLEALL keyword 716
NOTABLESPACE keyword 249
NOTRACKMOD keyword 253
NOTRIGGER keyword 249
NOUNIQUECNST keyword 249
NOUNLOADEMPTY keyword 69, 185
NOTILCOPY keyword 710, 711, 714
NOVIEW keyword 249
NOVIEWCHECK keyword 252
NOVVALPROP keyword 69
null characters 53, 181, 241, 485
number formats 176, 629
NUMPARTS attribute 252

0

objects
  baseline, browsing 187
  change flag keywords 249
  comparing 223
  displaying in Compare 244
  synchronizing 223
OBJECTS keyword 190
online Help 23
online reorg 701
online schema changes 701
options
  overriding in Analysis 61
  overriding in Compare 248
  overriding in Execution 88
  overriding in JCL Generation 90
ORDER BY clause 69
outbound migrate profiles
  creating change rules 152
  defining scope 40
  group ID 254
  locations 38
  name resolution in a catalog-to-catalog comparison 247
  scope rules 148
  specifying change rules 48
  used as scope 238
  used by Compare 258
  using change rules 38, 253
  using change rules for multiple production systems 48
  using locations 39, 253
  using scope to select objects 38
OUTPUT descriptors
  naming conventions 714
  using in the COPY PLUS utility 713
  using in the RECOVER PLUS utility 713
  using in the UNLOAD PLUS utility 683
override options
  Analysis, defaults 61, 346
  Execution 88
  JCL Generation 90
OVERRIDE( BASELINE-DELETE ) ALUIN keyword 193, 217

P

padding, changing for an index 397
PARALLEL ALUIN keyword 70, 71
PARALLEL keyword 711, 714
parameters
  Allocation Unit 340
  ALTER 255
  AU (Allocation Unit) 344
  Avg Len 342
parameters (continued)
  Avg Row Length 339
  CDLDEP 256
  Compress 340, 343
  CREATE 255
  DEFER YES 70
  DEFINE NO 69
  DETAIL 263
  Device Type 338
  Devt 342
  DROP 255
  Dssize 340
  Free Page 344
  Freepage 340
  Maxpartitions 340
  Maxrows 340
  NbrRows 339, 342
  NonLf KeyLn 344
  PARM 93
  Pct Compressed 342
  Pct Free 344
  Pctfree 340
  Percent Compressed 338
  Piecesize 343
  PriQty 343
  Rows/Key 342
  SecQty 343
  Secqty 340
  Segsize 339
  Unique 343
  UTILID 110
  WORKDDN 696, 701
PART parameter 93
PART symbolic variable 713
PARTCOPY keyword 711
partition-by-growth table spaces
  converting from index-controlled partitioned table spaces 479
  converting from nonpartitioned table spaces 479
  converting from segmented table spaces 479
  converting from table-controlled partitioned table spaces 479
  converting to index-controlled partitioned table spaces 510
  converting to range-partitioned table spaces 513
  converting to table-controlled partitioned table spaces 512
partitioned table spaces
  converting to nonpartitioned table spaces 493
  converting to partition-by-growth table spaces 507
  converting to range-partitioned 495
  creating auxiliary objects 525, 531
  partition-level image copies, creating 711
partitions
  change flag 252
  converting 429
  VCAT defined 64, 704
PATH attribute 252
Pct Compressed, Space Estimation parameter 342
Pct Free, Space Estimation parameter 344
PCTFREE attribute 49, 253
Pctfree, Space Estimation parameter 340
PCTPRIM UNLOAD PLUS installation option 683
PE (propagate estimates) action code 339, 343
PE (propagate estimates) command 343
FEALL (propagate attributes) command 339
PeopleSoft
  challenges 610
  creating copies of environments 613
  creating environments 611
  managing 609
  modifying application tables 610
  modifying PeopleTools tables 610
  optimizing the design 618
  taking baselines 611
  using DEFINE NO 615
PeopleTools tables, modifying 610
Percent Compressed, Space Estimation parameter 338
performance
  catalog-to-catalog comparison 250
  creating indexes for baselines 197
  creating indexes for comparisons 264
  improving in a baseline 197
  improving in a comparison 264
  periodic baseline 174
  permanent data sets, space required 730
  physical partition numbers 324
  PIC keyword 709
  PIECESIZE attribute 51, 244
  Piecesize, Space Estimation parameter 343
  PKEYPROP keyword 67
  PLAN_TABLE, adding columns 420
  post-Execution baseline 86, 349, 636
  post-Execution comparison 86, 349, 636
  PREALLOC DD 92
  pre-Execution baseline 86, 349, 636
  pre-Execution comparison 86, 349, 636
  preserving limit keys 323
  primary constraints
    creating 384
    including in scope 257
  primary copy, DD name 711
  primary keys
    editing 392
    naming 395
    propagation 67
  primary quantity, setting 129, 470
  PRIQTY attribute 49, 253
  PriQty, Space Estimation parameter 343
  Priqty, Space Estimation parameter 339
PROC statement 93
processes
  change 310
  change migration 572
  data structure recovery 173
  migration 33
  migration within same DB2 subsystem 34
product support 3
profiles
  application, specifying in 661, 668
catalog baseline 175
catalog to catalog comparison 247, 253
change rules 48, 247, 253, 333
DDL baseline 175
inbound migrate 332, 333
locations 39
outbound migrate 38
task ID, specifying in 634
propagating
dropped table columns to indexes 77
name changes 56
PROTECT keyword 193
publications, related 23
punch data sets
dynamically allocating 703
receive-type work ID
  creating 157
  creating in batch 156
  re-creating an existing 157
  used by Execution 90
receiving data structure changes 587, 590
receiving DDL 576
receiving node 89
receiving subsystem 575
receiving worklists 576
RECLnnnn DD name 711
RECOVER PLUS IMPORT command 691
RECOVER PLUS utility
creating partition-level image copies 711
rebuilding indexes 706
restarting 109
unload options 689
using 689
recovering data structures
  establishing a baseline 579
  with current data 602
  with old data 602
recovery baseline with data, creating 203
recovery point
  overview 171
  structure and data 186
  structure-only 174
RECRnnnn DD name 711
REDEFINE NO parameter 695
referential constraint violations, checking 699
REGALL installation option 708
REGENIDENTITY keyword 328
REGISTER keyword 708
related publications 23
rename table
  used by Compare 259, 576
  used in change migration process 576
reorg data sets
dynamically allocating 703
REORG PLUS utility
creating partition-level image copies 711
restarting 109
specifying 700
reorg utilities 705
REORG utility 109, 701
REORGALL ALUIN keyword 72, 699, 704
REORGALT ALUIN keyword 704
REORGALT keyword 72
REORGNONE ALUIN keyword 702
REORGONLINE keyword 702
REORGPENDONLY ALUIN keyword 72, 699, 704
REORGPENDONLY keyword 72, 699, 704
REORGREF ALUIN keyword 702
REPAIR utility 70
REPLACEWORKID keyword 157, 334
replacing a work ID in batch 157
Q
QE (quick edit) action code
  Mixed List panel 44
  moving multiple tables 406
QUIESCE command 736
R
RACF (IBM Resource Access Control Facility), authorizing
  procedures for XIM 733
range-partitioned table spaces
  converting from index-controlled partitioned table spaces 506
  converting from table-controlled partitioned table spaces 498
  converting to index-controlled partitioned table spaces 500
RCV_STRUC_CUR_DATA script 596
RCV_STRUC_OLD_DATA script 602
rebind option 65
REBUILD INDEX utility 706
REBUILD keyword 59
rebuild utilities 705
rebuilding a worklist 59
REC_CHG_SR_BL script 590
REC_CHG_SS script 587
REC_DDL script 411
Index

replacing an existing receive-type work ID 157
REPLICATE_WORKID script
changing column definitions for imported DDL 370
modifying a work ID with values in a user-defined table 366
replicating multiple work IDs 362
REPORT keyword 263
reports
generating a baseline report 190, 212
generating a comparison report 263
REPORTTYPE keyword 190
requirements, using worklist parallelism 740
RESMAXIC keyword 689
restarting
CM/PILOT worklist 633, 642
LINEONEPARGS 109
multistep jobs 352
processing 107
SSID command 110
UTILID 110
utility with the RESTARTPARM keyword 110
utility without the RESTARTPARM keyword 109
worklist 106
RESTARTPARM keyword
including 110
omitting 109
restoring previous versions of data structures 595, 601
RETAINMAX keyword
deleting baselines 194
overriding 195
return codes, Execution 98
REUSE parameter 697
reusing a task ID 637
reviewing and executing data structure changes 349
-REXC command 80
REXX executable 80
REXX macro 80
-RNST command
processing in parallel 717
used in a parallel worklist 741
rotating partitions 324
ROWID columns
adding to a table 520, 544
creating a unique index 522
defined as GENERATED ALWAYS 676
defined as GENERATED BY DEFAULT 676
migrating 676
ROWID GENERATED BY DEFAULT column 326
ROWID SYSREC data set 681, 683
Rows/Key, Space Estimation parameter 342
RPLUSYNC installation option 693
RPLUSYNCAUTO ALUIN keyword 693
RPLUSYNCREPLACE ALUIN keyword 693
rules
change 333
scope 178
RUNSTATS utility
using 716
using in a parallel worklist 741
S
SBCS (single byte character set) 53
scope
catalog-to-baseline comparison 238
catalog-to-DDL comparison 238
catalog-to-worklist comparison 238
DDL 240
defining 40, 236
migrate profile 239
MIGSCOPE keyword 60
specifying a table owner 244
specifying information included in a comparison report 263
specifying locations 245
using in outbound migrate profile 38
SCOPE keyword 237, 239
scope rules
auxiliary object 41, 240
catalog baseline profile 179, 199
Exclude option 148
Include option 40
outbound migrate profile 148
overview 178
used in a baseline 178
used in a comparison 238
scope type
baseline profile 238
DDL 240
migrate profile 239
worklist 240
SCOPERULE keyword 237, 245
SCOPETYPE keyword 237, 239, 240
SCPYLOCB descriptor 714
SCPYLOCP descriptor 714
SCPYREMB descriptor 714
SCPYREMP descriptor 714
script
baselines, specifying 634
browsing 662
changing data structures using a DML trigger 415
changing data structures using an existing work ID 408
changing step settings 634
CMP_STRUCTURE_CHG 300
comments, adding 648
creating a Baseline report step 657
creating a Baseline step 650
creating a Compare step 651
creating a full-recovery baseline 207
creating a work ID step 658
creating an Analysis step 655
creating an Execution step 656
creating an Import step 653
data sets for CHANGE MANAGER 635
DELETE_WORKID 373, 376
deleting 663
deleting sync table entries 376
deleting work IDs 373
editing 660
existing CHANGE MANAGER objects required 646
listing 659
listing steps for a script 661
listing task IDs for a script 665
maintaining 646
overview 646
profiles 634
RCV_Struct_Cur_Data 596
RCV_Struct_Old_Data 602
RECV_Chg_SR_BL 590
RECV_Chg_SS 587
RECV_DDL 411
receiving data structure changes (strong receiver) 590
receiving data structures (strong sender) 587
receiving DDL to create data structure 411
receiving data structure with current data 596
receiving data structure with old data 602
REPLICATE_WORKID 362, 366, 370
scope for a comparison, specifying 634
selecting 646
selecting a script 646
short name 646
specifying information for baselines 634
specifying scope for a comparison 634
steps, changing settings 634
WKID_STRUCTURE_CHG 408
work ID types created 629
secondary quantity, setting 129, 131, 470, 471
SecQTY attribute 49, 253
SecQty, Space Estimation parameter 343
Secqty, Space Estimation parameter 340
security
specifying for a task ID 631
specifying for an application 661
using XIM 733
segmented table spaces, converting to partition-by-growth
table spaces 479
SECSIZE attribute 49, 251
Segsize, Space Estimation parameter 339
sending node 89
sending subsystem 573
SENDONLY keyword 89
SEQI keyword 60, 61
-SETP command 102
-SETS command 94, 102, 182
setting a secondary quantity in a table partition 471
share level on image copies 686, 690
SHOWRULES keyword 257
SHRLEVEL CHANGE 686, 690
SHRLEVEL CHANGE HSSC process, creating objects 473
SHRLEVEL CHANGE online reorg 702
SHRLEVEL NONE option 73
SHRLEVEL NONE reorg 702
SHRLEVEL on image copies 686, 690
SHRLEVEL REFERENCE 686, 690
SHRLEVEL REFERENCE HSSC process, modifying and
migrating objects 476
SHRLEVELCHG keyword 686, 690
SHRLEVELREF keyword 686, 690
SHUTDOWN command 736
shutting down XIM 736
single byte character set 53
SINGLEPHASE keyword 71
skeleton library compiler. See SLIB (skeleton library)
compiler
SLIB (skeleton library) compiler
changing an ISPF skeleton 721
compiling 720, 723
ISPF file tailoring 722
processing 723
runtime report 723
runtime report summary 725
runtime unit 723
testing changes 722
SMS (Storage Management Subsystem) 64, 730
sorting work IDs 359
sortout data sets
dynamically allocating 703
Space Estimation
BMCSPACE keyword 471
calculation of levels 341
calculation of pages 337
compression percentage 337
estimating primary quantities 470
estimating secondary quantities 470
estimating space requirements 453
index 341
overview 336
setting parameters 467
table space 337
Specification
Auxiliary Objects List panel 317
blank values 52
CD Tables, using 41
copied fields 327
data structure changes 345
default values 51
detail panels 326
Mixed List panel 41
new values based on a current values 50
specifying a blank value for an attribute 52
specifying a current value for an attribute 50
-SQL command 63, 102, 182
SQL terminator 181, 243, 330
-SSID command 110, 182
SSID, specifying in EXEC statement 93
STANDALONESTATS keyword 716, 717
START attribute 252
starting a worklist over. See restarting
starting XIM 735
STARTOVER keyword 111
statistics
generating 718
override options 715
utilities 715
statistics, cardinality 717
STATUS command 734
status of XIM, determining 734
status, work ID 100
STOGROUP attribute 252
STOP command 102
stop list, creating 144
STOPCOMMIT keyword 70
STOPLIST ALUIN keyword 144
STOPLIST installation option 144
storage group
specifying default (<DEFLT>) 51
used by Analysis 74
stored procedures
migrating 140
updating options 451
updating parameters 452
stored procedures, migrating 45
strong receiver
overview 577
receiving data structure changes 590
strong sender
overview 578
receiving data structure changes 587
structure recovery points 180
structure-only baseline 180
submitting JCL
for CHANGE MANAGER worklist processing 642
for CM/PILOT worklist processing 635
subsystem, merging 585
support, customer 3
suppress-type change rules 46
SYClmnnn DD name 711
SYCrmnmm DD name 711
symbolic variables, PART 713
SYNC AUTO option 693
-SYNC command 102
sync point frequency, override option 62
sync records
conditions for writing 102
deleting for a work ID 102
error 102
overview 84
SYNC REPLACE option 693
sync table
columns 101
deleting records 102
entries, deleting 376
maintaining 360, 373, 641
overview 103
used by Execution 101, 107
viewing task ID entries 639
SYNCAUTH column 101
SYNCCMD column 101
SYNCDATA column 101
SYNCEDELETE keyword 102
SYNCGLID column 101
SYNCSEQ column 101
SYNCSETP column 101
SYNCSETS column 101
synonyms
applying change rules 48
creating 428
syntax format for diagrams 26
syntax statement conventions 25
SYSCOPY data set, dynamically allocating 712
SYSnmmnn data set 106
SYSREC data sets
dynamically allocating 682, 687
LOB 681, 683
ROWID 681, 683
sysrec data sets
dynamically allocating 703
system-period temporal table, creating 435
SYSUDDUMP data set 730
SYSTUT data set 695, 700
sysut data sets
dynamically allocating 703
SYSTUT work data sets 697

T

table space partitions
adding 480
setting primary quantities 470
setting secondary quantities 131, 470, 471
table spaces
adding partitions 480
auxiliary 318
changing multi-table table spaces 550
changing the encoding scheme 518
changing to DEFINE NO 132
changing VCAT-defined 704
creating 548
creating auxiliary objects 525, 531
creating implicitly 325
explicit 327
implicit 326
reorganizing 699
specified with DEFINE NO 557
specifying <DEFLT> as the name 325
updating to DEFINE NO 554
with existing data set 427
table spaces, converting 320
index-controlled partitioned to partition-by-growth 479
index-controlled partitioned to range-partitioned 505, 506
index-controlled partitioned to table-controlled partitioned 500, 504
nonpartitioned to partition-by-growth 479, 491
nonpartitioned to partitioned 480
nonpartitioned to range-partitioned 490
nonpartitioned to table-controlled partitioned 486
partition-by-growth to indexed-controlled partitioned 510
partition-by-growth to range-partitioned 513
partition-by-growth to table-controlled partitioned 512
partitioned to nonpartitioned 493
partitioned to partition-by-growth 507
partitioned to range-partitioned 495
range-partitioned to index-controlled partitioned 500
segmented to partition-by-growth 479
single-table to partition-by-growth 514
table-controlled partitioned to index-controlled partitioned 497
table-controlled partitioned to partition-by-growth 479
table-controlled partitioned to range-partitioned 498
tables in explicit databases to implicit databases 515
TABLEALL keyword 716
table-controlled partitioning
changing LIKEd objects 503
converting from index-controlled partitioned table spaces 504
converting from index-controlled partitioning 322
converting to index-controlled partitioning 497
converting to partition-by-growth table spaces 479
converting to range-partitioned table spaces 498
migrating to range partitioning 134
preserving limit keys 323
renaming a table 503
renaming a table space 503
rotating partitions 324
tables
adding a column 402
adding a LOB column 521
adding a ROWID column 520, 544
adding WITH RESTRICT ON DROP 426
auxiliary 318
baseline 187
creating a base table 519
creating a primary constraint 384
creating a unique constraint 386
dropping a column 404
moving 552
moving multiple 406
propagating dropped columns to indexes 77
space estimation 427, 548, 557
tapes
data sets, LOADPLUS utility 697
data sets, used in restart processing 108
stacking 677, 688
task IDs
browsing 640
browsing DML statements 641
changing work ID when executing 632
CM/PILOT worklist failure 632, 633, 642, 644
CM/PILOT worklist processing options 633
comments, adding 630
comments, changing 631
creating 628
deleting 641
editing 631
editing CM/PILOT worklists 635
executing 632, 635
execution status, viewing 639
listing 630
listing for a script 665
maintaining 627
modifying 631
options, processing 633, 635
overriding CHANGE MANAGER options 634
overview 627
profiles, specifying 634
restart processing for a CM/PILOT worklist 642
restarting CM/PILOT worklist execution 633
reusing 637
starting CM/PILOT worklist execution over 633, 644
status for restart processing of CM/PILOT worklist 643
sync table entries, viewing 636, 639
types of work IDs required 628
viewing information 630, 640
viewing sync table entries 636, 639
viewing the execution status of a task ID 639
technical support 3
TEMPLATE descriptors, naming conventions 714
templates 318
terminators 181, 243, 330
THRESHOLD keyword 197, 265
-TIME command 182
TRACKMOD attribute 253
trigger text
applying change rules to 335
change propagation 76
change rules 335
trigger text validation
by Analysis 75
by Compare 258, 334
troubleshooting, execution of XIM 739
U

U (Undo) action code 44, 316
UNDOAALL command 44, 316
Unicode objects
  loading with the LOAD utility 697
  loading with the LOADPLUS utility 695, 696
  unloading with the BASIC UNLOAD utility 676
  unloading with the UNLOAD PLUS utility 678
unique constraints
  creating 386
  editing 390
  excluding from scope 257
unique index, creating for a ROWID column 522
Unique, Space Estimation parameter 343
uniqueness, changing for an index 397
-UNLB command
  used in a parallel worklist 741
  used with -BMCD command 679
UNLDCOLL keyword 684
-UNLI command
  deleting data sets 687
  unloading LOB data 688
  unloading XML data 688
  using in a full-recovery baseline 184
  using multitasking 688
unload data set
  -BMCD command 189
  dynamically allocating 682, 687
  -JCLP command 189
  shown in baseline report 191
  -UNRC command 189
  used by Baseline 184, 189
  used by Execution 99
UNLOAD PLUS installation options
  NBRSECD 683
  PCTPRIM 683
UNLOAD PLUS utility
  converting data 695
  generating single worklist command 678
  restarting 109
  unloading data into separate data sets 678
  used in a parallel worklist 741
  using with LOB DATA MOVER program 679
unload utilities 674
UNLOAD utility 686
UNLOADCOPY keyword 685
UNLOADCOPYDS keyword 685
unloading data 688
unloading data into separate data sets 678
-UNRC command
  DEFINENO parameter 185
  EMPTY parameter 185
  FILEREF parameter 185
  used by Baseline 189
  used by Execution 189
UOWTRnnn data set 105

UPDATEMAXA YES parameter, BMC LOADPLUS 327
  updating length of inline LOB columns 558
USE (Use) action code 44
user-defined table, applying changes 366
UTILCOPY keyword 704, 710, 711, 714
UTILID parameter 110
utilities
  BASIC UNLOAD 675
  BMCSTATS 109
  BMCTRIG 109
  CHECK DATA 698
  CHECK PLUS 109, 698
  COPY 708
  COPY PLUS 109
  DSN1COPY 587
  LOAD 697
  LOADPLUS 109
  REBUILD INDEX 706
  RECOVER PLUS 109, 689, 706
  REORG 109, 701
  REORG PLUS 109
  REPAIR 70
  restarting 108
  RUNSTATS 716
  starting over 111
  UNLOAD 686
  UNLOAD PLUS 109
  used by Analysis 674
  UTILSTATS keyword 717

V

values
  blank, specifying 52
  default, specifying 51
  hexadecimal 53
  new, specifying based on current values 50
  nonnumeric, matching 51
  nonprintable characters 53
  nonstandard characters 53
  nonviewable characters 53
  numeric, matching 49
  specifying a range 49
  specifying blank 52
  specifying default 51
  specifying nonnumeric 51
  specifying special numeric 51
VALWARN keyword 68
variables
  XCF_GROUP 738
  XIM_GROUP 738
VCAT-defined table space, reorganizing 703
version, establishing 573
VIEW command 636
view text
  applying change rules to 335
  change propagation 76
  change rules 335
view text validation
  by Analysis 75
  by Compare 258, 334
view, creating an MQT from 431
viewing
  change rules 335
  execution status of a task ID 639
  information about a task ID 630
  sync table entries for a task ID 639
VSAM data sets, creating 70
VVALPROP keyword 68

W
-WKID command 98
WKID_MIGRATE script 141
WKID_STRUCTURE_CHG script 408
WKNAME column 101
WKOWNER column 101
work IDs
  alter-type 357, 575
  automatically creating a new work ID in batch 90
  baseline 188
  CDL conversion 263, 297, 329
  change rules 46, 59
  changing data structure using an existing 408
  changing when executing a task ID 632
  comparing contents 329
  converting to CDL 297
  creating 357
  creating a step in a script 658
  creating in batch 157
  creating in batch mode 115, 358
  deleting 360, 373
  deleting multiple work IDs 373
  deleting sync records 102
  deleting sync table entries 376
  execution 157
  generating CDL from 297
  migrate options 59
  migrate worklist 90
  migrate-type 37, 89
  migrating data structure using an existing 141
  modifying 366
  name template 350, 629, 644, 645, 666, 668
  receive-type 157
  replacing in batch 157
  replicating 658
  replicating multiple work IDs 362
  restart processing 643
  sorting 359
  space estimation, setting 427, 548, 557
  specifying for a task ID 632
  status 100
  status for restart processing of CM/PILOT worklist 643
  status update 100
  types for scripts 629
  types required for task IDs 628
  used by Baseline 188
  used by Execution 90, 100
  using in change migration process 575
  using to change data structures 408
WORKDDN parameter 696, 701
workflow
  alter process 311
  baseline process 171
  comparison process 221
  migrate process 34
WORKID keyword 98, 188, 333
WORKIDMERGE keyword 334
working with applications 666
worklist parallelism
  AEXPRINT data set 729
  allocating space 730
  available in Database Administration 727
  cleanup job set for data sets 93
  creating permanent work data sets 92
  JES3 restrictions 732
  preallocation step 92
  processing utility commands 741
  requirements 740
  restarting 112
  stacked tapes 729
  starting over 112
  using in a PeopleSoft environment 727
worklists
  AEXPRINT diagnostic output 637
  alter-type 347
  analysis 78, 347
  baseline 182
  catalog baseline 182
  CM/PILOT 642
  compare 240
  controlling contents 63
  DDL baseline 182
  editing 635
  executing 562
  generating multiple 38
  generation, alter-type worklist 560
  generation, multiple locations 60
  hash verification numbers 78, 347
  including command types 63
  migrate-type 78, 82
  modifying 80
  multiple 111
  parallel processing 728
  processed by Analysis 345
  rebuilding 59
worklists (continued)
   restarting 106
   sequence number increment 60, 81
   sequencing 81
   starting over 111
   starting worklist execution over 644
   tailoring with change rule 48
   used by Analysis 347
   used by Baseline 182
   used by Execution 82, 98
worklist-to-DDL file comparison 228, 280
worklist-to-worklist comparison 228, 280
worklog 103, 353
WRK100n data set 703

X

X'00' 181
XCF_GROUP variable 738
XIM (Cross-System Image Manager)
   controlling execution 733
   determining status 734
   modifying active initiators 737
   shutting down 736
   starting 735
   troubleshooting 739
XIM commands
   ACTIVATE 737
   QUIESCE 736
   SHUTDOWN 736
   STATUS 734
XIM initiators
   activating 737
   inactivating 736
XIM_GROUP variable 738
XIMACM started task 735
XIMPCTSURMN AEXPIN keyword 93
XIMRETRYCOUNT AEXPIN keyword 93
XIMRETRYMINS AEXPIN keyword 92
XIMRETRYSECS AEXPIN keyword 92
XM (Auxiliary Object XML) action code 316
XML data
   loading with IBM LOAD 697
   unloading with BMC UNLOAD PLUS 678
   unloading with IBM UNLOAD 688

Z

ZOOM command 43