ALTER and CHANGE MANAGER®
for DB2®
User Guide

Supporting

Version 10.1 of Administrative Assistant for DB2
Version 10.1 of ALTER for DB2
Version 10.1 of CHANGE MANAGER for DB2
Version 10.1 of Database Administration for DB2

April 2011
Contacting BMC Software

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

United States and Canada

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

Telephone
713 918 8800 or
800 841 2031

Fax
713 918 8000

Outside United States and Canada

Telephone
(01) 713 918 8800

Fax
(01) 713 918 8000

© Copyright 1998-2011 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 UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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 Software Customer Support 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 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 ........................................... 23
Related publications ........................................ 24
Conventions ................................................. 24
Syntax statements ....................................... 26
  Syntax diagrams ....................................... 27

Part 1 Managing objects with ALTER and CHANGE MANAGER 31

Chapter 1 Migrating database environments 33
Before you begin ........................................ 34
Overview ................................................. 35
Migrating data structures ............................... 35
Deciding how to migrate data structures ............. 39
  Using migrate-type work IDs ........................ 39
  Using outbound migrate profiles .................... 40
  Using Data Manipulation Language ............... 41
Defining the scope of a migration .................. 42
  Defining the scope with a profile ................ 42
  Specifying the data structures for a migration 43
Identifying changes to data structures ............... 47
Specifying change rules for a migration .......... 47
  Specifying change rules for a migrate-type work ID 48
  Specifying change rules for an outbound migrate profile 49
Specifying values for change rule attributes ...... 50
Resolving change rules ................................. 55
Specifying migrate options ............................ 58
Analyzing the effects of a migration ............... 59
  Analyzing requests using work IDs ............ 60
  Analyzing requests using migrate profiles .... 61
Selecting the run type .................................. 61
Overriding the default processing options ....... 62
Validity and performance checks .................... 74
Reviewing the results of Analysis ................. 79
Executing changes ....................................... 83
  Specifying options for Execution .............. 83
  Controlling authorizations ....................... 94
Processing a worklist .................................. 98
Reviewing the results of a migration .................................................. 103
Restart methods .............................................................................. 107
Accomplishing your goals. ................................................................. 114
Creating a migrate-type work ID ...................................................... 115
Creating a migrate-type work ID in batch mode ................................. 116
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 ... 119
Specifying the data structures in a migrate-type work ID ................... 120
Migrating the auxiliary objects for a LOB column ............................... 121
Migrating data structures by using DML ........................................... 123
Assigning a unique name by importing DML ................................. 129
Defining primary and secondary quantities by importing DML ........... 130
Setting a secondary quantity to a factor of a primary quantity by importing DML .......................................................... 132
Changing table spaces and indexes to DEFINE NO by importing DML ... 133
Migrating a partitioned table space to a range-partitioned table space by
importing DML .................................................................................. 135
Migrating a single-table table space to a partition-by-growth table space by
importing DML .................................................................................. 137
Migrating tables from an explicit database to an implicit database by
importing DML .................................................................................. 138
Migrating tables that contain inline LOB columns by importing DML .... 139
Migrating data structures by using a migrate-type work ID ................. 141
Creating an outbound migrate profile .............................................. 144
Creating scope rules for an outbound migrate profile ....................... 146
Specifying locations in an outbound migrate profile .......................... 147
Creating change rules for a migrate profile ........................................ 149
Analyzing a migrate-type work ID and generating a worklist .............. 151
Creating a receive-type work ID ....................................................... 154
Executing a migrate-type worklist .................................................... 156
Restarting a worklist .......................................................................... 162
Starting a worklist over ...................................................................... 164
Where to go from here .................................................................... 166

Chapter 2  Taking a snapshot of a database environment ........................ 167
Before you begin .............................................................................. 168
Overview ......................................................................................... 168
Using baselines ............................................................................... 171
Establishing baselines ..................................................................... 171
Determining what type of baseline to establish .................................... 172
Creating baseline profiles ................................................................. 173
Creating structure-only baselines .................................................... 178
Using a DB2 catalog to create a catalog baseline ............................... 178
Using a DDL file to create a DDL baseline ....................................... 179
Using a migrate-type worklist to create a DDL baseline .................... 180
Creating full-recovery baselines ...................................................... 181
<table>
<thead>
<tr>
<th>Accomplishing your goals</th>
<th>349</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating an alter-type work ID</td>
<td>352</td>
</tr>
<tr>
<td>Creating an alter-type work ID in batch mode</td>
<td>353</td>
</tr>
<tr>
<td>Sorting work IDs</td>
<td>354</td>
</tr>
<tr>
<td>Deleting a work ID</td>
<td>355</td>
</tr>
<tr>
<td>Creating an inbound migrate profile in CHANGE MANAGER</td>
<td>356</td>
</tr>
<tr>
<td>Replicating multiple work IDs</td>
<td>357</td>
</tr>
<tr>
<td>Modifying a work ID with values in a user-defined table</td>
<td>361</td>
</tr>
<tr>
<td>Changing column definitions for imported DDL</td>
<td>365</td>
</tr>
<tr>
<td>Deleting multiple work IDs</td>
<td>368</td>
</tr>
<tr>
<td>Deleting sync table entries for multiple work IDs</td>
<td>371</td>
</tr>
<tr>
<td>Importing a CDL, DDL, or DML file in CHANGE MANAGER</td>
<td>374</td>
</tr>
<tr>
<td>Importing a DDL file in ALTER</td>
<td>376</td>
</tr>
<tr>
<td>Specifying changes to data structures</td>
<td>378</td>
</tr>
<tr>
<td>Creating a primary constraint for a table</td>
<td>379</td>
</tr>
<tr>
<td>Creating a unique constraint for a table</td>
<td>381</td>
</tr>
<tr>
<td>Editing the columns for a unique constraint</td>
<td>385</td>
</tr>
<tr>
<td>Editing a primary key</td>
<td>387</td>
</tr>
<tr>
<td>Naming a primary key</td>
<td>390</td>
</tr>
<tr>
<td>Changing the attributes of an index</td>
<td>392</td>
</tr>
<tr>
<td>Creating indexes for ROWID GENERATED ALWAYS columns by importing DML</td>
<td>395</td>
</tr>
<tr>
<td>Adding a column to a table</td>
<td>397</td>
</tr>
<tr>
<td>Dropping a column from a table</td>
<td>399</td>
</tr>
<tr>
<td>Changing the data type and length of a column</td>
<td>400</td>
</tr>
<tr>
<td>Moving multiple tables using Quick Edit</td>
<td>401</td>
</tr>
<tr>
<td>Changing data structures by using an existing work ID</td>
<td>403</td>
</tr>
<tr>
<td>Receiving DDL to create data structures</td>
<td>406</td>
</tr>
<tr>
<td>Changing data structures by using DML</td>
<td>410</td>
</tr>
<tr>
<td>Adding columns to the DB2 PLAN_TABLE table by importing DML</td>
<td>415</td>
</tr>
<tr>
<td>Deleting aliases by importing DML</td>
<td>417</td>
</tr>
<tr>
<td>Copying a column by importing DML</td>
<td>418</td>
</tr>
<tr>
<td>Updating index partitions for a database by importing DML</td>
<td>419</td>
</tr>
<tr>
<td>Adding WITH RESTRICT ON DROP to tables by importing DML</td>
<td>421</td>
</tr>
<tr>
<td>Updating table spaces or indexes with existing data sets by importing DML</td>
<td>422</td>
</tr>
<tr>
<td>Creating synonyms for selected tables by importing DML</td>
<td>423</td>
</tr>
<tr>
<td>Converting VCAT-defined partitions to STOGROUP-defined partitions by importing DML</td>
<td>424</td>
</tr>
<tr>
<td>Creating a materialized query table (MQT)</td>
<td>426</td>
</tr>
<tr>
<td>Estimating the space requirements for a table space</td>
<td>428</td>
</tr>
<tr>
<td>Estimating the space requirements for an index</td>
<td>434</td>
</tr>
<tr>
<td>Estimating space requirements for a table space based on user-specified values</td>
<td>438</td>
</tr>
<tr>
<td>Estimating space requirements for an index based on user-specified values</td>
<td>440</td>
</tr>
<tr>
<td>Setting space estimation parameters by importing DML</td>
<td>441</td>
</tr>
<tr>
<td>Estimating the primary and secondary quantities in table space and index partitions by importing DML</td>
<td>444</td>
</tr>
<tr>
<td>Setting a secondary quantity in table space and index partitions by importing DML</td>
<td>445</td>
</tr>
</tbody>
</table>
Part 2  Managing environments with ALTER and CHANGE MANAGER  543

Chapter 5  Maintaining database environments  545

Before you begin .................................................. 545
Overview .......................................................... 546
Migrating changes to data structures ............................ 546
  Acting as the sending subsystem ............................ 547
  Acting as the receiving subsystem ......................... 549
Maintaining applications ........................................ 551
  Implementing a strategy ...................................... 552
Recovering data structures ....................................... 553
Feeding back changes to data structures ....................... 554
Using data modeling tools ...................................... 555
  Importing DDL from a data modeling tool ................. 556
  Maintaining catalog information ............................ 557
Accomplishing your goals ......................................... 557
  Migrating data structure changes to other subsystems 558
  Merging subsystems by using ALTER ..................... 559
  Receiving data structure changes from a strong sender 561
  Receiving data structure changes from a strong receiver 564
  Restoring data structures with current data .............. 569
  Recovering data structures with current data ............ 570
  Restoring data structures with old data .................. 575
  Recovering data structures with old data ................ 576
Where to go from here ........................................... 581

Chapter 6  Managing your PeopleSoft environment with CHANGE MANAGER  583

Overview .......................................................... 583
Working in a PeopleSoft environment ........................ 584
Meeting the challenges of a PeopleSoft environment ....... 585
  Taking a snapshot of the initial environments .......... 585
  Creating copies of an existing environment ............ 587
  Optimizing the design of an environment ............... 592

Contents  11
Figures

Workflow for the migration process (part 1 of 2) .................................................. 37
List of DB2 data structures .......................................................... 44
Unattached objects ................................................................. 45
Work ID change rule .............................................................. 49
Specifying a range of values ..................................................... 51
Using the maximum value plus one to specify all values ................. 51
Specifying a new value ........................................................... 52
Specifying a default value with a change rule ............................ 53
Specifying a <BLANK> value with a change rule ......................... 54
Subsequent change rules ........................................................ 56
Application of change rules ...................................................... 56
Analysis Worklist Options panel ............................................. 63
Analysis Worklist Commands Options panel—migrate-type work ID 64
Analysis Input Stream Parameters panel—migrate-type Work ID .... 68
Explicit view column list .......................................................... 69
Implicit view column list ......................................................... 70
Analysis Input Stream Parameters panel—alter-type work ID ....... 73
Execution JCL Processing Interface panel—foreground ............... 87
Execution JCL Processing Interface panel—batch ....................... 88
Execution Override Options panel—migrate-type work ID .......... 89
Execution Status panel ........................................................... 100
Auxiliary Objects List panel .................................................... 123
DML for a MIGRATE statement ............................................... 125
DML for assigning a unique name .......................................... 129
DML for setting primary and secondary quantities .................... 130
Example DML for setting SQTY ............................................... 132
DML to migrate table spaces and indexes .................................. 134
DML to migrate table-controlled partitioned table spaces to range-partitioned table spaces .................................................. 135
DML to migrate table-controlled partitioned table spaces to range-partitioned table spaces by using a JOIN statement .................. 135
DML to migrate index-controlled partitioned table spaces to range-partitioned table spaces .................................................. 136
DML to migrate a table space to a partition-by-growth table spaces .. 137
DML to migrate tables from an explicit database to an implicit database .................................................. 138
DML to migrate tables from an explicit database to an implicit database using a JOIN .................................................. 138
DML to migrate tables with CLOB columns ................................ 140
DML to migrate all dependent objects of an inline CLOB column ... 140
Use of the NEWWORKID keyword ......................................... 155
Tables

Mixed List actions ................................................................. 46
Example work ID change rule .................................................. 49
<BLANK> support for attributes ............................................... 54
Ordering of change rules ....................................................... 57
Change rules for storage group and VCAT partition volume lists .......... 57
Execution return codes ......................................................... 99
Execution status fields ......................................................... 101
Sync table columns ............................................................. 102
Conditions for writing sync records ........................................... 102
Output files for worklist parallelism .......................................... 104
Reestablished sync values ..................................................... 108
Restarting utilities .............................................................. 110
Migration tasks ................................................................. 114
Worklist commands ............................................................ 180
Load requirements for full recovery ........................................... 183
Deleting baselines using DELETEAGE ........................................ 192
Deleting baselines using RETAINMAX ....................................... 193
DB2 catalog table indexes ..................................................... 194
Baseline tasks ................................................................. 195
Types of comparisons .......................................................... 231
Scope of a comparison ........................................................ 232
DD Names used in the comparison JCL ....................................... 253
Compare return codes .......................................................... 257
DB2 catalog table indexes ..................................................... 259
Comparison tasks .............................................................. 261
Mixed List actions ............................................................. 310
Change-level indicators ........................................................ 311
Supported types of conversions ............................................... 316
Converting to table-controlled partitioning .................................. 317
Referencing physical and logical partition numbers ......................... 319
Specifying <DEFLT> for the database or table space name .................. 320
Working with implicit table spaces ......................................... 321
Import input rules for CDL and DDL files .................................... 329
Attributes for estimating the table space .................................... 333
Attributes for modifying the table space .................................... 334
Attributes for estimating index space ....................................... 336
Attributes for modifying the index space .................................... 338
Name template characters ..................................................... 346
Comparing a baseline to a catalog ........................................... 346
Alter tasks ................................................................. 349
Maintenance tasks ................................................................. 557
Typical PeopleSoft environments ............................................. 587
Task ID tasks .............................................................................. 601
Work ID types for scripts ......................................................... 602
Status of task ID execution ....................................................... 614
Restarting processing of a CM/PILOT worklist ......................... 617
Startover processing of a CM/PILOT worklist ......................... 619
Script tasks ............................................................................. 621
Application tasks ................................................................. 641
OUTPUT descriptors used by BMC UNLOAD PLUS .................. 653
Requirements for image copies for UNLOAD PLUS ................ 659
TEMPLATE descriptors used by IBM UNLOAD ......................... 662
Utilities used to process Unicode, LOB, and XML data ............. 665
SHRLEVEL options ............................................................... 673
Primary and backup copy naming conventions for objects .......... 682
Naming conventions for OUTPUT descriptors ......................... 684
Naming conventions for TEMPLATE descriptors .................... 684
Runtime report statistics .......................................................... 697
Authorizing XIM procedures .................................................... 704
Worklist parallelism topics ....................................................... 713
About this book

The *ALTER and CHANGE MANAGER for DB2 User Guide* 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 Customer Support website at [http://www.bmc.com/support](http://www.bmc.com/support). Most product shipments also include the books on a documentation CD.

**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](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

The following related publications supplement this book and the online Help:

<table>
<thead>
<tr>
<th>Category</th>
<th>Document</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>installation</td>
<td>Administrative Products for DB2 Installation Guide</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Administrative Assistant for DB2 Installation Guide</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Database Administration for DB2 Installation Guide</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BMC Products and Solutions for DB2 for z/OS Installation Planning Guide</td>
<td></td>
</tr>
<tr>
<td>product use</td>
<td>ALTER and CHANGE MANAGER for DB2 Getting Started Guide</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALTER and CHANGE MANAGER for DB2 Reference Manual</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Administrative Products for DB2 Messages Manual</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DASD MANAGER PLUS for DB2 Reference Manual</td>
<td></td>
</tr>
<tr>
<td>notices</td>
<td>release notes</td>
<td></td>
</tr>
<tr>
<td>online help</td>
<td>ISPF Help panels</td>
<td></td>
</tr>
</tbody>
</table>

Conventions

This book uses the following special conventions:

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

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

  `testsys/instance/fileName`
- 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="image1" alt="Grid" /> <img src="image2" alt="Table" /></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="image3" alt="Barrel" /></td>
<td>represents files that usually reside on DASD, although they might be allocated on a tape unit.</td>
</tr>
<tr>
<td><img src="image4" alt="Circle" /></td>
<td>represents unload data sets that usually are allocated on tape, although they might reside on DASD.</td>
</tr>
<tr>
<td><img src="image5" alt="Rectangle" /> <img src="image6" alt="Diamond" /></td>
<td>represents ALTER or CHANGE MANAGER objects: Work IDs, migrate profiles, and baseline profiles.</td>
</tr>
<tr>
<td><img src="image7" alt="Rectangle" /> <img src="image8" alt="Box" /></td>
<td>represents components, the programs used to perform functions.</td>
</tr>
<tr>
<td><img src="image9" alt="Arrow" /></td>
<td>represents the flow of information or data from one entity to another.</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>
</table>
| 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. | alias  
databaseDirectory  
serverHostName |
| 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]  
[-full, -incremental, -level] (Unix) |
| Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item. | {DBDName | tableName}  
UNLOAD device={disk | tape, fileName | deviceName}  
{-a | -c} (Unix) |
| A vertical bar means that you can choose only one of the listed items. In the example, you would choose either commit or cancel. | {commit | cancel}  
{-commit | -cancel} (Unix) |
| An ellipsis indicates that you can repeat the previous item or items as many times as necessary. | columnName . . . |
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.

```
DELETE tableName FROM alias;
WHERE CURRENT OF cursorName
```
This part presents the following topics:

Chapter 1  
Migrating database environments .................................................. 33

Chapter 2  
Taking a snapshot of a database environment ............................. 167

Chapter 3  
Comparing database environments ........................................... 215

Chapter 4  
Altering data structures in a database environment ..................... 303
This chapter presents the following topics:

Before you begin .................................................. 34
Overview .............................................................. 35
Migrating data structures ........................................ 35
Deciding how to migrate data structures ................. 39
   Using migrate-type work IDs .............................. 39
   Using outbound migrate profiles ......................... 40
   Using Data Manipulation Language ..................... 41
Defining the scope of a migration .......................... 42
   Defining the scope with a profile ....................... 42
   Specifying the data structures for a migration ...... 43
   Identifying changes to data structures ................ 47
Specifying change rules for a migration ................. 47
   Specifying change rules for a migrate-type work ID 48
   Specifying change rules for an outbound migrate profile 49
   Specifying values for change rule attributes ........ 50
   Resolving change rules ..................................... 55
Specifying migrate options .................................. 58
Analyzing the effects of a migration ..................... 59
   Analyzing requests using work IDs ..................... 60
   Analyzing requests using migrate profiles ............ 61
   Selecting the run type ..................................... 61
   Overriding the default processing options .......... 62
   Validity and performance checks ....................... 74
   Reviewing the results of Analysis ...................... 79
Executing changes ................................................. 83
   Specifying options for Execution ....................... 83
   Controlling authorizations ................................ 94
   Processing a worklist ..................................... 98
   Reviewing the results of a migration .................. 103
   Restart methods ............................................ 107
Accomplishing your goals .................................... 114
   Creating a migrate-type work ID ....................... 115
   Creating a migrate-type work ID in batch mode ..... 116
   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 Getting Started Guide.
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.
Figure 1  Workflow for the migration process (part 1 of 2)

1. **Start**
2. **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 change rules in a migrate-type Work ID
   - **Define the scope of the migration by specifying the data structures**
3. **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
  - Execute the worklist
  - Did you execute the worklist for a receive-type Work ID? 
    - yes
    - Stop
    - no
  - no

- Are you migrating data and data structures to a different subsystem? 
  - yes
    - Do the subsystems share DASD? 
      - yes
      - Create a receive-type Work ID on the receiving subsystem
      - no
      - FTP the worklist, unloaded data, or both to the receiving subsystem
    - no
  - no
  - Stop
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 115.
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 144.

- 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 146.

- 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 specify locations in an outbound migrate profile, see page 147.

---

## 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.

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*.
Defining the scope of a migration

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.

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 following behavior occurs:

(\textit{DB2 Version 7}) The query uses an EBCDIC collating sequence to sort data. ALTER and CHANGE MANAGER use the same sequence to display the data on panels and in reports.

(\textit{DB2 Version 8 and later}) 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.

\textbf{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 Getting Started Guide.
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 FI on the Command line and specify a character string on which to search. For example, you can type 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 Z in the 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).

**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>
<thead>
<tr>
<th>Table 1</th>
<th>Mixed List actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Act</td>
<td>Action</td>
</tr>
<tr>
<td>M</td>
<td>Migrate</td>
</tr>
<tr>
<td>MO</td>
<td>Migrate Options</td>
</tr>
<tr>
<td>QE</td>
<td>Quick Edit</td>
</tr>
<tr>
<td>USE</td>
<td>Use</td>
</tr>
<tr>
<td>U</td>
<td>Undo</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.
Identifying changes to data structures

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.

stored procedures can only be migrated. You cannot create, like, edit, or drop stored procedures.

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

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 50).
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.

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>

**TIP**

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

---

Figure 4  Work ID change rule

```
ALUFRULL  ---------------  Change Rules  --------------------------  SCROLL  ------  PAGE  
WORKID . . . . : RDACRJ.MIG01

Type action next to existing rule or overtype existing values.

E=Edit  I=Insert  D=Delete  L=Like  C=Create

Rules 1 to 1 of 1

E=Edit  I=Insert  D=Delete  L=Like  C=Create

More:  >

Act  Opt  Obj  Attribute  Current  Attribute  Value  New  Attribute  Value

***************************************************************************** TOP  ****************************************************************************
C  C  DB  NAME  TEST*  PROD*
***************************************************************************** BOTTOM  ****************************************************************************
```
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.

**TIP**
To create change rules for a migrate profile, see page 149.

---

### 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>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 51 shows an example.

**Figure 6  Using the maximum value plus one to specify all 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>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.

---

**Figure 7** Specifying a new value

<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>TS</td>
<td>PRIQTY</td>
<td>&lt;400</td>
<td>&lt;CURRENT&gt;*1.33</td>
<td></td>
</tr>
</tbody>
</table>

**************************** TOP *****************************  
**************************** BOTTOM *****************************
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 Getting Started Guide.

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.

Figure 8 Specifying a default value with a change rule
Specifying values for change rule attributes

**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>
<thead>
<tr>
<th>Table 3</th>
<th><code>&lt;BLANK&gt;</code> support for attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td><strong>Editproc</strong></td>
</tr>
<tr>
<td>Table space</td>
<td>X</td>
</tr>
<tr>
<td>Table</td>
<td>X</td>
</tr>
<tr>
<td>Global Temporary Table</td>
<td>X</td>
</tr>
<tr>
<td>Index</td>
<td>X</td>
</tr>
<tr>
<td>View</td>
<td>X</td>
</tr>
<tr>
<td>Alias</td>
<td>X</td>
</tr>
<tr>
<td>All objectsa</td>
<td>X</td>
</tr>
</tbody>
</table>

a 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.

**Figure 9** Specifying a `<BLANK>` value with a change rule

```
Act Opt Obj Attribute Current Attribute Value New Attribute Value
******************************************************************************
C TB VALIDPROC <BLANK> CHKDATA
******************************************************************************
```
Resolving change rules

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 Getting Started Guide.

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>T</td>
<td>B</td>
<td>TB COLNAME</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>T</td>
<td>B</td>
<td>TB COLNAME</td>
<td>B</td>
<td>C</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 56 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.

**Figure 11  Application of 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>T</td>
<td>S</td>
<td>SECQTY</td>
<td>&lt;04</td>
<td>&lt;CURRENT&gt;*2.00</td>
</tr>
<tr>
<td>C</td>
<td>T</td>
<td>S</td>
<td>SECQTY</td>
<td>&lt;05</td>
<td>&lt;CURRENT&gt;+10</td>
</tr>
</tbody>
</table>

- Rules are applied only to existing objects. That is, rules are not applied to objects that are created in Specification for a migrate-type work ID.
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 (part 1 of 2)

<table>
<thead>
<tr>
<th>Order</th>
<th>Rule</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 1     | include | applied first by using the original volume list to determine which additional volumes should be included  

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.
Specifying migrate options

**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 in 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.

---

**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 151.
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 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 Getting Started Guide*. 
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 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 the default processing options

Chapter 1 Migrating database environments 63

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

```
ALUFDAN1 ----------------- 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 97.

**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 102.
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.

![Analysis Worklist Commands Options panel—migrate-type work ID](image)

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, and XML data. 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 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 72.

- 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 181.

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 66.

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 INCLUD\(E\)(DATA, SQL), which would have unload, SQL, load, and other specified data utilities and commands. A third worklist, using INCLUD\(E\)(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 INCLUD\(E\)(AMS) on the receiving subsystem
2. Worklist containing INCLUD\(E\)(DATA, SQL) in phase 1 on the sending subsystem
3. Worklist containing INCLUD\(E\)(DATA, SQL) in phase 2 on the receiving subsystem
4. Worklist containing INCLUD\(E\)(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 INCLUD\(E\)(DATA) or INCLUD\(E\)(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.
Figure 14  Analysis Input Stream Parameters panel—migrate-type Work ID

ALUFOAN39 ---------------- Analysis Input Stream Parameters ----------------
Command ===> 

Type information. Then press Enter to continue or PF12 for previous panel.

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
- NOUNLOADEMPTY   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
- SINGLEPHASE     Omit the -STOP command between phases of a migrate worklist

Commands: HELP END PREVIOUS

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

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.
Overriding the default processing options

NOTE
The CISIZE4K keyword overrides the value that you specify for the Variable CISIZE field on the Tablespace Parts List panel or the Auxiliary Tablespace Attributes List panel.

For partitioned table spaces, the product uses a 4 KB CISIZE for each partition of the table space. You cannot select the Create all VSAM data sets with a CISIZE of 4 KB option for only certain partitions in a table space.

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 62.

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**

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

NOTE
Whether or not the Make tables inaccessible during execution option is selected, the SHRLEVEL NONE option is used as the default in a REORG TABLESPACE statement in a -BMCR or -REOR worklist command.

Overriding the default settings for utilities and statistics

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, see Appendix B, “Using utilities with ALTER and CHANGE MANAGER.”

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.
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 69.

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 55.

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.
Validity and performance checks

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 RESTRICT 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 RESTRICT.
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.
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 100.

**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 74. 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

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.

**TIP**

To execute a migrate-type worklist, see page 156.
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 91.

- 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

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 102.)

  For more information about restarting a worklist, see “Restart processing” on page 108.
Specifying options for Execution

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 154.

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 112.

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.
**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 344.

---

**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**
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

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 Getting Started Guide*.

**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**

```
ALUFOEXI -------------- 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```

---

Chapter 1  Migrating database environments  89
Specifying options for Execution

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 154.

**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
- individual data set characteristics
  - primary and secondary quantity
  - threshold values
  - SMS names
Specifying options for Execution

- 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 Getting Started Guide.

**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:

- XIMRETRYMINST 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 XIMRETRYMINST 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 XIMRETRYMINST keyword.

- 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.
Controlling authorizations

- 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.

**NOTE**

BMC recommends that you specify `AUTHSW=N`. 
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 Getting Started Guide.

**NOTE**

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

- 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.

**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 6  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>

**NOTE**

The SQL SELECT statement is not supported by Execution.

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).

**NOTE**

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 707.
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 8  Sync table columns**

<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>SYNSEQ</td>
<td>worklist sequence number</td>
</tr>
<tr>
<td>SYNCAUTH</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 (part 1 of 2)**

<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>This command causes a sync record to be written.</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>This command causes a sync record to be written. SYNCDATA contains the actual SET CURRENT SQLID command text.</td>
</tr>
<tr>
<td>-SETP command</td>
<td>This command sets a path for triggers and views. The command is executed 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>
</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 Getting Started Guide.

### 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.

#### Table 9  Conditions for writing sync records (part 2 of 2)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-SYNC command</td>
<td>This command is an explicit sync request that causes the text on the command, along with the sequence number, to be written to the sync table. The -SYNC command causes an SQL COMMIT. A long object name in the -SYNC command might be truncated. 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>This 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>At the end of the worklist, an end of job (EOJ) sync record is written. The EOJ sync record is recognized by a SYNCCMD of DONE and a SYNCSEQ of 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. You can recognize error sync records by the characters EMSG in column SYNCCMD.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>#</th>
<th>Output file</th>
<th>ddname</th>
<th>Description and use</th>
</tr>
</thead>
</table>
| 1 | diagnostic output file | AEXPRINT   | 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. 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. **Note:** This file also provides detailed output for UOWs that have been restarted and are being processed serially. You can use this file to perform the following tasks:  
  - Verify that execution starts the initiators, sends each UOW to an initiator, completes the UOWs, and ends the initiators.  
  - Review Execution return codes and determine why a job has failed. 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. |
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>
<tr>
<td></td>
<td></td>
<td></td>
<td>The log contains the following columns:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TIME—time at which Execution began</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Because each XIM initiator starts and completes at various times, the sequence of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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>- SEQ—sequence number for the worklist command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- CMND—name of the worklist command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- INIT—number of the XIM initiator that is processing a UOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SYID—subsystem ID</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>- Monitor the status of a worklist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Monitor the status of each XIM initiator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- INITS—the numbers of the active XIM initiators</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If one of the following conditions exists, a hyphen (-) replaces the number of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initiator:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— an initiator is not active at the specified time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— a UOW is pending and a response has not yet been received from an initiator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can monitor the status of each XIM initiator by reviewing the INITS column and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the messages at the end of the log.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- COMMAND TEXT—portion of the text for the worklist statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can manually insert the STATUSINFO NONE keyword in the AEXPIN input stream to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>eliminate this log and all of the AEXSTnnn logs.</td>
</tr>
</tbody>
</table>
### 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>
<tbody>
<tr>
<td>3</td>
<td>worklist execution log for XIM initiators</td>
<td>AEXPRnnn</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>worklist command status log for XIM initiators</td>
<td>AEXSTnnn</td>
<td>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.</td>
</tr>
<tr>
<td>5</td>
<td>system messages and job log for XIM initiators</td>
<td>AEXSYnnn</td>
<td>contains system messages and job information for each XIM initiator. You can use this file to view the allocation of data sets.</td>
</tr>
<tr>
<td>6</td>
<td>tracing log</td>
<td>AEXPTRAC</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>tracing log for XIM initiators</td>
<td>UOWTRnnn</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 108.

**NOTE**

If all of the following conditions exist, Execution automatically restarts executing the work ID:

- Execution updated the status of the work ID to Exec Strt (Execution started).
- An error occurred or Execution encountered a -STOP command.
- You submitted the initial Execution JCL again, instead of submitting restart JCL.
- The AEXIN input stream does not include the RESTART keyword.

During worklist processing, Execution creates permanent work data sets and sets the disposition of the data sets to new, catalog, catalog (DISP=(NEW,CATLG,CATLG)). You might need to change the disposition of or delete the permanent work data sets.

**TIP**

To restart a worklist, see “Restarting a worklist” on page 162.

---

**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>SYSnnnnn</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

- **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. For more information, see “Startover processing” on page 112.

**TIP**
To start a worklist over, see “Starting a worklist over” on page 164.

**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 109
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

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).

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

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

**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 113.

**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 Repeating 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 DASD MANAGER PLUS components:</td>
<td>Rerun</td>
<td>SSID, VCATname, ASUIDOPTSname, WORKID</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>Rerun</td>
<td>SSID, UTILID, NEW/RESTART</td>
</tr>
<tr>
<td>BMC LOADPLUS</td>
<td>Rerun</td>
<td>SSID, UTILID, NEW/RESTART</td>
</tr>
<tr>
<td>BMC RECOVER PLUS</td>
<td>Rerun</td>
<td>SSID, UTILID, NEW/RESTART</td>
</tr>
<tr>
<td>BMC UNLOAD PLUS</td>
<td>Rerun</td>
<td>SSID, UTILID, NEW/RESTART</td>
</tr>
<tr>
<td>BMC DASD MANAGER PLUS components:</td>
<td>Rerun</td>
<td>SSID, VCATname, ASUIDOPTSname, WORKID</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>IBM REORG</td>
<td>Restart</td>
<td>SSID, UTILID, Restart</td>
</tr>
</tbody>
</table>

- 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.

- 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.

- For other IBM utilities, except IBM REORG, Execution issues only the command to terminate execution of the utility before rerunning the utility.
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:

```
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 `SSID,UTILID,NEW /RESTART` parameters

**NOTE**
Execution starts the COPY PLUS utility with the `SSID,UTILID,NEW /RESET` parameters, and the CHECK PLUS utility with the `SSID,UTILID,NEW` parameters.

**NOTE**
If the BMC or IBM utility encounters an existing UTILID or aUTILID 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 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 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 113.
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.

- Restarting a worklist produces a sort processing log. The ddname of the log is SYSnnn, where nnnn is the number of the message DDN. You can review messages in the log if the restart job abends.

- 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).

- When you enable worklist parallelism for the Database Administration solution and run your job in parallel, use the following guidelines:
  - 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.
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 115</td>
</tr>
<tr>
<td>“Creating a migrate-type work ID in batch mode”</td>
<td>page 116</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 119</td>
</tr>
<tr>
<td>“Specifying the data structures in a migrate-type work ID”</td>
<td>page 120</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 129</td>
</tr>
<tr>
<td>“Defining primary and secondary quantities by importing DML”</td>
<td>page 130</td>
</tr>
<tr>
<td>“Setting a secondary quantity to a factor of a primary quantity by importing DML”</td>
<td>page 132</td>
</tr>
<tr>
<td>“Changing table spaces and indexes to DEFINE NO by importing DML”</td>
<td>page 133</td>
</tr>
<tr>
<td>“Migrating a partitioned table space to a range-partitioned table space by importing DML”</td>
<td>page 135</td>
</tr>
<tr>
<td>“Migrating a single-table table space to a partition-by-growth table space by importing DML”</td>
<td>page 137</td>
</tr>
<tr>
<td>“Migrating tables from an explicit database to an implicit database by importing DML”</td>
<td>page 138</td>
</tr>
<tr>
<td>“Migrating tables that contain inline LOB columns by importing DML”</td>
<td>page 139</td>
</tr>
<tr>
<td>“Migrating data structures by using a migrate-type work ID”</td>
<td>page 141</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.

   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.
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 144 and page 146).

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 151).

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 115.

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

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 115.

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.

D Type the Current Attribute Value (or name) of the data structure that you want to change.

E Type the New Attribute Value (or name) of the data structure that you want to change.

F *(optional)* Type the first and second 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.

7 Press END.

The Change Rules panel is displayed.

8 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.

9 When you have finished specifying the change rules, press END until the WORKID Action Menu is 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 115.

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.

**TIP**
To create a migrate-type work ID, see page 115.

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.

   Any dependent object types will be included when you specify migrating an object (see “Defining migrate options for a migrate-type work ID” on page 116).

4. Press Enter.

   The Mixed List panel is displayed.

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 46.

**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 <strong>Analyze WORKID and create a worklist.</strong></td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 151</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing a migrate-type worklist” on page 156</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 115.
Migrating the auxiliary objects for a LOB 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** 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**).
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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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 144.

---

**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.

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.”
**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**

   ```sql
   MIGRATE DATABASE
   INCLUDE DATA, TABLES, FOREIGNKEYS, INDEXES, VIEWS, TRIGGERS, TABLESPACES, AUTHORIZATIONS, SYNONYMS, Aliases, CHECKS, CONSTRAINTS, AUXILIARY
   SET
       NAME = 'DEMOCJ'
       WHERE
       NAME = 'DEMOBJ'
   ;
   ```
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.

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 616.

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 Getting Started Guide.

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.

---

**TIP**
To execute a worklist, see page 156.
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,ISO),6,2)
  CONCAT 'D' CONCAT
  SUBSTR(CHAR(CURRENT DATE,ISO),9,2)
WHERE
  NAME = 'ALUQATBS'
;
MIGRATE TABLESPACE
INCLUDE ALL
SET
  NAME = 'TSM' CONCAT
  SUBSTR(CHAR(CURRENT DATE,ISO),6,2)
  CONCAT SUBSTR(NAME,6,3)
WHERE
  DBNAME = 'ALUQATBS'
;
```

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 DBMmmDdd, where *mm* is the month, and *dd* is the day of the month. Table spaces resolve to TSMmmnnn, 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 374.

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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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.

Figure 25  DML for setting primary and secondary quantities (part 1 of 2)

```
MIGRATE TABLEPART
JOIN DEM.JOINPARTS
SET
   PQTY       = PRI_QTY,
   SQTY       = SEC_QTY
WHERE
   DBNAME     = QUALIFIER AND
```
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 374.

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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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

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*.

---

**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 374.

**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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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*.

---

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

```
AND IX.DBNAME LIKE 'CRJ%
```
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

```
MIGRATE TABLESPACE
  INCLUDE DATA, TABLES, FOREIGNKEYS, INDEXES, VIEWS,
  AUTHORIZATIONS, SYNONYMS, ALIASES, CHECKS, TRIGGERS
  SET
    DEFINE = 'N'
  WHERE
    NAME = 'TSNAME'  --specify existing TS name
    AND  DBNAME = 'DBNAME' --specify existing DB name
;
MIGRATE 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 374.

**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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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

```
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)

```
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 by using a JOIN statement (part 2 of 2)

```sql
SYSIBM.SYSTABLESPACE.DBNAME LIKE 'DEMO%'  --specify existing DB name
AND TBLS.PARTKEYCOLNUM > 0  --indicates table-controlled partitioning
AND TBLS.DBNAME = SYSIBM.SYSTABLESPACE.DBNAME
AND TBLS.TSNAME = SYSIBM.SYSTABLESPACE.NAME
;
```

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

**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 151</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing a migrate-type worklist” on page 156</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 374.

**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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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
;
```
2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 151</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis</strong>.</td>
<td>“Executing a migrate-type worklist” on page 156</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

Figure 34  DML to migrate tables with CLOB columns

| MIGRATE COLUMNS | -- migrates all tables with LOB columns that match the results of the WHERE clause |
| SET | LENGTH = 104 | -- sets length for an inline LOB to 4 plus the inline length (in bytes) |
| WHERE | COLTYPE IN ('CLOB') AND TBCREATOR = 'COAIN1' ; |

Figure 35 illustrates how to migrate the database, table space, table, and all dependent objects for the table when the table includes an inline CLOB column.

Figure 35  DML to migrate all dependent objects of an inline CLOB column

| MIGRATE COLUMNS | INCLUDE PARENTS | -- includes the database, table space, table, and all dependent objects for the table |
| SET | LENGTH = 104 | -- sets length for an inline LOB to 4 plus the inline length (in bytes) |
| WHERE | LENGTH > 4 AND COLTYPE IN ('CLOB') AND TBCREATOR = 'C2BIN1' ; |

2 Import the DML file.

For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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 115.

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 144.

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 616.

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.
Migrating data structures by using a migrate-type work ID

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 Getting Started Guide.

---

**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.

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.
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.

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 156.
C (optional) To specify scope rules for the migrate profile, select Scope Rules.

NOTE
If you specify scope rules in a migrate profile, you do not need to specify a baseline profile reference.

D (optional) To specify the locations for the migrate profile, select Outbound Locations.

TIP
To specify locations, see page 147.

E (optional) To specify change rules for the migrate profile, select Change Rules.

TIP
To create change rules, see page 149.

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.
Creating scope rules for an outbound migrate profile

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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing a migrate-type work ID and generating a worklist” on page 151</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing a migrate-type worklist” on page 156</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 144.

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 in the Act column and press Enter.

The Edit Scope Rule panel is displayed.
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.

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.

NOTE
You cannot select the Exclude option in the Edit Scope Rule panel for an auxiliary table space, table, or index.

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.

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.
Specifying locations in an outbound migrate profile

TIP
To create an outbound migrate profile, see page 144.

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.

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.
**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 144. To create an inbound migrate profile, see page 356.

---

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.

---

**TIP**

To specify change rules for a location, see page 149.

---

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.

---

7. When you have finished specifying the locations, press **END**.

   The Edit Migrate Profile panel is displayed.

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

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.

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 50.

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 50.

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 133.

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 62.

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 view or modify the worklist, type S to select Edit Worklist.

**NOTE**

If you specified an outbound migrate profile that uses locations, the Edit Worklist option is not displayed.

F Press Enter.

7 Press END.

The WORKID Action Menu is displayed.

**Where to go from here**

After you analyze a migrate-type work ID and generate the worklist, select Execute WORKID worklist created by Analysis to execute the worklist.

**TIP**

To execute a worklist, see page 156.
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.

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` (where `a.b` is the new work ID that you want to create) keyword into the AEXIN data input stream, as shown in Figure 36. The work ID must not currently exist.

Figure 36 Use of the NEWWORKID keyword

```plaintext
//AEXIN DD *
ACM
SSID DEBA NEWWORKID RDAMAF.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 602.

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 37.

Figure 37 Use of the REPLACEWORKID keyword

```plaintext
//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.
Executing a migrate-type worklist

**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 602.

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 156.

---

**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 158 or “Migrating data only or data structures and data” on page 159.

**Before you begin**

Before you can execute a worklist, you must analyze the work ID.

**TIP**

To analyze a work ID, see page 151.
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 539.

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.
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 83.

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.

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 154.

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 162.
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 160.

- Otherwise, see “If data set sizing is required” on page 161.

---

**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.

---

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 154.

   **NOTE**

   Ensure that the following symbolic variables are not included in your data set prefixes:

   - &DB
   - &TS
   - &OBNOD
   - &SSID

   Use the &MSSID symbolic variable in the unload data set prefix.

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 Getting Started Guide*.

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 154.

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 162.

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.
Restarting a worklist

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.

NOTE
When you submit the new execution JCL, you will be submitting the JCL for phase 2 of the worklist.

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 107.

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.
Restarting a worklist

Chapter 1  Migrating database environments 163

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.

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.
Starting a worklist over

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 107.

**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.

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**.
Press Enter.

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>
This chapter presents the following topics:

Before you begin ................................................................. 168
Overview ........................................................................ 168
Using baselines ................................................................. 171
  Establishing baselines ..................................................... 171
  Determining what type of baseline to establish ............... 172
Creating baseline profiles .................................................. 173
Creating structure-only baselines ..................................... 178
  Using a DB2 catalog to create a catalog baseline .......... 178
  Using a DDL file to create a DDL baseline .................... 179
  Using a migrate-type worklist to create a DDL baseline ... 180
Creating full-recovery baselines ......................................... 181
Generating a baseline ....................................................... 184
  Using the baseline table information ............................. 184
  Baseline diagnostic output ............................................. 187
Generating a baseline report ............................................. 187
  Baseline report diagnostic output ................................. 189
Managing baselines ......................................................... 190
  Protecting baselines from deletion ............................... 190
  Deleting baselines ...................................................... 191
Improving the performance of a baseline ......................... 194
Accomplishing your goals ................................................ 195
  Creating a catalog baseline profile .............................. 196
  Creating scope rules for a catalog baseline profile ........ 197
  Creating a structure-only catalog baseline ................... 199
  Creating a full-recovery baseline ................................. 201
  Using a script to create a full-recovery baseline .......... 205
Browsing a baseline ......................................................... 208
Generating a baseline report ........................................... 209
Deleting baselines automatically ..................................... 210
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 38).
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.”

Figure 39 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 39  Workflow for the baseline process

1. Start
2. Create a baseline profile
   - Do you want to create a full-recovery baseline for fallback? (Yes/No)
     - Yes: Go to alter process
     - No: Create a baseline

3. Create a baseline
   - Do you want a report of the baseline? (Yes/No)
     - Yes: Generate baseline report
     - No: Go to comparison process

4. Do you want to perform a comparison? (Yes/No)
   - Yes: Go to comparison process
   - No: Do you want to alter the data structures? (Yes/No)
     - 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.
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 196.

---

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 190.
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 190.

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 190. For more information about the baseline tables in CHANGE MANAGER, see the ALTER and CHANGE MANAGER for DB2 Getting Started Guide.

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 40). 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.
Creating baseline profiles

Chapter 2 Taking a snapshot of a database environment

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 197.
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. To include dependent objects in the baseline, the scope rules must specify the parent object. For example, to include a table in the baseline, the scope rule must specify the table space.

TIP

To create a catalog baseline, see page 199.
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 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 (_). The --#SET TERMINATOR statement is the convention that is used by the IBM DSNTEP2 sample dynamic SQL program and SPUFI.

- 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 Getting Started Guide.

A sample DDL file is shown in Figure 41.
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 42 shows the ALUIN input stream for Analysis to request a full-recovery baseline.
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.

---

**Figure 42  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>

---
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 15  Load requirements for 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.

The baseline is established when the Execution component processes the worklist.
Generating a baseline

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 201 or “Using a script to create a full-recovery baseline” on page 205.

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 43) 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.

![Figure 43 Browse Baseline panel](image)

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.
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_UNLOADDDSN 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 44 shows an example of the ALUIN keywords that are used to create a baseline report.

**Figure 44** ALUIN input stream for a baseline report

```
SSID DEBA
REPORTTYPE BASELINE
OBJECTS (RDACRJ.BLO2)
```
Generating a baseline report

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)

■ 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 209.

--- 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.

**TIP**
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.

When you create a baseline and designate it as protected, the PROTECT keyword is placed in the ALUIN input stream.

**NOTE**
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

**NOTE**

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.
Deleting baselines

**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 02/28/2011. The baselines that are deleted are shaded.

<table>
<thead>
<tr>
<th>Baseline Name</th>
<th>Date Created</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL1</td>
<td>02/18/2011</td>
<td>Incomplete</td>
</tr>
<tr>
<td>BL2</td>
<td>02/19/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL3</td>
<td>02/20/2011</td>
<td>Incomplete</td>
</tr>
<tr>
<td>BL4</td>
<td>02/21/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL5</td>
<td>02/22/2011</td>
<td>Incomplete</td>
</tr>
<tr>
<td>BL6</td>
<td>02/23/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL7</td>
<td>02/24/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL8</td>
<td>02/25/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL9</td>
<td>02/26/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL10</td>
<td>02/27/2011</td>
<td>Complete</td>
</tr>
</tbody>
</table>

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 02/28/2011. 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.

### Table 17  Deleting baselines using RETAINMAX

<table>
<thead>
<tr>
<th>Baseline Name</th>
<th>Date Created</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL1A</td>
<td>02/19/2011</td>
<td>Incomplete</td>
</tr>
<tr>
<td>BL2A</td>
<td>02/20/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL3A</td>
<td>02/21/2011</td>
<td>Incomplete</td>
</tr>
<tr>
<td>BL4A</td>
<td>02/22/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL5A</td>
<td>02/23/2011</td>
<td>Incomplete</td>
</tr>
<tr>
<td>BL6A</td>
<td>02/24/2011</td>
<td>Complete</td>
</tr>
<tr>
<td>BL7A</td>
<td>02/25/2011</td>
<td>Complete</td>
</tr>
</tbody>
</table>

### 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.

**TIP**

To delete baselines automatically, see page 210.
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.

TIP
To delete baselines explicitly, see page 213.

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 18  DB2 catalog table indexes

<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 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 (part 1 of 2)**

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating a catalog baseline profile”</td>
<td>page 196</td>
</tr>
<tr>
<td>“Creating scope rules for a catalog baseline profile”</td>
<td>page 197</td>
</tr>
<tr>
<td>“Creating a structure-only catalog baseline”</td>
<td>page 199</td>
</tr>
<tr>
<td>“Creating a full-recovery baseline”</td>
<td>page 201</td>
</tr>
<tr>
<td>“Using a script to create a full-recovery baseline”</td>
<td>page 205</td>
</tr>
<tr>
<td>“Browsing a baseline”</td>
<td>page 208</td>
</tr>
<tr>
<td>“Generating a baseline report”</td>
<td>page 209</td>
</tr>
<tr>
<td>“Deleting baselines automatically”</td>
<td>page 210</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.

   **NOTE**

   You cannot specify nonzero values for both Deleteage and Retainmax.

   E. *(optional)* Specify a Comment for the baseline profile.
Creating scope rules for a catalog 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 197.

**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.

**TIP**
To create a catalog baseline profile, see page 196.

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.
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.

Press END.

The Scope Rules panel is displayed.

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.

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.

   **NOTE**

   If you want to list baseline profiles instead, type a wildcard pattern for Baseline Profile, and press Enter. The Baseline Profile Selection List panel is displayed. Type S in the Act column adjacent to the profile that you want to use and press Enter. The Baseline Interface panel is displayed again.

4. On the Baseline JCL Processing Interface panel, if the Baseline Name is not displayed, type the name of the baseline.

   **NOTE**

   An owner and a name can be specified. If you omit the owner, the owner that is specified in the baseline profile is set as the default.
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.

**NOTE**
You cannot specify nonzero values for both **Deleteage** and **Retainmax**.

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.
Creating a full-recovery baseline

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 62.

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 45. Note that a full-recovery baseline has been specified.

Figure 45  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  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  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  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.

12 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 62.

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 196.

---

**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 616.

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 Getting Started Guide.

---

**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 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.
**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.

**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.

3 Press END to return to the Baseline Action Menu.

### 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.
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.
Deleting baselines using a baseline profile

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.

NOTE

A baseline must meet the Deleteage or Retainmax criteria for deletion to be included in the Baseline Deletion List.
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.

   **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.

4. Specify one of the Delete Options and press Enter.
Where to go from here

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**.

Where to go from here

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 ................................................................. 216
Overview ................................................................. 216
Comparing data structures ........................................... 219
  Synchronizing data structures ........................................ 219
  Versioning data structures ........................................... 225
Deciding which type of comparison to perform ................. 231
Defining the scope of a comparison ................................. 232
  Using scope rules as the scope ..................................... 233
  Using the objects in the input data structures as the scope 237
Resolving changes to names ........................................... 241
  Using an outbound migrate profile to override the rules 242
Specifying the options for a comparison .......................... 243
  Specifying the types of objects ................................... 243
  Specifying the object attributes ................................... 245
  Specifying an outbound migrate profile .......................... 248
Overriding the default values for a comparison ................ 249
  Allocating data sets ................................................... 250
  Including statements and dependent objects .................. 250
  Displaying change rules ............................................. 252
  Keeping the original owner name for an object ................ 252
Generating the JCL or the input for the comparison ......... 253
Generating CDL ............................................................. 253
  Validation of object text dependencies .......................... 253
  Management of renamed objects ................................... 254
Results of a comparison .................................................. 256
  Diagnostic log .......................................................... 257
  CDL file ................................................................. 257
Generating a CDL file from an alter-type work ID ............ 258
Applying the CDL .......................................................... 259
Improving the performance of a comparison .................... 259
Accomplishing your goals .............................................. 261
  Comparing a DB2 catalog to a DB2 catalog ..................... 261
  Using a profile to scope a catalog to a catalog comparison 267
  Comparing a DB2 catalog to a DDL file .......................... 271
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.

Figure 47 illustrates the process of comparing your data structures using CHANGE MANAGER.
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 48). 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 48  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 261.

---

**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:

```
SELECT NAME FROM remoteSubsytem.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).

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 254.

**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 49) 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 271.

**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 50) 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.
Figure 50  Comparing a DDL file to a DDL file

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 274.

---

**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 51) 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 278.
Comparing a catalog baseline to a DB2 catalog

A catalog baseline to a DB2 catalog comparison (see Figure 52) 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.

**Figure 52  Comparing a baseline to a DB2 catalog**

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 281.

**Comparing a DB2 catalog to a catalog baseline**

A DB2 catalog to a catalog baseline comparison (see Figure 53) 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 53  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 285.

---

**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 54) 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 288.
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>
<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></td>
</tr>
<tr>
<td>Compare two objects in the same subsystem’s catalog, or across subsystems where DDF is implemented</td>
<td></td>
</tr>
<tr>
<td>Synchronize objects</td>
<td>DB2 Catalog to a migrate-type Worklist</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>DB2 Catalog to a Catalog Baseline (baseline established earlier)</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>Type of comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL Baseline</td>
<td>DDL baseline</td>
<td>Compare uses all of the objects captured in both baselines. The subsystems can be either local or remote.</td>
</tr>
<tr>
<td></td>
<td>Catalog baseline</td>
<td>Compare uses all of the objects captured in both baselines. The subsystems can be either local or remote.</td>
</tr>
<tr>
<td>Catalog Baseline</td>
<td>DDL or migrate-type worklist</td>
<td>Compare uses all of the objects in both the baseline and the DDL file or worklist.</td>
</tr>
<tr>
<td></td>
<td>DB2 catalog</td>
<td>Error This type of comparison cannot be performed.</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>Baseline to Baseline</td>
</tr>
</tbody>
</table>
Chapter 3  Comparing database environments  233

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 comparison:

- baseline profile
- outbound migrate profile
- DDL file or migrate-type worklist

---

**Table 21  Scope of a comparison (part 2 of 2)**

<table>
<thead>
<tr>
<th>Primary input (Compare1)</th>
<th>Secondary input (Compare2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DDL baseline</td>
</tr>
<tr>
<td>DDL or Migrate-Type Worklist</td>
<td>Compare uses all of the objects in both the baseline and the DDL file or worklist.</td>
</tr>
<tr>
<td>DB2 Catalog</td>
<td>Error</td>
</tr>
</tbody>
</table>

---

**Tip**

To use a profile to scope a comparison, see page 267.
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 SCOPE1 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 55 shows the keywords used for comparison of a DDL file to a DB2 catalog.

**Figure 55  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>
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.

The SCOPETYPE and SCOPE\textsubscript{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 56 shows the keywords that are used for a DB2 catalog to a DB2 catalog comparison.

**Figure 56  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 57.

Figure 57 Keywords used for a migrate-type worklist to a DB2 catalog comparison

<table>
<thead>
<tr>
<th>CMPTYPE1 WORKLIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE2 LOCAL</td>
</tr>
<tr>
<td>SCOPETYPE 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 Getting Started Guide.

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.

**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\(n\) and CMPIN\(n\) keywords are used in the ALUIN input stream to indicate the use of a baseline in a comparison. The example in Figure 58 shows the use of these keywords for a baseline to baseline comparison.

**NOTE**

The LOCATION\(n\) keyword will be added if either of the baselines is a remote baseline.
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 233.

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 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 (_). The --#SET TERMINATOR statement is the convention that is used by the IBM DSNTEP2 sample dynamic SQL program.

- 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 --#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 59). 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 233.)

NOTE

You cannot use the scope of a profile when a DB2 catalog and a baseline are used in a comparison.

Figure 59  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 Getting Started Guide*.

**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 SCOPERULE\(n\) 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:

| SCOPERULE1 (MAF8220 DB MAF8D22A ) |
| SCOPERULE2 (DSN8220 DB DSN8D22A ) |

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 60* 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>Figure 60  Keywords used in ALUIN input stream for catalog-to-catalog comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1 LOCAL</td>
</tr>
<tr>
<td>CMPTYPE2 REMOTE</td>
</tr>
<tr>
<td>LOCATION2 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 61 illustrates the input data structures in the comparison.

Figure 61   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 61, 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

**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.

### 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.
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

**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 259.

**TIP**

To create an outbound migrate profile, see page 146.

---

Chapter 3 Comparing database environments 243
Specifying the types of objects

- 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 259.

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 62.
Specifying the object attributes

Figure 62  Compare Object Attributes panel

![Figure 62 Compare Object Attributes panel]

**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 248.

- **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.

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.

**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 63.

Figure 63 Use of ALTER, CREATE, and DROP in the ALUIN input stream

<table>
<thead>
<tr>
<th>CMPDB1 BASELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1 BMC.BLUND201102110001</td>
</tr>
<tr>
<td>CMPD1 LOCAL</td>
</tr>
<tr>
<td>INCLUDE (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 64.

**Figure 64  Use of CDLDEP in the ALUIN input stream**

<table>
<thead>
<tr>
<th>CMPTYPE1 BASELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPIN1 RDACRJ.BL01</td>
</tr>
<tr>
<td>LOCATION1 DEBA</td>
</tr>
<tr>
<td>CMPTYPE2 BASELINE</td>
</tr>
<tr>
<td>CMPIN2 RDACRJ.BL02</td>
</tr>
<tr>
<td>LOCATION2 DEAH</td>
</tr>
<tr>
<td>CHANGERULESIN2 RDACEL.C7LBA1$C7LBD1</td>
</tr>
<tr>
<td>INCLUDE (CDLDEP)</td>
</tr>
<tr>
<td>MIGLOCATIONS (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.

NOTE

When you exclude a unique constraint as a dependency of an object, the product excludes the unique key constraint and the primary key constraint. To include the primary key constraint in the scope, you must manually add the OVERRIDE(KEEP-ALL-PRIMARY) keyword to 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 241.)

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></td>
<td>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.</td>
</tr>
<tr>
<td>ALUIN</td>
<td>the name of the parameter input stream</td>
</tr>
</tbody>
</table>

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)
Management of renamed objects

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 74.

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 65 for a catalog baseline to a DB2 catalog comparison:

**Figure 65  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.BLUNLD201104170001

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.
Generating a CDL file from an alter-type work ID

NOTE
By default, DB2 assigns values to certain attributes of DB2 objects. When CDL is generated by comparing a DDL file to another data structure, Compare uses those system defaults if the attributes are not included in the DDL. In a production environment, however, these defaults might not be acceptable. When you evaluate the CDL statements, you should be aware of situations in which you have modified your system to use values other than DB2 defaults and situations in which Compare attempts to reset the values to the defaults. To make or suppress changes to the data structures, you can apply the change rules or apply suppress-type change rules in an outbound migrate profile.

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.

NOTE
The comparison report information reflects the state of the objects before any change rules that are defined in an outbound migrate profile (with the CDLCHANGERULES keyword) are applied.

You can edit a CDL file with the standard ISPF editor or with any other text editor.

NOTE
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.
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 291.

---

**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 24  DB2 catalog table indexes**

<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.SYSRELXS</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 Administrative Products for DB2 Installation 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 261</td>
</tr>
<tr>
<td>“Using a profile to scope a catalog to a catalog comparison”</td>
<td>page 267</td>
</tr>
<tr>
<td>“Comparing a DB2 catalog to a DDL file”</td>
<td>page 271</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 274</td>
</tr>
<tr>
<td>“Comparing a baseline to a baseline”</td>
<td>page 278</td>
</tr>
<tr>
<td>“Comparing a baseline to a DB2 catalog”</td>
<td>page 281</td>
</tr>
<tr>
<td>“Comparing a DB2 catalog to a baseline”</td>
<td>page 285</td>
</tr>
<tr>
<td>“Comparing a baseline to a DDL file”</td>
<td>page 288</td>
</tr>
<tr>
<td>“Generating a CDL file from a work ID”</td>
<td>page 291</td>
</tr>
<tr>
<td>“Using a script to perform a comparison”</td>
<td>page 294</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 267.

The Compare Catalog to Catalog panel is displayed, as shown in Figure 66.

**Figure 66  Compare Catalog to Catalog panel**

```
ACMFCMPC ---------------- Compare Catalog to Catalog ----------------
Command ===>
CDL will be generated to make Compare1 look like Compare2.
Type the values for Compare1, where the resulting CDL will be applied.
  Object Type ( SG DB TS TB IX VW SY AL ) . . . DB
  Name - part 1 . . . . . . . . . . . . . DEMOCJ
  Name - part 2 . . . . . . . . . . . .
  Display list of objects (Y/N) . . . . . . . N
  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.
  Object Type ( SG DB TS TB IX VW SY AL ) . . . DB
  Name - part 1 . . . . . . . . . . . . . DEMOME
  Name - part 2 . . . . . . . . . . . .
  Display list of objects (Y/N) . . . . . . . N
  Optional table owner (to limit scope) . . . . *
  Optional location name for remote DB2 . . . DBDC
  Optional outbound migrate profile (owner.name) containing change rules to apply BEFORE the Compare (CHANGERULESIN2) . .
```
4 Specify the values for Compare1.

A Type the abbreviation for an object type.

B 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.

C If you used a wildcard for Name - part 1 or Name - part 2, type Y to display a list of objects in a mixed list.

D (optional) To limit the scope of the comparison, type the name of the table owner.

E (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.

5 Specify the values for Compare2.

A Repeat steps 4A on page 263 through 4E.

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.

6 Press Enter.

If you chose to display a list of objects in a mixed list in step 4 or step 5, the Mixed List panel is displayed. Otherwise, the Compare CDL Options panel is displayed (proceed to step 9).

---

**NOTE**

If you chose to display a list of objects in a mixed list in both steps 4 and 5, the Mixed List panel is displayed twice: once for Compare1 and once for Compare2.
7 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.

8 Press Enter.

The Compare CDL Options panel is displayed.

9 On the 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 Type S to generate comparison report information as comments in the CDL file.

D (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).

E Select a Run Type.

Be aware that specifying foreground for large comparisons might make your TSO session unavailable for a lengthy period of time.

F Press Enter.

If you chose to display a list of the object attributes in step 9B on page 264, the Compare Object Attributes panel is displayed. Otherwise, the Compare JCL Processing Interface panel is displayed (proceed to step 12).

10 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 in step 10A, 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 Compare CDL Options panel.
11 Press Enter.

The Compare JCL Processing Interface panel is displayed.

12 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 12C, specify the Sysout Class.

13 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.

14 Press Enter.

The Compare Override Options panel is displayed.

15 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 67 shows an example of the ALUIN input stream.

Press END to return to the Compare JCL Processing Interface panel.

---

**Figure 67  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>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 374.

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 profile for Compare1 or type a wildcard pattern to display a list of profiles.

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 profile for Compare2 or type a wildcard pattern to display a list of profiles.

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 10A, 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.
Using a profile to scope a catalog to a catalog comparison

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.
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.

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 374.

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).

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 a DDL file (part 1 of 2)
Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist

1 On the CHANGE MANAGER Main Menu, select Compare (CDL Build), and press Enter.

The Compare Type Selection panel is displayed.

---

**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.

---

**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 374.
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.

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.
Comparing a DDL file or a migrate-type worklist to a DDL file or a migrate-type worklist

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—DDL file to a DDL file

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>DDL</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>DDL</td>
</tr>
<tr>
<td>INCLUDE (ALTER CREATE DROP )</td>
<td></td>
</tr>
<tr>
<td>NODEFINE</td>
<td></td>
</tr>
</tbody>
</table>

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.

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 374.
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.

   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 for Compare2 or type a wildcard pattern to display a list of baselines.

   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.

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.
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.

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.

Specify the data set names to be used in the comparison.

A. 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.

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.
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 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—baseline to a baseline

<table>
<thead>
<tr>
<th>SSID</th>
<th>DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>BASELINE</td>
</tr>
<tr>
<td>CMPIN1</td>
<td>RDACRJ.BLUNLD200704240001</td>
</tr>
<tr>
<td>LOCATION1</td>
<td>DEBA</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>BASELINE</td>
</tr>
<tr>
<td>CMPIN2</td>
<td>RDACRJ.BLUNLD200704240002</td>
</tr>
<tr>
<td>LOCATION2</td>
<td>DEAH</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>(ALTER CREATE DROP )</td>
</tr>
<tr>
<td>NODEFİNE</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 374.

**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.
Comparing a baseline to a DB2 catalog

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.

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, Priqy, or Secqy.

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.
Comparing a baseline to a DB2 catalog

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 DB2 catalog

<table>
<thead>
<tr>
<th>SSID DEBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1 BASELINE</td>
</tr>
<tr>
<td>CMPIN1 RDACRJ.BLUNLD200704240001</td>
</tr>
<tr>
<td>CMPTYPE2 LOCAL</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.

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 374.
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.
Comparing a DB2 catalog to a baseline

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 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.

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—DB2 catalog to a baseline

PROGRAM SSID DEBA
CMPTYPE1 LOCAL CMPTYPE2 BASELINE
CMPIN2 RDACRJ.BL200704260001 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.

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.
Comparing a baseline to a DDL file

**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 374.

**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.
Comparing a baseline to a DDL file

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.
Comparing a baseline to a DDL file

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 Enter 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—baseline to a DDL file

<table>
<thead>
<tr>
<th>SSID</th>
<th>DBDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPTYPE1</td>
<td>BASELINE</td>
</tr>
<tr>
<td>CMPTYPE2</td>
<td>DDL</td>
</tr>
<tr>
<td>CMIN1</td>
<td>RDACRJ.BL01</td>
</tr>
<tr>
<td>INCLUDE (ALTER CREATE DROP )</td>
<td>NODEFINE</td>
</tr>
</tbody>
</table>
Generating a CDL file from a work ID

You can convert an alter-type work ID to a CDL file. The CDL that results acts as a record of changes to the data structures. You can also use the resulting CDL to transmit those changes to another local subsystem or to transmit inter-version changes to remote subsystems.

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.

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 374.
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.

   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 75** shows an example of the ALUIN input stream.

      Press END to return to the Compare JCL Processing Interface panel.

---

**Figure 75** ALUIN input stream—convert work ID to CDL

```plaintext
SSID DEBA
CMPTYPE1 WORKID
CMPIN1 RDACRJ.ALT01
INCLUDE (ALTER CREATE DROP )
NODEFINE
```
Using a script to perform a comparison

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.

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 374.

**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 238</td>
</tr>
<tr>
<td>CHANGE MANAGER worklist</td>
<td></td>
</tr>
<tr>
<td>DDL baseline</td>
<td>“Using a DDL file to create a DDL baseline” on page 179</td>
</tr>
<tr>
<td>catalog baseline</td>
<td>“Using a DB2 catalog to create a catalog baseline” on page 178</td>
</tr>
<tr>
<td>catalog baseline profile containing scope rules</td>
<td>“Creating a catalog baseline profile” on page 196</td>
</tr>
<tr>
<td>outbound migrate profile containing scope rules, change rules, or locations</td>
<td>“Creating an outbound migrate profile” on page 144</td>
</tr>
<tr>
<td>inbound migrate profile containing change rules</td>
<td>“Creating an inbound migrate profile in CHANGE MANAGER” on page 356</td>
</tr>
<tr>
<td>alter-type work ID</td>
<td>“Creating an alter-type work ID” on page 352</td>
</tr>
</tbody>
</table>

To create a task ID


   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.
Using a script to perform a comparison

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 616.

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 Getting Started Guide.

**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.
The Compare Catalog to Catalog panel is displayed.

8 Specify the values for Compare1.

A Type the abbreviation for an object type.

B 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.

C If you used a wildcard for Name - part 1 or Name - part 2, type Y to display a list of objects in a mixed list.

D (optional) To limit the scope of the comparison, type the name of the table owner.

E (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.

9 Specify the values for Compare2.

A Repeat steps 8A on page 298 through 8E.

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.
10 Press Enter.

If you chose to display a list of objects in a mixed list in step 8 on page 298 or step 9, the Mixed List panel is displayed. Otherwise, the Taskid Compare CDL Options panel is displayed (proceed to step 13).

NOTE

If you chose to display a list of objects in a mixed list in both steps 8 and 9, the Mixed List panel is displayed twice: once for Compare1 and once for Compare2.

11 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.

12 Press Enter.

The Taskid CDL Options panel is displayed.

13 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 13B, the Compare Object Attributes panel is displayed. Otherwise, the TASKID Migrate Profiles panel is displayed (proceed to step 16).

14 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.
15 Press Enter.

The TASKID Migrate Profiles panel is displayed.

16 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.

17 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.

18 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 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 537.

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.”
4 Altering data structures in a database environment

This chapter presents the following topics:

Before you begin ................................................. 306
Overview .................................................................. 306
Altering data structures ........................................... 309
Specifying changes to data structures ......................... 309
  Identifying changes to data structures ....................... 311
  Specifying changes to auxiliary objects for a LOB column 312
  Converting table spaces ........................................ 315
  Considerations for changing data structures ................ 321
Importing files ....................................................... 323
  Deciding what type of file to import ......................... 324
  Deciding the type of work ID to use ......................... 327
  Validity checks .................................................. 328
  Reviewing the results of the import ......................... 330
Estimating space ................................................... 331
  Estimating space for a table space .......................... 332
  Estimating space for an index ............................... 335
Estimating space requirements based on user-specified values 339
Analyzing changes ............................................... 340
  Overriding the default processing options ................. 341
  Validity and performance checks ............................ 342
  Reviewing the results of Analysis ......................... 342
Generating JCL ..................................................... 343
Executing changes ............................................... 344
  Specifying options for Execution ............................ 344
  Controlling authorizations .................................... 347
  Processing a worklist ......................................... 347
  Reviewing the results of your changes .................... 348
Accomplishing your goals ........................................ 349
  Creating an alter-type work ID ............................... 352
  Creating an alter-type work ID in batch mode .......... 353
  Sorting work IDs .............................................. 354
Deleting a work ID .................................................. 355
Creating an inbound migrate profile in CHANGE MANAGER ........ 356
Replicating multiple work IDs ........................................ 357
Modifying a work ID with values in a user-defined table ............. 361
Changing column definitions for imported DDL ...................... 365
Deleting multiple work IDs .......................................... 368
Deleting sync table entries for multiple work IDs .................... 371
Importing a CDL, DDL, or DML file in CHANGE MANAGER ........ 374
Importing a DDL file in ALTER .................................... 376
Specifying changes to data structures ................................. 378
Creating a primary constraint for a table ............................ 379
Creating a unique constraint for a table .............................. 381
Editing the columns for a unique constraint .......................... 385
Editing a primary key ............................................... 387
Naming a primary key ............................................... 390
Changing the attributes of an index .................................. 392
Creating indexes for ROWID GENERATED ALWAYS columns by importing DML ........................................ 395
Adding a column to a table ......................................... 397
Dropping a column from a table .................................... 399
Changing the data type and length of a column ...................... 400
Moving multiple tables using Quick Edit ............................. 401
Changing data structures by using an existing work ID .............. 403
Receiving DDL to create data structures ............................. 406
Changing data structures by using DML ............................. 410
Adding columns to the DB2 PLAN_TABLE table by importing DML ... 415
Deleting aliases by importing DML .................................. 417
Copying a column by importing DML ................................ 418
Updating index partitions for a database by importing DML .......... 419
Adding WITH RESTRICT ON DROP to tables by importing DML .......... 421
Updating table spaces or indexes with existing data sets by importing DML .......................................... 422
Creating synonyms for selected tables by importing DML .......... 423
Converting VCAT-defined partitions to STOGROUP-defined partitions by importing DML ..................................... 424
Creating a materialized query table (MQT) .......................... 426
Estimating the space requirements for a table space ................. 428
Estimating the space requirements for an index ...................... 434
Estimating space requirements for a table space based on user-specified values ........................................... 438
Estimating space requirements for an index based on user-specified values ........................................... 440
Setting space estimation parameters by importing DML ............ 441
Estimating the primary and secondary quantities in table space and index partitions by importing DML ......................... 444
Setting a secondary quantity in table space and index partitions by importing DML ........................................... 445
Creating objects for the SHRLEVEL CHANGE HSSC process ........ 447
Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC process ........................................... 450
Converting table spaces to partition-by-growth table spaces by importing DML . 452
Converting a nonpartitioned table space to an index-controlled partitioned table space . 454
Converting a nonpartitioned table space to a table-controlled partitioned table space . 460
Converting a nonpartitioned table space to a range-partitioned table space . 464
Converting a nonpartitioned table space to a partition-by-growth table space . 465
Converting a partitioned table space to a nonpartitioned table space . 467
Changing a partitioned table space to a range-partitioned table space by importing DML . 469
Converting a table-controlled partitioned table space to an index-controlled partitioned table space . 471
Converting table-controlled partitioned table spaces to range-partitioned table spaces by importing DML . 472
Converting a range-partitioned table space to an index-controlled partitioned table space . 474
Converting an index-controlled partitioned table space to a table-controlled partitioned table space . 475
Converting index-controlled partitioned table spaces to table-controlled partitioned table spaces by importing DML . 478
Converting an index-controlled partitioned table space to a range-partitioned table space . 480
Converting index-controlled partitioned table spaces to range-partitioned table spaces by importing DML . 480
Converting a partitioned table space to a partition-by-growth table space . 482
Converting a partition-by-growth table space to an index-controlled partitioned table space . 485
Converting a partition-by-growth table space to a table-controlled partitioned table space . 487
Converting a partition-by-growth table space to a range-partitioned table space . 488
Changing a single-table table space to a partition-by-growth table space by importing DML . 489
Changing tables in an explicit database to an implicit database by importing DML . 490
Converting explicit databases and implicit table spaces to implicit databases and table spaces by importing DML . 492
Changing the encoding scheme for a table space by importing DML . 493
Adding a ROWID and a LOB column to create a base table . 494
Creating auxiliary objects for a partitioned table space . 500
Creating auxiliary objects for a single partition in a partitioned table space . 506
Liking a base table and creating auxiliary objects for a table space . 511
Creating auxiliary objects for a nonpartitioned table space . 513
Converting a VARCHAR data type to a LOB data type . 518
Creating a new database and new table spaces for tables by importing DML . 523
Changing multi-table table spaces to single-table table spaces by importing DML . 525
Moving all empty tables in a database to one table space by importing DML . 527
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 Getting Started Guide.

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 76 illustrates the alter process.

**Figure 76  Workflow for the alter process (part 1 of 2)**
Figure 76  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 309) or import external files (see page 323). 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 352.

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.

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 77) displays lists of different types of DB2 data structures.
Figure 77  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.
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 43. 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 378.

---

### 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>
<thead>
<tr>
<th>Indicator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Undo</td>
</tr>
<tr>
<td>XM</td>
<td>Auxiliary Object (XML) displays the auxiliary objects for XML columns that are associated with a base table You can only view the name of an auxiliary object for an XML column. You cannot drop, like, or edit the object.</td>
</tr>
</tbody>
</table>

Table 26  Mixed List actions (part 2 of 2)

<table>
<thead>
<tr>
<th>Act</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Undo</td>
<td>reverses the action</td>
</tr>
<tr>
<td>XM</td>
<td>Auxiliary Object (XML)</td>
<td>displays the auxiliary objects for XML columns that are associated with a base table You can only view the name of an auxiliary object for an XML column. You cannot drop, like, or edit the object.</td>
</tr>
</tbody>
</table>
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 78). 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 27  Change-level indicators (part 2 of 2)**

<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 312.</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

Chapter 4 Altering data structures in a database environment

Figure 78 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.

**TIP**

To create auxiliary objects, see page 500 and page 513.

---

**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 <NOT FOUND>. 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 segmented</th>
<th>index-controlled partitioned</th>
<th>table-controlled partitioned</th>
<th>range-partitioned</th>
<th>partition-by-growth</th>
<th>implicit database and table space</th>
<th>explicit database and implicit table space</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonpartitioned simple</td>
<td>Supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nonpartitioned segmented</td>
<td>page 454</td>
<td>page 460</td>
<td>page 464</td>
<td>page 465</td>
<td>page 465</td>
<td>page 465</td>
<td>page 465</td>
</tr>
<tr>
<td>index-controlled partitioned</td>
<td>page 467</td>
<td></td>
<td>page 475</td>
<td>page 480</td>
<td>page 482</td>
<td>page 465</td>
<td>page 465</td>
</tr>
<tr>
<td>explicit database and implicit table space</td>
<td>page 467</td>
<td>page 485</td>
<td>page 487</td>
<td>page 488</td>
<td>page 320</td>
<td>page 320</td>
<td></td>
</tr>
<tr>
<td>implicit database and table space</td>
<td>page 467</td>
<td>page 485</td>
<td>page 487</td>
<td>page 488</td>
<td>page 320</td>
<td>page 320</td>
<td></td>
</tr>
<tr>
<td><em>(DB2 Version 9)</em> explicit database and implicit table space</td>
<td>page 467</td>
<td>page 485</td>
<td>page 487</td>
<td>page 488</td>
<td>page 320</td>
<td>page 320</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 317.

*b* For more information about implicit databases and table spaces, see “Working with implicit objects in partition-by-growth table spaces” on page 320.
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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CREATE TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CREATE TABLE (using table-controlled partitioning)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.
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.

**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.
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>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>
<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><code>CREATE TABLE tableName;</code></td>
<td>(DB2 Version 9) DB2 creates the table in an implicit database (named DSNnnnnn) and an implicit table space. (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><code>CREATE TABLE tableName IN DATABASE 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><code>ERROR</code></td>
<td>You cannot explicitly create a table space in an implicit database in ALTER and CHANGE MANAGER.</td>
</tr>
</tbody>
</table>
Considerations for changing data structures

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. Note: 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 518.

---

**TIP**
For a list of tasks that you can perform to alter data structures, see Table 40 on page 349.

---

### 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.

*TIP*

To import a CDL file, see page 374.

**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.

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**

You can use a character other than the semicolon to separate DDL statements if the --#SET TERMINATOR value control statement is in the DDL file. 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 (_). The --#SET TERMINATOR statement is the convention that is used by the IBM DSNTEP2 sample dynamic SQL program.

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.”

**TIP**

To import a DDL file, see page 376.

### 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
  — ACMDMLL1
  — ACMDMLL2
  — ACMDMLM1
  — ACMDMLT1
  — ACMDMLT2
  — ACMDMLUA
  — ACMDMLUB
  — ACMDMLUC
  — ACMDMLUD
  — ACMDMLUF
  — ACMDMLU1
  — ACMDMLU2
  — ACMDMLU3
  — ACMDMLU4
  — ACMDMLU5
  — ACMDMLU6
  — ACMDMLU7
  — ACMDMLU8
  — ACMDMLU9

- 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 SYNTABLE, DELETE WORKID, REPPLICATE WORKID, and SET OMITUNICODE statements.
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 50.

- 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 374.
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 356.

### 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.

To merge the changes in the CDL or DDL file for the same work ID, choose to replace the changes in the work ID. Then, edit the JCL or the input that is created. Change REPLACEWORKID workID to WORKID workID WORKIDMERGE.

### 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.
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

For more information, see “Validity and performance checks” on page 74.

### Application of change rules to view and trigger text

CHANGE MANAGER applies change rules to view text and trigger text. For more information, see “Resolving change rules” on page 55.

---

### 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>
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 = 0 (successful import)

When you import a DML file, the diagnostic output includes a DML report. For more information about DML reports, see “DML report” on page 611.

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 Administrative Products for DB2 Messages Manual.

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 79.

**Figure 79  Tablespace Estimation panel (nonpartitioned)**

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 estimate the space for a table space, you can modify the values for the fields that are listed in Table 34. 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 34 Attributes for estimating the table space

<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. Valid values are 3340, 3350, 3375, 3380, 3390, or 9345.</td>
</tr>
<tr>
<td>Percent Compressed</td>
<td>the percentage used to reduce the average row length. User-supplied percentages are stored on the CD tables. Percent Compressed is applied to the entire table space. It is not applied to the compressed average row length or estimated space if the average row length that is displayed was determined by the BMCSTATS utility and the table space or table space partition was already compressed. In that case, compression will already be applied to those values. Valid values are 00 through 99. The default is zero.</td>
</tr>
<tr>
<td>Avg Row Length</td>
<td>the average row length for the table. The Avg Orig field indicates the origination or source of the Avg Row Length field. To derive the value of the Avg Row Length, 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 row 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.</td>
</tr>
<tr>
<td>NbrRows</td>
<td>the number of rows in the table.</td>
</tr>
</tbody>
</table>

To modify the data structure definitions, you can edit the values for the fields that are listed in Table 35. 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 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. 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>
<tr>
<td>Secqty</td>
<td>the secondary space (expressed in <strong>Allocation Unit</strong>) 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>
</tbody>
</table>
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 80.

---

**Table 35  Attributes for modifying the table space (part 2 of 2)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 428.
To estimate the space for an index, you can modify the values for the fields that are listed in Table 36. 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 36 Attributes for estimating index space (part 1 of 2)

<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>
</tbody>
</table>
To modify the data structure definitions, you can edit the values for the fields that are listed in Table 37. 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>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Len (Average</td>
<td>for padded indexes, the maximum key length; for nonpadded indexes, the</td>
</tr>
<tr>
<td>Key Length)</td>
<td>average length of the index key</td>
</tr>
<tr>
<td></td>
<td>The Key Orig field indicates the origination or source of the Avg Len field.</td>
</tr>
<tr>
<td></td>
<td>To derive the value of the Avg Len field, the products gather information</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>system catalog (Rnst)</td>
</tr>
<tr>
<td></td>
<td>3. a projected value calculated by the products (Proj)</td>
</tr>
<tr>
<td></td>
<td>If a value for the average key length was saved in the product CD</td>
</tr>
<tr>
<td></td>
<td>tables, space estimation uses that value, unless the value was</td>
</tr>
<tr>
<td></td>
<td>projected. In that case, the products recalculate the value.</td>
</tr>
<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>
### Table 37 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</td>
</tr>
<tr>
<td></td>
<td>Valid values are U (Unique), D (Duplicates), and W (defined as UNIQUE WHERE NOT NULL).</td>
</tr>
<tr>
<td>Piecesize</td>
<td>maximum piecesize for nonpartitioned indexes</td>
</tr>
<tr>
<td></td>
<td>Avoid wasting space by ensuring that the sum of the PriQty and the SecQty values evenly divide into the Piecesize value.</td>
</tr>
<tr>
<td>Note</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>■ If a value is specified for the DSSIZE for the auxiliary table space, Import uses that value for the PIECESIZE.</td>
</tr>
<tr>
<td></td>
<td>■ 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.</td>
</tr>
<tr>
<td>Compress</td>
<td>whether the index uses compression</td>
</tr>
<tr>
<td>PriQty</td>
<td>the primary space (expressed in Aloc Unit) 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 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.</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 PE command-line command propagates the estimated space to the PriQty field for all of the partitions in the index.</td>
</tr>
<tr>
<td></td>
<td>To ensure that you do not waste space, specify a value for the PriQty that does not exceed the value of Piecesize.</td>
</tr>
<tr>
<td>SecQty</td>
<td>the secondary space (expressed in Aloc Unit) allocated for the DB2-defined data set</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
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 81. 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 37 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. 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. 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. 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. 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 434.
Analyzing changes

For a description of each field that accepts user-specified values, see the online Help.

**Figure 81 Tablespace Estimation panel**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pagesize</td>
<td>16 (4,8,16,32)</td>
</tr>
<tr>
<td>Segsize</td>
<td>4 (0-64)</td>
</tr>
<tr>
<td>Maxrows</td>
<td>255 (0-255)</td>
</tr>
<tr>
<td>Dsize</td>
<td>1</td>
</tr>
<tr>
<td>Pctfree</td>
<td>0 (0-99)</td>
</tr>
<tr>
<td>Freepage</td>
<td>0 (0-255)</td>
</tr>
<tr>
<td>Compress</td>
<td>N (Y or blank)</td>
</tr>
<tr>
<td>Priqty</td>
<td>2000</td>
</tr>
<tr>
<td>Secqty</td>
<td>100</td>
</tr>
<tr>
<td>Alloc Unit</td>
<td>T (T or C)</td>
</tr>
<tr>
<td>Device Type</td>
<td>3390</td>
</tr>
<tr>
<td>Extents</td>
<td>1</td>
</tr>
<tr>
<td>Data Sets</td>
<td>0</td>
</tr>
<tr>
<td>Space (Trks)</td>
<td>2</td>
</tr>
<tr>
<td>Space (KB)</td>
<td>96</td>
</tr>
<tr>
<td>Data Sets</td>
<td>0</td>
</tr>
<tr>
<td>Extents</td>
<td>1</td>
</tr>
<tr>
<td>Data Sets</td>
<td>0</td>
</tr>
</tbody>
</table>

Tables: Enter I to Insert, R to Repeat, or D to Delete a line.

<table>
<thead>
<tr>
<th>Avg Row</th>
<th>#Rows</th>
<th>Pages/Page</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**TIP**
To estimate the space requirements based on user-specified values, see page 438 and page 440.

**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.”

For information about overriding the default processing options, see “Overriding the default processing options” on page 62.
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 74.

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.

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

For more information about worklist commands, see the 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.

**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 80.

**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 74. For information about error and warning messages, see the *Administrative Products for DB2 Messages Manual*.

**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 84.
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 537.

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 83.

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 82.

**NOTE**

Because Compare and Baseline are not components in the ALTER product, you cannot create a multi-step job in ALTER.
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 38.
Specifying options for Execution

Table 38  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 YYYYMMDD</td>
</tr>
</tbody>
</table>

Figure 83 shows the use of a name template for a baseline to catalog comparison that uses an alter-type work ID as the primary input.

Table 39  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 539.

### 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

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 40 lists the tasks that you can perform to alter data structures.

Table 40  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 352</td>
</tr>
<tr>
<td>“Creating an alter-type work ID in batch mode”</td>
<td>page 353</td>
</tr>
<tr>
<td>“Sorting work IDs”</td>
<td>page 354</td>
</tr>
<tr>
<td>“Deleting a work ID”</td>
<td>page 355</td>
</tr>
<tr>
<td>“Creating an inbound migrate profile in CHANGE MANAGER”</td>
<td>page 356</td>
</tr>
<tr>
<td>“Replicating multiple work IDs”</td>
<td>page 357</td>
</tr>
<tr>
<td>“Modifying a work ID with values in a user-defined table”</td>
<td>page 361</td>
</tr>
<tr>
<td>“Changing column definitions for imported DDL”</td>
<td>page 365</td>
</tr>
<tr>
<td>“Deleting multiple work IDs”</td>
<td>page 368</td>
</tr>
<tr>
<td>“Deleting sync table entries for multiple work IDs”</td>
<td>page 371</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 374</td>
</tr>
<tr>
<td>“Importing a DDL file in ALTER”</td>
<td>page 376</td>
</tr>
<tr>
<td>Specifying changes to objects</td>
<td></td>
</tr>
<tr>
<td>“Specifying changes to data structures”</td>
<td>page 378</td>
</tr>
<tr>
<td>“Creating a primary constraint for a table”</td>
<td>page 379</td>
</tr>
<tr>
<td>“Creating a unique constraint for a table”</td>
<td>page 381</td>
</tr>
<tr>
<td>“Editing the columns for a unique constraint”</td>
<td>page 385</td>
</tr>
<tr>
<td>“Editing a primary key”</td>
<td>page 387</td>
</tr>
<tr>
<td>“Naming a primary key”</td>
<td>page 390</td>
</tr>
<tr>
<td>“Changing the attributes of an index”</td>
<td>page 392</td>
</tr>
<tr>
<td>“Creating indexes for ROWID GENERATED ALWAYS columns by importing DML”</td>
<td>page 395</td>
</tr>
<tr>
<td>Task</td>
<td>Reference</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>“Adding a column to a table”</td>
<td>page 397</td>
</tr>
<tr>
<td>“Dropping a column from a table”</td>
<td>page 399</td>
</tr>
<tr>
<td>“Changing the data type and length of a column”</td>
<td>page 400</td>
</tr>
<tr>
<td>“Moving multiple tables using Quick Edit”</td>
<td>page 401</td>
</tr>
<tr>
<td>“Changing data structures by using an existing work ID”</td>
<td>page 403</td>
</tr>
<tr>
<td>“Receiving DDL to create data structures”</td>
<td>page 406</td>
</tr>
<tr>
<td>“Changing data structures by using DML”</td>
<td>page 410</td>
</tr>
<tr>
<td>“Adding columns to the DB2 PLAN_TABLE table by importing DML”</td>
<td>page 415</td>
</tr>
<tr>
<td>“Deleting aliases by importing DML”</td>
<td>page 417</td>
</tr>
<tr>
<td>“Copying a column by importing DML”</td>
<td>page 418</td>
</tr>
<tr>
<td>“Updating index partitions for a database by importing DML”</td>
<td>page 419</td>
</tr>
<tr>
<td>“Adding WITH RESTRICT ON DROP to tables by importing DML”</td>
<td>page 421</td>
</tr>
<tr>
<td>“Updating table spaces or indexes with existing data sets by importing DML”</td>
<td>page 422</td>
</tr>
<tr>
<td>“Creating synonyms for selected tables by importing DML”</td>
<td>page 423</td>
</tr>
<tr>
<td>“Converting VCAT-defined partitions to STOGROUP-defined partitions by importing DML”</td>
<td>page 424</td>
</tr>
<tr>
<td>“Creating a materialized query table (MQT)”</td>
<td>page 426</td>
</tr>
</tbody>
</table>

**Changing space for an object**

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Estimating the space requirements for a table space”</td>
<td>page 428</td>
</tr>
<tr>
<td>“Estimating the space requirements for an index”</td>
<td>page 434</td>
</tr>
<tr>
<td>“Estimating space requirements for a table space based on user-specified values”</td>
<td>page 438</td>
</tr>
<tr>
<td>“Estimating space requirements for an index based on user-specified values”</td>
<td>page 440</td>
</tr>
<tr>
<td>“Setting space estimation parameters by importing DML”</td>
<td>page 441</td>
</tr>
<tr>
<td>“Estimating the primary and secondary quantities in table space and index partitions by importing DML”</td>
<td>page 444</td>
</tr>
<tr>
<td>“Setting a secondary quantity in table space and index partitions by importing DML”</td>
<td>page 445</td>
</tr>
</tbody>
</table>

**Working with table spaces**

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating objects for the SHRLEVEL CHANGE HSSC process”</td>
<td>page 447</td>
</tr>
<tr>
<td>“Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC process”</td>
<td>page 450</td>
</tr>
<tr>
<td>“Converting table spaces to partition-by-growth table spaces by importing DML”</td>
<td>page 452</td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to an index-controlled partitioned table space”</td>
<td>page 454</td>
</tr>
</tbody>
</table>
### Table 40  Alter tasks (part 3 of 4)

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Converting a nonpartitioned table space to a table-controlled partitioned table space”</td>
<td>page 460</td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to a range-partitioned table space”</td>
<td>page 464</td>
</tr>
<tr>
<td>“Converting a nonpartitioned table space to a partition-by-growth table space”</td>
<td>page 465</td>
</tr>
<tr>
<td>“Converting a partitioned table space to a nonpartitioned table space”</td>
<td>page 467</td>
</tr>
<tr>
<td>“Changing a partitioned table space to a range-partitioned table space by importing DML”</td>
<td>page 469</td>
</tr>
<tr>
<td>“Converting a table-controlled partitioned table space to an index-controlled partitioned table space”</td>
<td>page 471</td>
</tr>
<tr>
<td>“Converting table-controlled partitioned table spaces to range-partitioned table spaces by importing DML”</td>
<td>page 472</td>
</tr>
<tr>
<td>“Converting a range-partitioned table space to an index-controlled partitioned table space”</td>
<td>page 474</td>
</tr>
<tr>
<td>“Converting an index-controlled partitioned table space to a table-controlled partitioned table space”</td>
<td>page 475</td>
</tr>
<tr>
<td>“Converting index-controlled partitioned table spaces to table-controlled partitioned table spaces by importing DML”</td>
<td>page 478</td>
</tr>
<tr>
<td>“Converting an index-controlled partitioned table space to a range-partitioned table space”</td>
<td>page 480</td>
</tr>
<tr>
<td>“Converting index-controlled partitioned table spaces to range-partitioned table spaces by importing DML”</td>
<td>page 480</td>
</tr>
<tr>
<td>“Converting a partitioned table space to a partition-by-growth table space”</td>
<td>page 482</td>
</tr>
<tr>
<td>“Converting a partition-by-growth table space to an index-controlled partitioned table space”</td>
<td>page 485</td>
</tr>
<tr>
<td>“Converting a partition-by-growth table space to a table-controlled partitioned table space”</td>
<td>page 487</td>
</tr>
<tr>
<td>“Converting a partition-by-growth table space to a range-partitioned table space”</td>
<td>page 488</td>
</tr>
<tr>
<td>“Changing a single-table table space to a partition-by-growth table space by importing DML”</td>
<td>page 489</td>
</tr>
<tr>
<td>“Changing tables in an explicit database to an implicit database by importing DML”</td>
<td>page 490</td>
</tr>
<tr>
<td>“Converting explicit databases and implicit table spaces to implicit databases and table spaces by importing DML”</td>
<td>page 492</td>
</tr>
<tr>
<td>“Changing the encoding scheme for a table space by importing DML”</td>
<td>page 493</td>
</tr>
<tr>
<td><strong>Working with LOBs</strong></td>
<td></td>
</tr>
<tr>
<td>“Adding a ROWID and a LOB column to create a base table”</td>
<td>page 494</td>
</tr>
<tr>
<td>“Creating auxiliary objects for a partitioned table space”</td>
<td>page 500</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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating auxiliary objects for a single partition in a partitioned table space”</td>
<td>page 506</td>
</tr>
<tr>
<td>“Liking a base table and creating auxiliary objects for a table space”</td>
<td>page 511</td>
</tr>
<tr>
<td>“Creating auxiliary objects for a nonpartitioned table space”</td>
<td>page 513</td>
</tr>
<tr>
<td>“Converting a VARCHAR data type to a LOB data type”</td>
<td>page 518</td>
</tr>
<tr>
<td>“Creating a new database and new table spaces for tables by importing DML”</td>
<td>page 523</td>
</tr>
<tr>
<td>“Changing multi-table table spaces to single-table table spaces by importing DML”</td>
<td>page 525</td>
</tr>
<tr>
<td>“Moving all empty tables in a database to one table space by importing DML”</td>
<td>page 527</td>
</tr>
<tr>
<td>“Setting table spaces and indexes to DEFINE NO by importing DML”</td>
<td>page 529</td>
</tr>
<tr>
<td>“Updating table spaces or indexes defined with DEFINE NO by importing DML”</td>
<td>page 532</td>
</tr>
<tr>
<td>“Updating the length of an inline LOB column by importing DML”</td>
<td>page 533</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 535</td>
</tr>
<tr>
<td>“Executing an alter-type worklist”</td>
<td>page 537</td>
</tr>
<tr>
<td>“Generating a multi-step job”</td>
<td>page 539</td>
</tr>
</tbody>
</table>
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 <strong>Specify DB2 definitions</strong>.</td>
<td>“Specifying changes to data structures” on page 378</td>
</tr>
<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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis</strong>.</td>
<td>“Executing an alter-type worklist” on page 537</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 374 or page 376)*.

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 374 or page 376)*.
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.

   C 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 378</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 535</td>
</tr>
<tr>
<td>In the Act column, type X.</td>
<td>“Executing an alter-type worklist” on page 537</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 Getting Started Guide. 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 368.

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.

3. Select Delete WORKID and press Enter.

**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.
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 149.

   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.

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 378</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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>
Replicating multiple work IDs

The CM/PILOT REPLICATE_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 84 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 84  DML for replicating a work ID**

<table>
<thead>
<tr>
<th>REPLICATE WORKID SET</th>
<th>WKOWNER = 'RDACRJ'</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE</td>
<td>WKOWNER = 'DEM'</td>
</tr>
<tr>
<td>;</td>
<td></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.

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 85.
Figure 85  DML for replicating a database

<table>
<thead>
<tr>
<th>CHANGE DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET CDNAME = 'RDAMKE',</td>
</tr>
<tr>
<td>CDOWNER = 'RDACRJ',</td>
</tr>
<tr>
<td>DBNAME = 'DEMOCJ',</td>
</tr>
<tr>
<td>NEW_DBNAME = 'DEMOME'</td>
</tr>
<tr>
<td>WHERE</td>
</tr>
<tr>
<td>CDOWNER = 'DEM'</td>
</tr>
<tr>
<td>AND CDNAME LIKE 'RDAE%'</td>
</tr>
<tr>
<td>AND DBNAME = 'AAX100'</td>
</tr>
<tr>
<td>AND NEW_DBNAME LIKE 'AAMX%'</td>
</tr>
<tr>
<td>AND NEW_BPOOL = 'BP1'</td>
</tr>
<tr>
<td>AND CHANGELVL = 'L'</td>
</tr>
<tr>
<td>AND DBOWNER LIKE 'DEMOC%'</td>
</tr>
<tr>
<td>;</td>
</tr>
</tbody>
</table>

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 616.

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 Getting Started Guide.
Replicating multiple work IDs

**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.

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.
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.

**NOTE**
An Analysis worklist is not created and is not required.

### Modifying a work ID with values in a user-defined table

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.

**TIP**
To replicate multiple work IDs, see page 357.
Modifying a work ID with values in a user-defined table

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 86 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 86  DML for applying changes from a user-defined table**

| REPLICATE WORKID
| SET WKOWNER = 'RDADEM'
| WHERE
| WKOWNER = 'DEM'
| AND WKNAME = 'WRK1620'
| ;
| CHANGE TABLEPART
| JOIN DEM.JOINPARTS
| SET NEW_PQTY = PRI_QTY,
| NEW_SQTY = SEC_QTY
| WHERE
| (DBNAME = QUALIFIER AND
| TSNAME = NAME)
| OR
| (IXOWNER = QUALIFIER AND
| IXNAME = NAME)
| ;

The DML in Figure 86 replicates the old work ID. Any table partitions that match the user-defined table are modified. In this example, DEM.JOINPARTS 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.
3 When you have finished editing the statements, press EN D 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 616.

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 Getting Started Guide.

---

**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.

**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.

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.

--- NOTE ---

An Analysis worklist is not created and is not required.
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.

**TIP**

To replicate multiple work IDs, see page 357.

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**.
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 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*.

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 616.

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 Getting Started Guide.

---

**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.
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 Getting Started Guide.

In this task, you will

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

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.
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 88, if necessary.

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

**NOTE**
You can copy the DML statement from member ACMDMLD2 in the HLQ.DBCNTL data set.

### Figure 88  DML for deleting work IDs

```sql
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 616.

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

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**

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 Getting Started Guide*.

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.
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 89, 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 89  DML for deleting sync table entries**

```
DELETE SYNCTABLE
WHERE
  STATUS = 'C'  -- LIMITS TO STATUS COMPLETE
  AND (WKOWNER LIKE 'ABC%' OR
       WKOWNER LIKE 'DEF%')
  AND DATE(MODDATE) < DATE('2009-11-01')
;```
Deleting 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 616.

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.
Importing a CDL, DDL, or DML file in CHANGE MANAGER

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.

**NOTE**
An Analysis worklist is not created and is not required.

---

**Importing a CDL, DDL, or DML file in CHANGE MANAGER**

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.

---

**TIP**
To create an alter-type work ID, see page 352. To create an inbound migrate profile, see page 356.
Importing a CDL, DDL, or DML file in CHANGE MANAGER

Chapter 4 Altering data structures in a database environment

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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 352.

---

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 352.

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 43.

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 43 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 397.

   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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
Creating a primary constraint for a table

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.

---

**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 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 90.

**Figure 90 Mixed List panel**

```
ALUSMXL ER ------------------------- Mixed List -------------------------
Command ===> Scroll. . CSR
WORKID . . . : RDACRJ.ALT10 Commands: CANCEL

Type action next to object and press Enter.
E=Edit  L=Like  D=Drop  U=Undo

Objects 100 to 113 of 146
More:  - +

Act Object-Type   Objects
TB . . . . DEMOCJ     TBS2A
    IXU . . . DEMOCJ     IXS2A
TS . . . . DEMOCJ     S22
TB . . . . DEMOCJ     TBS22A
    IXP . . . DEMOCJ     IXS22A
TB . . . . DEMOCJ     TBS22B
    IXP . . . DEMOCJ     IXS22B
TB . . . . DEMOCJ     TBS22C
TB . . . . DEMOCJ     TBS22D
    IXP . . . DEMOCJ     IXS22D
TS . . . . DEMOCJ     S23
TB . . . . DEMOCJ     TBS23A
*L UCP . . . DEMOCJ     TBS23A  PKEY1
    IX . . . DEMOCJ     IXS23A
TS . . . . DEMOCJ     S24AAAAA
```
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 91.

**Figure 91  Constraint Key Mixed List panel**

```
ALUSUC2 ER  ------------------- Constraint Key Mixed list -------------------
                         Command ===>                                                    Scroll. PAGE
                         WORKID.... RDACRJ.UC01  Commands: CANCEL
                         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 Nl FP
                         ******************************************* TOP ********************************************
                         A   CONSTRAINT KEY .:
                         PARENT TABLE .: DEMOCJ T0110_REFERRAL
                         C   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
                             CASE_POOL_NUM  SMALLINT  N  N
```
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 92. The column specified for the constraint is displayed below the constraint key.

**Figure 92  New constraint key column**

```
ALUSUC2 ER   --------------------------- Constraint Key Mixed list ---------------------------
Command ===> Scroll . . PAGE
WORKID ...... : RDACRJ.UC01 Commands: CANCEL
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
Lines 1 to 7 of 15
More: +

Act Colname          Column Type    UDT Length NI FP
*******************************************************
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 93.
Creating a unique constraint for a table

**Figure 93  Mixed List panel**

<table>
<thead>
<tr>
<th>ALUSMXL ER</th>
<th>Mixed List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command ===&gt;</td>
<td>Scroll . CSR</td>
</tr>
<tr>
<td>WORKID . . . : RDACRJ.UC01</td>
<td>Commands: CANCEL</td>
</tr>
<tr>
<td>Type action next to object and press Enter.</td>
<td></td>
</tr>
<tr>
<td>E=Edit  L=Like  D=Drop  U=Undo</td>
<td></td>
</tr>
<tr>
<td>Objects 2 to 15 of 146</td>
<td>More: - +</td>
</tr>
<tr>
<td>Act</td>
<td>Object-Type</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>IXU . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>*L</td>
<td>UCU . . .</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>DEMOCJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCJ</td>
</tr>
</tbody>
</table>

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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“ Executing an alter-type worklist” on page 537</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 94.

---

**Figure 94  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>1 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>CC</td>
<td>COLUMN_2</td>
<td>SMALLINT</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLUMN_3</td>
<td>CHAR</td>
<td>12</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>COLUMN_4</td>
<td>CHAR</td>
<td>7</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>COLUMN_5</td>
<td>CHAR</td>
<td>4</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>COLUMN_6</td>
<td>VARCHAR</td>
<td>55</td>
<td>N</td>
<td>N</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 95. The column added for the constraint is displayed below the constraint key.

**Figure 95  Edited constraint key columns**

```
ALUSUC2 ER ------------------ Constraint Key Mixed List ------------------
Command ===>                                                    Scroll. . PAGE
WORKID . . . : RDACRJ.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:     +
Act         Colname                          Column Type      UDT Length N1 FP
************************************ TOP **************************************
CONSTRAN     KEY .:                                        *****************
   L 1   COLUMN_1
   L 2   COLUMN_2
   L 3   COLUMN_3
   4   COLUMN_15
PARENT TABLE .: ACMX01.T_X01PS
    COLUMN_1                         INTEGER                      N  N
    COLUMN_2                         SMALLINT                     N  N
    COLUMN_3                         CHAR                     12  N  N
```
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 <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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</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 96**.

**Figure 96  Constraint Key Mixed List panel**

```
ALUSUC2 ER ------------------------ Constraint Key Mixed list ------------------------
Command ===> Scroll . . PAGE
WORKID .. . : RDACRJ.UC01 Commands: CANCEL
Table . . . : RDACRJ.TB_N2A
Const Name . : TN2A_TS_1

(C)opy column from the Table, indicate position in Key Columns using (A)fter/(B)efore, and press ENTER.

Act       Colname                  Column Type      UDT Length N1 FP
**************************************** TOP ****************************************
A    CONSTRAINT KEY .:
  1   TN2A_TS_1
D    2   TN2A_C4_1
C    PARENT TABLE . ..: RDACRJ TB_N2A
    C1                               CHAR                      3  N  N
    TN2A_C4_1                        CHAR                      4  N  N
    TN2A_C4_2                        CHAR                      4  N  N
    TN2A_C6_3                        CHAR                      6  N  N
    TN2A_C4_4                        CHAR                      4  N  N

Lines 1 to 7 of 19
More: +
```

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.
B 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.

C Press Enter.

The Constraint Key Mixed List panel is displayed as shown in Figure 97. The column added for the constraint is displayed below the constraint key.

**Figure 97 Constraint Key Mixed List panel**

<table>
<thead>
<tr>
<th>Action</th>
<th>Column Name</th>
<th>Column Type</th>
<th>UDT Length</th>
<th>Null</th>
<th>Foreign Primary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C1</td>
<td>CHAR</td>
<td>3</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>TN2A_C4_1</td>
<td>CHAR</td>
<td>4</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>TN2A_C4_2</td>
<td>CHAR</td>
<td>4</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>TN2A_C6_3</td>
<td>CHAR</td>
<td>6</td>
<td>N</td>
<td>N</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.
Naming a primary key

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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 98.
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

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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.
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 99.

**Figure 99  Index Key Mixed List panel**

<table>
<thead>
<tr>
<th>Action</th>
<th>Column Name</th>
<th>Column Type</th>
<th>UDT Length</th>
<th>Null</th>
<th>Pad</th>
</tr>
</thead>
<tbody>
<tr>
<td>A INDEX KEY . . : DEMOCJ IXS23A</td>
<td>TABLE COLUMNS . : DEMOCJ TBS23A</td>
<td>S</td>
<td>C1</td>
<td>CHAR</td>
<td>3 N N</td>
</tr>
<tr>
<td>C</td>
<td>TS23A_C4_1</td>
<td>CHAR</td>
<td>4 N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_C4_2</td>
<td>CHAR</td>
<td>4 N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_C6_3</td>
<td>CHAR</td>
<td>6 N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_C4_4</td>
<td>CHAR</td>
<td>4 N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_VC10_1</td>
<td>VARCHAR</td>
<td>10 N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_SI_1</td>
<td>SMALLINT</td>
<td>N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_I_1</td>
<td>INTEGER</td>
<td>N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_FS_1</td>
<td>FLOAT</td>
<td>21 N N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS23A_FD_1</td>
<td>FLOAT</td>
<td>53 N N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 100, displays the index key column that has been added.

**Figure 100 Index Key Mixed List panel—column added**

<table>
<thead>
<tr>
<th>ALUSIXXR R</th>
<th>Index Key Mixed List</th>
<th>Command ===&gt;</th>
<th>Scroll. . PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKID . . . : RDACRJ.CHGCOL</td>
<td></td>
<td>Commands: CANCEL END</td>
<td></td>
</tr>
<tr>
<td>(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.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objects 1 to 9 of 23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>More:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Act</td>
<td>Colname</td>
<td>Column Type</td>
<td>UDT Length Nl FP</td>
</tr>
<tr>
<td>INDEX KEY . . : DEMOCJ IXS23A</td>
<td>L 1 C1</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>TABLE COLUMNS . : DEMOCJ TBS23A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>CHAR</td>
<td>3 N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_C4_1</td>
<td>CHAR</td>
<td>4 N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_C4_2</td>
<td>CHAR</td>
<td>4 N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_C6_3</td>
<td>CHAR</td>
<td>6 N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_C4_4</td>
<td>CHAR</td>
<td>4 N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_VC10_1</td>
<td>VARCHAR</td>
<td>10 N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_SI_1</td>
<td>SMALLINT</td>
<td>N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_I_1</td>
<td>INTEGER</td>
<td>N N</td>
<td></td>
</tr>
<tr>
<td>TS23A_FS_1</td>
<td>FLOAT</td>
<td>21 N N</td>
<td></td>
</tr>
</tbody>
</table>

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 <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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis</strong>.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.

**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 101, and provide TBCREATOR information.

**NOTE**
You can copy the DDL statement from member ACMDMLU8 in the HLQ.DBCNTL data set.

---

**Figure 101  DDL to create a view**

```sql
CREATE VIEW DEM.NOINDEX_FOR_ROWID (CREATOR, NAME, COLUMN) AS
    SELECT ALL
        COLS.TBCREATOR,           COLS.TBCREATOR = IXS.TBCREATOR
        COLS.TBNAME,                COLS.TBNAME = IXS.TBNAME
        COLS.NAME                   AND COLS.NAME = KYS.COLNAME
FROM SYSIBM.SYSCOLUMNS COLS
WHERE COLS.DEFAULT = 'A'
    AND TBCREATOR LIKE 'DEMO922%'  -- change or remove
    AND NOT EXISTS (SELECT * FROM
        SYSIBM.SYSINDEXES IXS,       AND IXS.UNIQUERULE = 'U'
        SYSIBM.SYSKEYS    KYS
WHERE
        COLS.TBCREATOR = IXS.TBCREATOR
        COLS.TBNAME     = IXS.TBNAME
        COLS.NAME       = KYS.COLNAME
        AND KYS.IXCREATOR = IXS.CREATOR
        AND KYS.IXNAME   = IXS.NAME
        AND IXS.UNIQUERULE = 'U'
        AND IXS.COLOCOUNT = 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 374.

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

**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.

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.

**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.

The Table Detail panel is displayed.

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.
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 102.

**Figure 102 Table Quick Edit Confirmation dialog**

![Table Quick Edit Confirmation dialog](image)

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.
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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 352. To define the data structure changes, see page 378.

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 616.

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**.
Changing data structures by using an existing work ID

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 Getting Started Guide.

**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.
Receiving DDL to create data structures

7 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 537.

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 356.
Receiving DDL to create data structures

Chapter 4 Altering data structures in a database environment

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.”

   TIP

   If you use an inbound migrate profile with this script and you receive frequent changes for the DB2 application, define the profile to a Application. Then you can easily reuse the profile’s change rules by specifying the Application each time that you receive data structure changes to the DB2 application.

   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.
Receiving DDL to create data structures

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 616.

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 Getting Started Guide.

   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 409 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.
Changing data structures by using DML

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 537.

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.

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.
- 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 616.

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 Getting Started Guide.

---

**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.
7 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 537.

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 410.

Before you begin

Create a temporary table, as shown in Figure 104, 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 104 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
;
```
To create and import the DML statement

1. In an ISPF editor, type the DML statement to match the one shown in Figure 105 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 105  DML to add columns to the PLAN_TABLE

```sql
-- EXAMPLE DML TO ADD PLAN TABLE COLUMNS
LIKE   COLUMNS
JOIN SYSIBM.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 SYSIBM.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 374.

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

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 410.

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.
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 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.
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*.

**Figure 107  DML for copying a column**

```sql
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 374.

**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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
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 updating index parts**

```
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 108, 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 374.

**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.
Adding WITH RESTRICT ON DROP to tables by importing DML

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 410.

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 to add WITH RESTRICT ON DROP**

   ```
   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 374.

**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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>
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 410.

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.

   **TIP**

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

```
Figure 110  DML to locate indexes with existing data sets

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 374.
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.

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 statement to create synonyms for selected tables from member ACMDMLU3 in the HLQ.DBCNTL data set.

Figure 111  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 111, 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 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.
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 statements to convert the partitions from member ACMDMLU7 in the HLQ.DBCNTL data set.

---

**Figure 112  DML to convert VCAT-defined partitions to STOGROUP-defined partitions**

<table>
<thead>
<tr>
<th>UPDATE TABLEPARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET STORTYPE = 'I',</td>
</tr>
<tr>
<td>STORNAME = 'ACMX',</td>
</tr>
<tr>
<td>VCATNAME = ''</td>
</tr>
<tr>
<td>WHERE</td>
</tr>
<tr>
<td>VCATNAME = 'DEBACAT'</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>UPDATE INDEXPARTS</td>
</tr>
<tr>
<td>SET STORTYPE = 'I',</td>
</tr>
<tr>
<td>STORNAME = 'ACMX',</td>
</tr>
<tr>
<td>VCATNAME = ''</td>
</tr>
<tr>
<td>WHERE</td>
</tr>
<tr>
<td>VCATNAME = 'DEBACAT'</td>
</tr>
<tr>
<td>;</td>
</tr>
</tbody>
</table>

In Figure 112, 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 374.

**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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
H Type S to select MQT Parameters.

The Create MQT Parameters panel is displayed.

7 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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
On the Tablespace Estimation panel, you can modify the values for the fields.

- To estimate the space for a nonpartitioned table space (Figure 113), you can complete one of the following steps:

**Figure 113 Tablespace Estimation panel (nonpartitioned)**

**Tablespace Name:** DEMOLAB1.DE0030  
**Pagesize:** 4  
**Segsize:** 4 (0-64)  
**Priqty:** 100  
**Secqty:** 100  
**Allocation Unit:** K (K,C,T)  
**Device Type:** 3390  
**Freepage:** 1 (0-255)  
**Pctfree:** 10 (0-99)  
**Compress:** N (Y,N)  

---

**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.

- 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 114), type over the values for **Average to Use in Estimate**. Then press END.
To estimate the space for a partitioned table space (Figure 115), you can complete one of the following steps:

Figure 115 Tablespace Estimation panel (partitioned)
Estimating the space requirements for a table space

Chapter 4 Altering data structures in a database environment 431

— 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 116), type over the values that you want to change. Then press END.

Figure 116 Space Estimation Attribute Propagation panel

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 114), 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 117), 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.

Figure 117 Tablespace Estimation panel (partition-by-growth table space)

<table>
<thead>
<tr>
<th>ALUSTGSE R</th>
<th>------------------ Tablespace Estimation (Partition By Growth) ------------------</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Scroll. . PAGE</td>
</tr>
<tr>
<td>Tablespace Name: J9QTALT.TRT10AIM</td>
<td></td>
</tr>
<tr>
<td>TB Name</td>
<td>J9QTALT.T_T10AIM</td>
</tr>
<tr>
<td>Pagesize</td>
<td>8</td>
</tr>
<tr>
<td>Segsize</td>
<td>4 (1-64)</td>
</tr>
<tr>
<td>PriQty</td>
<td>-1</td>
</tr>
<tr>
<td>SecQty</td>
<td>-1</td>
</tr>
<tr>
<td>Allocation Unit</td>
<td>K (K,C,T)</td>
</tr>
<tr>
<td>Device Type</td>
<td>3390</td>
</tr>
<tr>
<td>Freepage</td>
<td>0 (0-255)</td>
</tr>
<tr>
<td>Pctfree</td>
<td>5 (0-99)</td>
</tr>
<tr>
<td>Compress</td>
<td>N (Y,N)</td>
</tr>
<tr>
<td>Maxrows</td>
<td>255</td>
</tr>
<tr>
<td>D ssize</td>
<td>4 G</td>
</tr>
<tr>
<td>Pct Compressed</td>
<td>0 (0-99)</td>
</tr>
<tr>
<td>Maxpartitions</td>
<td>256</td>
</tr>
<tr>
<td>Avg Part</td>
<td>NbrRows Avg Orig Top Space Rows/Page</td>
</tr>
<tr>
<td>1</td>
<td>0 843 Proj 0 9</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.

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 35 on page 334.

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 118.

**Figure 118 Mixed List panel**

<table>
<thead>
<tr>
<th>ALUSMXL ER</th>
<th>--------------------------- Mixed List ---------------------------</th>
<th>Scroll. . PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKID . . .</td>
<td>: RDACRJ.ALTER</td>
<td>Commands: CANCEL</td>
</tr>
<tr>
<td>Type action next to object and press Enter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E=Edit L=Like D=Drop U=Undo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Act</td>
<td>Object-Type</td>
<td>Objects</td>
</tr>
<tr>
<td>$</td>
<td>TB . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>FK . . .</td>
<td>RDACRJ</td>
<td>&quot;tbpl1a&quot;</td>
</tr>
<tr>
<td>IXC. . .</td>
<td>RDACRJ</td>
<td>IXP7A__C</td>
</tr>
<tr>
<td>IXP. . .</td>
<td>RDACRJ</td>
<td>IXP7A_2</td>
</tr>
<tr>
<td>UCP. .</td>
<td>RDACRJ</td>
<td>&quot;tbpl1a&quot;</td>
</tr>
<tr>
<td>AL . . .</td>
<td>RDACRJ</td>
<td>ALP1A__C</td>
</tr>
<tr>
<td>TS . . .</td>
<td>DEMOCRJ</td>
<td>ROBBINS</td>
</tr>
<tr>
<td>TB . . .</td>
<td>RDACRJ</td>
<td>JGS0004_A17_LOWC</td>
</tr>
<tr>
<td>TB . . .</td>
<td>RDACRJ</td>
<td>JNMP007_BOHEADER</td>
</tr>
<tr>
<td>TB . . .</td>
<td>RDACRJ</td>
<td>JNMP029_BO_CMDTY</td>
</tr>
<tr>
<td>TS . . .</td>
<td>DEMOCRJ</td>
<td>SFI</td>
</tr>
<tr>
<td>TS . . .</td>
<td>DEMOCRJ</td>
<td>SF1X</td>
</tr>
<tr>
<td>TS . . .</td>
<td>DEMOCRJ</td>
<td>S1</td>
</tr>
<tr>
<td>TB . . .</td>
<td>RDACRJ</td>
<td>TBS1A</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.

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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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 119.

*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.
Estimating the space requirements for an index

Chapter 4 Altering data structures in a database environment

Figure 119 Index Space Estimation panel (partitioned)

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 (Alloc 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 (Alloc 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 120), 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.
Estimating the space requirements for an index

Figure 120  Space Estimation Attribute Propagation panel

<table>
<thead>
<tr>
<th>IX</th>
<th>ALUSPASX</th>
<th>Space Estimation Attribute Propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB</td>
<td>Command ===&gt;</td>
<td>K</td>
</tr>
<tr>
<td>Tab</td>
<td>Command ===&gt;</td>
<td></td>
</tr>
<tr>
<td>Uní</td>
<td>Index Name . . : RDACRJ.IXP7A__C</td>
<td>----</td>
</tr>
<tr>
<td>Device Type . . : 3390</td>
<td>f 7</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>Make any changes necessary or blank out to leave current value; changes will be applied to all parts where applicable.</td>
<td>-----</td>
</tr>
<tr>
<td>pa</td>
<td>AVG ROWS/ ALOC</td>
<td>0</td>
</tr>
<tr>
<td>NBRROWS LEN KEY PRIQTY SECOTY UNIT</td>
<td>0 4 1 12 12 K</td>
<td></td>
</tr>
<tr>
<td>Commands: END CANCEL</td>
<td>*****</td>
<td></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.

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 37 on page 338. 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 121.
Estimating the space requirements for an index

Figure 121 Mixed List panel

<table>
<thead>
<tr>
<th>ALUSMXL ER</th>
<th>Mixed List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Scroll . PAGE</td>
</tr>
<tr>
<td>WORKID</td>
<td>RDACRJ.ALTER</td>
</tr>
<tr>
<td>Type action next to object and press Enter.</td>
<td>Commands: CANCEL</td>
</tr>
<tr>
<td>E=Edit L=Like D=Drop U=Undo</td>
<td></td>
</tr>
<tr>
<td>Act</td>
<td>Object-Type</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>FK . . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>IXC . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>IXP . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>UCP . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>AL . . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCRJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>RDACRJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCRJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCRJ</td>
</tr>
<tr>
<td>TS . . . .</td>
<td>DEMOCRJ</td>
</tr>
<tr>
<td>TB . . . .</td>
<td>RDACRJ</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.

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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 438
- “To estimate space requirements for a partitioned table space” on page 439
- “To estimate space requirements for a partition-by-growth table space” on page 440

**NOTE**
Space estimation is not available for XML objects or for 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 81 on page 340) 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 81 on page 340) as needed to correspond to your table space.

--- NOTE ---
For more information about a specific field, press F1.

--- 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 81 on page 340) 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 or for 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 81 on page 340) 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 331.

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 410.
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 122** shows an example of setting the cardinality of a table to 50 and setting the average row length to 2000.

**Figure 122  DML to update a table**

```sql
UPDATE TABLES SET
  CARDF = 50,
  ROWAVG = 2000
WHERE
  SYSLD.SYSTABLES.DBNAME = 'ALUQA75'
  AND SYSLD.SYSTABLES.CREATOR = 'ALUO750'
  AND SYSLD.SYSTABLES.NAME = 'TBS1A'
;
```

**Figure 123** shows an example of setting the cardinality of a table partition to 153.

**Figure 123  DML to update a table partition**

```sql
UPDATE TABLEPART SET
  CARDF = 153
WHERE
  PARTITION > 0
  AND TSNAME = 'P1'
  AND DBNAME = 'ALUQA75'
;
```

**Figure 124** shows an example of setting the cardinality of an index partition to 50 and setting the ROWSPKEY to 500.

**Figure 124  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 125** 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 125** 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 126** 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 126** 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 127** 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 127** DML to update an index partition by using the JOIN command (part 1 of 2)

```
UPDATE INDEXPARTS
JOIN ACM83.CD_TABLEPART CD
SET CARDF = CD.CARDF,
        ROWSPKEYF = CD.ROWSPKEYF,
        PQTY = 100
WHERE
    ;  
```

---
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 410.

1 In an ISPF editor, type the DML statements to match the one shown in Figure 128. 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 127  DML to update an index partition by using the JOIN command (part 2 of 2)

<table>
<thead>
<tr>
<th>Table: INDEXPARTS.</th>
<th>CD.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IXNAME</td>
<td>IXNAME</td>
</tr>
<tr>
<td>IXCREATOR</td>
<td>IXOWNER</td>
</tr>
<tr>
<td>IXPARTITION</td>
<td>PARTID</td>
</tr>
<tr>
<td>PARTTYPE</td>
<td>'I'</td>
</tr>
<tr>
<td>CDNAME</td>
<td>'SE531OA'</td>
</tr>
<tr>
<td>CDOWNER</td>
<td>'AUC'</td>
</tr>
</tbody>
</table>

; 

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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 410.

1 In an ISPF editor, type the DML statements to match the one shown in Figure 128. Customize the database name to match your local requirements.

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

---

444 ALTER and CHANGE MANAGER for DB2 User Guide
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 128  DML to estimate the primary and secondary quantities

```sql
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 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
Setting a secondary quantity in table space and index partitions 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 410.

1 In an ISPF editor, type one of the DML statements shown in Figure 129 and provide the information that is unique to your site.

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

**NOTE**
PRIQTY and SECQTY are expressed in 4K pages.

**Figure 129**  Example DML for setting SQTY

```sql
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 374.
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</strong> and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID</strong> worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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 130**) 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*. 

Chapter 4  Altering data structures in a database environment  447
Figure 130  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
--
-- 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>'
```
Figure 130  DML statements to create and modify the target set of objects (part 2 of 3)

```sql
;
-- 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>'
;
-- LIKE ORIGINAL FK AND REFERENCE NEW TABLE - AS PARENT - ONLY
LIKE RELS
SET REFTBNAME = REFTBNAME
WHERE
```
Modifying and migrating objects for the SHRLEVEL REFERENCE HSSC process

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

Modifying and migrating objects for the SHRLEVEL REFERENCE 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 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 131) 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 131  DML statements to modify and migrate objects (part 1 of 2)

--- After you edit the DML statements, you can use CHANGE MANAGER to import them.
--- 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
--- IMPORT THIS DML INTO AN ALTER WORKID. IT WILL CONVERT THE WORKID TO A MIGRATE WORKID
--- AFTER DML IS IMPORTED USE SPECIFICATION TO MODIFY NEW TS WITH DESIRED CHARACTERISTIC
--- MIGRATE TS FOR ORIGINAL TABLE.
MIGRATE TABLESPACE
JOIN SYSIBM.SYSTABLES TBLS
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.TNAME
AND SYSIBM.SYSTABLES.CREATOR = '<TBOWNER>'
AND SYSIBM.SYSTABLES.NAME = '<TBNAME>'
;
--- MIGRATE ORIGINAL TABLE
Converting table spaces to partition-by-growth table spaces by importing DML

Figure 131  DML statements to modify and migrate objects (part 2 of 2)

```sql
-- ADDING NEW PREFIX TO ITS NAME
MIGRATE TABLES
INCLUDE AUTHORIZATIONS, CHECKS,
INDEXES, CONSTRAINTS
SET NAME = SUBSTR('NEW' || NAME, 1, 100)
WHERE
CREATOR = '<TBOWNER>'
AND NAME = '<TBNAME>'
;
-- MIGRATE INDEXES ON ORIGINAL TABLE
-- ADDING NEW PREFIX TO THEIR NAMES
MIGRATE INDEXES
SET NAME = SUBSTR('NEW' || NAME, 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 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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
Converting table spaces to partition-by-growth table spaces by importing DML

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 410.

1. Copy the DML statement from member ACMDMLUC in the HLQ.DBCNTL data set.

2. Edit the DML statement (Figure 132) 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*.

3. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 500.

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

**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.

   **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
   - Alloc 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.
To create the clustering, partitioning index for the base table

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 133.
Converting a nonpartitioned table space to an index-controlled partitioned table space

Figure 133  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 134. The index key column is displayed below the index key.
Converting a nonpartitioned table space to an index-controlled partitioned table space

**Figure 134  Index Key Mixed List panel**

- 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**.

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.
Converting a nonpartitioned table space to an index-controlled partitioned table space

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
- **Alloc 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 Getting Started Guide*.

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 500.

9 Press END until the WORKID Action Menu is displayed.
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 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). 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.

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.
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
Converting a nonpartitioned table space to a range-partitioned table space

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 460.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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. convert to an explicit table space (page 465), or to an implicit database or table space (page 467)

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 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.
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 135).

   NOTE

   The messages that are displayed in the dialog vary, depending on the values that you have set for the DSSIZE and SEGSIZE.
Converting a nonpartitioned table space to a partition-by-growth table space

**Figure 135 Partition-by-growth dialog**

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.

**4** If multiple tables exist in the table space, drop or move all of the tables except the table that you want to convert:

- To drop a table on the Mixed List panel, type **D** in the **Act** column adjacent to the table and the table’s dependent objects.
- 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 implicit database or table space

1 Perform step 4 on page 466.

2 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>.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.

6 On the Tablespace Parts List panel, perform the following actions:

A For the partitioning status, select Non-Partitioned.

B In the Dsize 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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.

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 136.

**Figure 136** DML to change table-controlled partitioned table spaces to range-partitioned table spaces

```
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
;```

Chapter 4  Altering data structures in a database environment  469
To change table-controlled table spaces by using a JOIN statement, type the DML shown in Figure 137. 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 137  DML to change table-controlled partitioned table spaces to range-partitioned table space by using a JOIN**

```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
  AND TBLS.PARTKEYCOLNUM > 0 -- indicates table-controlled partitioning
  AND TBLS.DBNAME = SYSIBM.SYSTABLESPACE.DBNAME
  AND TBLS.TSNAME = SYSIBM.SYSTABLESPACE.NAME;
```

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 138.

**Figure 138  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 374.

**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 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.

<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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.
1 Copy the DML statements from member ACMDMLUB in the HLQ.DBCNTL data set.

2 Edit one of the DML statements (Figure 139) 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.

**Figure 139  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
;
```

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

**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 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 471.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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 471.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 141:
     
     — 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 477.

   - Otherwise, the Preserve Limitkey Values panel is displayed, as shown in Figure 140.
Converting an index-controlled partitioned table space to a table-controlled partitioned table space

### Figure 140  Preserve Limitkey Values panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Table Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>WOR</td>
<td>ALUSTKP Preserve Limitkey Values</td>
</tr>
<tr>
<td>TB</td>
<td>N (Y/N) Should the product preserve the limit key values of the last partition?</td>
</tr>
<tr>
<td>Dat</td>
<td>Y - If the data stored in the last partition is beyond the limit key value, the utility might discard rows when it reorganizes the partition.</td>
</tr>
<tr>
<td>Aud</td>
<td>N - DB2 can automatically assign the limit key values for the last partition to MAXVALUE or MINVALUE.</td>
</tr>
<tr>
<td>Val</td>
<td>Should the product preserve the limit key values of the last partition?</td>
</tr>
<tr>
<td>Edi</td>
<td>Should the product preserve the limit key values of the last partition?</td>
</tr>
<tr>
<td>S</td>
<td>PF1 - Help PF3 - Exit/Process Response</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sel</th>
<th>Table Column List _ Table Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>_ Primary Key Columns _ Auxiliary Objects List</td>
</tr>
<tr>
<td></td>
<td>_ Partitioning Key Columns _ Parts List and LimitKey Values</td>
</tr>
<tr>
<td></td>
<td>_ MQT Parameters _ Implicit TS Dsize</td>
</tr>
</tbody>
</table>

**Commands:** HELP END CANCEL PF4=ZOOM

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 318.

**B** Press END to exit the panel.

The Table Detail panel is displayed, as shown in Figure 141.
Converting an index-controlled partitioned table space to a table-controlled partitioned table space

Chapter 4 Altering data structures in a database environment

Figure 141 Table Detail panel

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.

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 LIKEEd 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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
Converting index-controlled partitioned table spaces to table-controlled partitioned table spaces by importing DML

1 Copy the DML statement from member ACMDMLU9 in the HLQ.DBCNTL data set.

2 Edit the DML statement (Figure 142) 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.

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 475.

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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

Converting index-controlled partitioned table spaces to range-partitioned table spaces by importing DML

You can import DML to convert t 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 410.
1 Copy the DML statements from member ACMDMLUA in the HLQ.DBCNTL data set.

2 Edit the DML statements (Figure 143) 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 143  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 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>
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.

**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 144).

**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 144  Partition-by-growth dialog**

```
ALUS9TSP R  ---------------------- Tablespace Parts List ----------------------
Command ===> Scroll . . CSR
WORKID . . . : RDACRJ.ICPPBG Commands: CANCEL
Database Name . . : C9AIP1 Select partitioning status.

<table>
<thead>
<tr>
<th>ALUSTSG</th>
</tr>
</thead>
</table>
| 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
```

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.
Converting a partitioned table space to a partition-by-growth table space

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.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 454.

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 Analyze WORKID and create a worklist.</td>
<td>“Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER” on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 460.

8 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.

9 Type S to select Tablespace Partition List.

10 On the Tablespace Parts List 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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 487.

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.
Changing a single-table table space to a partition-by-growth table space by importing DML

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.

1 In an ISPF editor, type the appropriate DML statement (**Figure 145**) 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 145  DML to change a table space to a partition-by-growth table space**

```sql
UPDATE TABLESPACE
SET
```
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 410.

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 146.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

Figure 145  DML to change a table space to a partition-by-growth table space

MAXPARTITIONS = 10
WHERE DBNAME = 'MG025'
AND NTABLES = 1
;
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 147. The second DML statement deletes the table spaces that contain more than one table.

2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.

1. Copy the DML statements from member ACMDMLUD 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. Delete the other statement.

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

Figure 148  DML to convert to implicit databases and table spaces (part 1 of 2)

```sql
-- 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 410.

1. Copy the DML statements from member ACMDMLUF 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. Delete the other statement.

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

Figure 148  DML to convert to implicit databases and table spaces (part 2 of 2)

```
WHERE
    DBNAME LIKE 'MG5%' AND
    NTABLES > 1
;
```

3. Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

**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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

Figure 149  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 149  DML to change the encoding scheme for a table space (part 2 of 2)

```sql
UPDATE TABLESPACE
SET
  ENCODING_SCHEME = 'U'  -- sets ENCODING_SCHEME to UNICODE
WHERE
  NAME = 'TS04P' AND
  DBNAME = 'DEMO105G'
```

3 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</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 150.
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 151. The index key column is displayed below the index key.
Adding a ROWID and a LOB column to create a base table

Chapter 4  Altering data structures in a database environment  499

Figure 151  Index Key Mixed List panel

D  Press END.

The Create Index panel is displayed.

4  Press END.

The Mixed List panel is displayed.

NOTE

The object type for the new index is displayed as IXU. After you execute the worklist, the object type for the index is displayed as IXG.

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 500.

- To create objects in a nonpartitioned table space, see “Creating auxiliary objects for a nonpartitioned table space” on page 513.
Creating auxiliary objects for a partitioned table space

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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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.
Creating auxiliary objects for a partitioned table space

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 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 three-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.
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 152.

Figure 152  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>&quot;Analyzing an alter-type work ID and generating a worklist in CHANGE MANAGER&quot; on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>&quot;Executing an alter-type worklist&quot; on page 537</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.
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.
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 153.
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 153 Auxiliary Objects List—single partition**

<table>
<thead>
<tr>
<th>Action</th>
<th>Object Type</th>
<th>Objects</th>
<th>Column</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>CJACML01</td>
<td>L01PSL1C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>RDACRJ</td>
<td>T_L01PSL1C</td>
<td>XCLOB</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>RDACRJ</td>
<td>I_L01PSL1C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>RDACRJ</td>
<td>LATP</td>
<td>XCLOB</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>RDACRJ</td>
<td>LIXP</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.
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

Liking a base table and creating auxiliary objects for a table space

You can like an existing base table that contains LOB columns. After you like the table, you must create the new auxiliary objects and associate them with the new table. You cannot like 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.
Liking a base table and creating auxiliary objects for a table space

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.

---

G To create a primary key for the table, type S to select Primary Key Columns.

---

**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.

---

H Press END.

The Mixed List panel is displayed.

---

**NOTE**

No auxiliary objects exist in the LIKEd base table.

---

6 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 500.

- To create objects in a nonpartitioned table space, see “Creating auxiliary objects for a nonpartitioned table space” on page 513.
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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 154.

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.
2 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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 data in XML and LOB columns” on page 652.

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.
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.

3 Press Enter.

4 Press END.

The Mixed List panel is displayed.

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.
Converting a VARCHAR data type to a LOB data type

**NOTE**
The object type for the new index is displayed as IXU. After you execute the worklist, the object type for the index is displayed as IXG.

8 Press END until the WORKID Action Menu is displayed.

9 Analyze the alter-type work ID.

**TIP**
To analyze the work ID, see page 535.

10 Execute the alter-type worklist.

**TIP**
To execute the worklist, see page 537.

**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.
Converting a VARCHAR data type to a LOB data type

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 500.

- To create objects in a nonpartitioned table space, see “Creating auxiliary objects for a nonpartitioned table space” on page 513.
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 <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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</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 410.

1. In an ISPF editor, type the DML statement to match the one shown in Figure 155. 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 155**  DML for creating a database and table spaces (part 1 of 2)

```sql
SET SPACE ESTIMATION --space estimation turned on
;
LIKE DATABASE
```
Creating a new database and new table spaces for tables by importing DML

Figure 155  DML for creating a database and table spaces (part 2 of 2)

```sql
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.SYSTABLEPART.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 155, 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 374.

**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 an alter-type work ID and generating a worklist in CHANGE MANAGER&quot; on page 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>&quot;Executing an alter-type worklist&quot; on page 537</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 410.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 156. 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

**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.

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 156  DML to change multi-table table spaces to single-table table spaces (part 1 of 2)

```sql
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.SYSTABLES.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.SYSTABLES.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
```

526  ALTER and CHANGE MANAGER for DB2 User Guide
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.

Figure 156  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
```

In the sample DML that is displayed in Figure 156, 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 374.

**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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

**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.
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 410.

1 In an ISPF editor, type the DML statement to match the one shown in Figure 157. 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 ACMDMLL2 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 157  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
```
2 Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis.</strong></td>
<td>“Executing an alter-type worklist” on page 537</td>
</tr>
</tbody>
</table>

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 410.
1 In an ISPF editor, type the DML statements to match those shown in Figure 158 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 158**  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 374.

### 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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 159.

**TIP**
You can copy the DML statement to perform this task from member ACMDMLU4 in the HLQ.DBCNTL data set.

**Figure 159  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;
```
Updating table spaces or indexes defined with DEFINE NO by importing DML

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 410.

1. In an ISPF editor, type the DML statements to match those shown in Figure 160 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 160  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 374.

**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 535</td>
</tr>
<tr>
<td>Select Execute WORKID worklist created by Analysis.</td>
<td>“Executing an alter-type worklist” on page 537</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 410.

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 161** 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.

**Figure 161** DML to set the length of an inline CLOB to the system default

```
UPDATE COLUMNS
  SET
    LENGTH = -2
  WHERE
    LENGTH > 4 AND
    COLTYPE IN ('CLOB') AND
    TBCREATOR = 'COAIN1'
```

- **Figure 162** illustrates how to change an inline CLOB column to a non-inline CLOB column.

**Figure 162** DML to change an inline CLOB column to a non-inline CLOB column

```
UPDATE COLUMNS
  SET
    LENGTH = 4
  WHERE
    LENGTH > 4 AND
    COLTYPE IN ('CLOB') AND
    TBCREATOR = 'D2NIC1'
```
Figure 163 illustrates how to change the length for non-inline LOB columns to inline LOB columns.

**Figure 163  DML to change non-inline LOB columns to inline LOB columns**

```sql
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
 TBCREATOR = 'C1AIB1'
;
```

Figure 164 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 164  DML to change the length of LOB columns in universal table spaces**

```sql
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 = 'COAIN1'
;
```

Figure 165 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 165  DML to set the length of the CLOB column based on the maximum length (part 1 of 2)**

```sql
UPDATE COLUMNS
SET
 LENGTH =
CASE
```
Import the DML file. For information, see “Importing a CDL, DDL, or DML file in CHANGE MANAGER” on page 374.

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 535</td>
</tr>
<tr>
<td>Select <strong>Execute WORKID worklist created by Analysis</strong>.</td>
<td>“Executing an alter-type worklist” on page 537</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 62.

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.

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 537.

**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 162. To start a worklist over, see page 164.

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 539.

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 Getting Started Guide.
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 83.

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 537 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>
Managing environments with ALTER and CHANGE MANAGER

This part presents the following topics:

Chapter 5
  Maintaining database environments ........................................ 545

Chapter 6
  Managing your PeopleSoft environment with CHANGE MANAGER ........ 583
Maintaining database environments

This chapter presents the following topics:

Before you begin .......................................................... 545
Overview ................................................................. 546
Migrating changes to data structures ................................ 546
  Acting as the sending subsystem .................................. 547
  Acting as the receiving subsystem ................................. 549
Maintaining applications ............................................... 551
  Implementing a strategy ............................................. 552
Recovering data structures ............................................. 553
Feeding back changes to data structures ........................... 554
Using data modeling tools .............................................. 555
  Importing DDL from a data modeling tool ..................... 556
  Maintaining catalog information ................................. 557
Accomplishing your goals ............................................... 557
  Migrating data structure changes to other subsystems ...... 558
Merging subsystems by using ALTER ................................ 559
Receiving data structure changes from a strong sender ....... 561
Receiving data structure changes from a strong receiver .... 564
Restoring data structures with current data ...................... 569
Recovering data structures with current data ................... 570
Restoring data structures with old data ......................... 575
Recovering data structures with old data ....................... 576
Where to go from here .................................................. 581

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.
Acting as the receiving subsystem

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.

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 41 lists the tasks that you can perform to maintain your database environments.

Table 41 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 558</td>
</tr>
<tr>
<td>“Merging subsystems by using ALTER”</td>
<td>page 559</td>
</tr>
<tr>
<td>“Receiving data structure changes from a strong sender”</td>
<td>page 561</td>
</tr>
<tr>
<td>“Receiving data structure changes from a strong receiver”</td>
<td>page 564</td>
</tr>
<tr>
<td>“Restoring data structures with current data”</td>
<td>page 569</td>
</tr>
<tr>
<td>“Recovering data structures with current data”</td>
<td>page 570</td>
</tr>
<tr>
<td>“Restoring data structures with old data”</td>
<td>page 575</td>
</tr>
<tr>
<td>“Recovering data structures with old data”</td>
<td>page 576</td>
</tr>
</tbody>
</table>
Migrating data structure changes to other subsystems

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 196.
   B Create a baseline from the baseline profile that you created in step 1A. See “Creating a structure-only catalog baseline” on page 199.

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 144.
   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 278.

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 352.
   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 374.

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 535.

5 Execute the worklist. For more information, see “Executing an alter-type worklist” on page 537.
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 115.

Using the Specification component in ALTER, migrate the DB2 objects that were specified in step 2A on page 559. For more information, see “Specifying the data structures in a migrate-type work ID” on page 120.

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.

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 151.

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

NOTE

When you add OBJINFO in the Analysis ALUIN input stream, additional comments are placed in the worklist which shows the object’s original OBID, DBID, PSID, ISOBID, and data set names. This information is useful if you are going to follow the procedures in the IBM documentation to merge your catalog using the IBM DSN1COPY utility outside of ALTER or the original data sets. If you use DSN1COPY or the original data sets to merge the catalog, you must reorder the worklist.

F Execute phase 1 of the worklist to unload the data from the tables. For more information, see “Restarting a worklist” on page 162.

3 Perform the following steps on the target subsystem.

A Create a receive work ID with the same name specified in step 2A on page 559. For more information, see “Creating a receive-type work ID” on page 154.

B Execute phase 2 of the worklist to create the objects on the target subsystem, run the utilities, and bind the packages and plans. See “Restarting a worklist” on page 162.

4 Repeat steps 2 and 3 for the other source subsystems.

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 356.

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.”

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.
Receiving data structure changes from a strong sender

3 Type the data set name for Diagnostics or type SYSOUT.

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 616.

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 Getting Started Guide for DB2.

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.
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 537.

---

**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.
Receiving data structure changes from a strong receiver

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 616.

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 Getting Started Guide for DB2.

--- 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.
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.

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.
Receiving data structure changes from a strong receiver

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 568 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.
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 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 537.

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 570.

Before you begin

Ensure that a catalog baseline exists. For information about catalog baselines, see Chapter 2, “Taking a snapshot of a database environment.”


To restore data structures

1 Compare the DB2 catalog to the baseline. See “Comparing a DB2 catalog to a baseline” on page 285.

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 352 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 374.

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 535.

   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 537.

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 196 and “Creating a structure-only catalog baseline” on page 199.

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 616.

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 Getting Started Guide for DB2.

---

**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.
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 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.

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 537.

**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 576.

**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 285.

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 352.

   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 374.
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 535.

**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 537.

---

**Recovering data structures with old data**

The CM/PILOT RCV_STRUCT_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 196 and “Creating a full-recovery 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 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.
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 616.

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 Getting Started Guide for DB2.

---

**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.
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 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 537.
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 ................................................................. 583
Working in a PeopleSoft environment ......................... 584
Meeting the challenges of a PeopleSoft environment ...... 585
Taking a snapshot of the initial environments .............. 585
Creating copies of an existing environment ................. 587
Optimizing the design of an environment .................... 592

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 42 lists these environments and the abbreviation that PeopleSoft uses to identify them.

Table 42  Typical PeopleSoft environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>application update</td>
<td>AUDB</td>
</tr>
<tr>
<td>patch</td>
<td>PATCH</td>
</tr>
<tr>
<td>development</td>
<td>DEV</td>
</tr>
<tr>
<td>conversion</td>
<td>CONV</td>
</tr>
<tr>
<td>test</td>
<td>TST</td>
</tr>
<tr>
<td>training</td>
<td>TRN</td>
</tr>
<tr>
<td>production</td>
<td>PROD</td>
</tr>
<tr>
<td>quality assurance</td>
<td>QA</td>
</tr>
</tbody>
</table>

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 166 illustrates the process of creating copies of an existing 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 BMIDB2I8 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 SYNCPPOINT 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.
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 Getting Started Guide*.

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 167 illustrates the process of optimizing the design of an environment.

**Figure 167  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 ACMDMLL2 member of the \( \text{HLQ.} \text{DBCNTL} \) 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.
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.

**For more information**

For more information about optimizing the design of an environment, see Chapter 4, “Altering data structures in a database environment.”
This part presents the following topics:

Appendix A
Using scripting tools to automate change management .......................... 599

Appendix B
Using utilities with ALTER and CHANGE MANAGER ........................... 647

Appendix C
Using the skeleton library compiler ................................................. 691

Appendix D
Using worklist parallelism ............................................................... 699
Using scripting tools to automate change management

This appendix presents the following topics:

Overview ................................................................. 600
Using task IDs .......................................................... 601
  Creating a task ID ................................................. 602
  Listing task IDs .................................................... 604
  Editing a task ID .................................................. 605
  Executing a task ID .............................................. 605
  Viewing the execution status of a task ID ...................... 613
  Browsing a task ID ............................................. 614
  Deleting a task ID ............................................... 615
Processing CM/PILOT worklists ........................................ 616
  Restarting a CM/PILOT worklist ............................... 616
  Starting a CM/PILOT worklist over ......................... 618
Using scripts .......................................................... 620
  Creating a script ............................................... 622
  Listing scripts ................................................... 633
  Editing a script .................................................. 634
  Listing the steps for a script ................................ 635
  Executing a user-defined script ............................... 636
  Browsing a script ............................................... 636
  Deleting a script ............................................... 637
  Creating a task ID from a script .............................. 638
  Listing task IDs for a script ................................ 639
  Copying a script supplied by BMC ............................ 639
  Copying a user-defined script ............................... 640
Using applications .................................................. 640
  Creating an application ....................................... 642
  Listing applications .......................................... 643
  Editing an application ........................................ 643
  Browsing an application ..................................... 644
  Deleting an application ..................................... 645
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.

### 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 43 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 602</td>
</tr>
<tr>
<td>“Listing task IDs”</td>
<td>page 604</td>
</tr>
<tr>
<td>“Editing a task ID”</td>
<td>page 605</td>
</tr>
<tr>
<td>“Executing a task ID”</td>
<td>page 605</td>
</tr>
<tr>
<td>“Viewing the execution status of a task ID”</td>
<td>page 613</td>
</tr>
<tr>
<td>“Browsing a task ID”</td>
<td>page 614</td>
</tr>
<tr>
<td>“Deleting a task ID”</td>
<td>page 615</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.

   Table 44 lists the available CM/PILOT scripts, the type of work ID that each script requires, and whether that work ID must be a new work ID or an existing work ID.

   **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.

   The abbreviated script names that are listed in the table are displayed on various panels in the dialog.

<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>
</tbody>
</table>
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.

**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 640.

**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 410.

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.

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.

**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 Type the name of a TASKID or a specify a wildcard pattern and select Edit a TASKID. Then, press Enter.

The Edit TASKID panel is displayed.

4 You can change the work ID name or work ID name template, application name, and comments. Then, press END.

**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 410.

**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 616 and “Starting a CM/PILOT worklist over” on page 618.

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 611.

6 Select the method of executing the worklist. For more information about worklists, see “Viewing the execution status of a task ID” on page 613.

- 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 Getting Started Guide.

---

**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 162 or “**Executing an alter-type worklist**” on page 537.

**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.
Executing a task ID

**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 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 168. 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 539.

Figure 168 Creating additional job steps in the Execution interface
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 348.

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.
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.

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 169.

---

**Figure 169  TASKID Execution Status panel**

---
Table 45 describes the fields on the TASKID Execution Status panel.

### Table 45  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>
</tbody>
</table>
| Sequence    | sequence numbers of the sync points in the CM/PILOT worklist  
Sync points are identified by the -SYNC command in the worklist.                         |
| Status      | status of the CM/PILOT worklist processing job  
The following statuses are valid for the job:                                 |
|             | ■ Exec Strt (Execution started)  
■ Exec Comp (Execution complete)                                                            |
| Step        | the CM/PILOT worklist command that is in effect                                                                                             |
| Sync Point  | the text of the -SYNC command in the CM/PILOT worklist                                                                                     |
| TASKID      | the name of the task ID that is associated with the CM/PILOT worklist                                                                          |

---

**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 643 and “Listing task IDs for an application” on page 646.

---

**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

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 Getting Started Guide.

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.

---

**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 643 and “Listing task IDs for an application” on page 646.
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 618.

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 613.

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 604 and “Listing task IDs for an application” on page 646.
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 46 shows the situations in which you can restart CM/PILOT worklist processing.

### Table 46  Restarting processing of a CM/PILOT worklist

<table>
<thead>
<tr>
<th>Task ID Status</th>
<th>Work ID Status</th>
<th>Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec Strt</td>
<td>Created</td>
<td>Yes</td>
</tr>
<tr>
<td>(execution started)</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></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>
<tr>
<td>Exec Comp</td>
<td>Created</td>
<td>No</td>
</tr>
<tr>
<td>(execution completed)</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 170. 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 613.

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.

Figure 170  AEXIN input stream for restarting a worklist

```
//AEXIN       DD *
SSID DBAH WORKID RDAMAF2.DMLMIG
DASDDOPT DS611R25  RESTART
```

NOTE
If you are using a work ID name template, the last work ID that was created using the template is used in the restart processing.
Table 47 shows the situations in which you can start the CM/PILOT worklist processing over.

**Table 47  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></td>
<td>Exec Strt</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Exec Comp</td>
<td>Yes</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 171. Selecting **Start Over** for **Worklist Execution** on the TASKID Interface panel inserts this keyword into the AEXIN input stream.

**Figure 171  AEXIN input stream for starting over a worklist**

```
//AEXIN   DD *
SSID DEBA WORKID RDAMAF2.DMLMIG
DASDDOPT 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 172). 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 172  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 173), which is displayed when you create a new Task ID.
Using scripts

Figure 173   Script Selection List panel

AUCFSCPL ---------------------- Script Selection List -------------------------
Command ===>

Select a Script to create a TASKID. Then press Enter.
  _  1. Change data structures using a Compare trigger
  _  2. Change data structures using a DML trigger
  _  3. Migrate data structures using a DML trigger
  _  4. Change data structures using an existing WORKID
  _  5. Migrate data structures using an existing WORKID
  _  6. Receive data structure changes (strong sender)
  _  7. Receive data structure changes (strong receiver). create Baseline
  _  8. Receive DDL to create data structures
  _  9. Create a Full-Recovery Baseline
  _ 10. Recover data structures with current data
  _ 11. Recover data structures with old data
  _ 12. Select a user defined script
  _ 13. Replicate a WORKID
  _ 14. Delete one or more WORKIDs

Type an user defined script. or type a wildcard pattern for a selection list
if script is 12.
  Script . . .

Commands: HELP END CANCEL

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 611.

Table 48 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 48   Script tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating a script”</td>
<td>page 622</td>
</tr>
<tr>
<td>“Listing scripts”</td>
<td>page 633</td>
</tr>
<tr>
<td>“Editing a script”</td>
<td>page 634</td>
</tr>
<tr>
<td>“Listing the steps for a script”</td>
<td>page 635</td>
</tr>
<tr>
<td>“Executing a user-defined script”</td>
<td>page 636</td>
</tr>
<tr>
<td>“Browsing a script”</td>
<td>page 636</td>
</tr>
<tr>
<td>“Deleting a script”</td>
<td>page 637</td>
</tr>
<tr>
<td>“Creating a task ID from a script”</td>
<td>page 638</td>
</tr>
<tr>
<td>“Listing task IDs for a script”</td>
<td>page 639</td>
</tr>
<tr>
<td>“Copying a script supplied by BMC”</td>
<td>page 639</td>
</tr>
<tr>
<td>“Copying a user-defined script”</td>
<td>page 640</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 624.
■ To create a comparison step, see “To create a comparison step” on page 625.
■ To create an import step, see “To create an import step” on page 627.
■ To create an analysis step, see “To create an analysis step” on page 629.
■ To create an execution step, see “To create an execution step” on page 630.
■ To create a baseline report step, see “To create a baseline report step” on page 631.
■ To create a replicate work ID step, see “To create a replicate work ID step” on page 632.

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 623).

The Baseline Step Options panel is displayed, as shown in Figure 174.

Figure 174 Baseline Step Options panel

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 175.
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 623).

The Compare Step Options panel is displayed, as shown in Figure 176.
Figure 176 Compare Step Options panel

AUCFSTCO ---------------------- Compare Step Options --------------------------
Command =>

Script . . . . . . : RDACRJ.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 177.
Figure 177  Step Migrate Profile Options panel

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 623).

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 178.
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 179.
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 623).

The Analysis Step Options panel is displayed, as shown in Figure 180.
Figure 180  Analysis Step Options Panel

[Diagram of 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 623).

The Execution Step Options panel is displayed, as shown in Figure 181.
Creating a script

Appendix A Using scripting tools to automate change management

Figure 181 Execution Step Options panel

| AUCFSTEO -------------------- Execution Step Options -------------------- |
|-------------------------------|--------------------------|
| Command ==>                   |                          |
| Script . . . . . : RDACRJ.SCR01|
| Step Number . . . : 1         |                          |

Select execution step options. Then press Enter.

Dataset Sizing

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

Pre-Execution Baseline
1. Do not build a baseline before execution
2. Build a baseline before execution

Post-Execution Compare
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. 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 623).

The Baseline Report Step Options panel is displayed, as shown in Figure 182.

Appendix A Using scripting tools to automate change management 631
Creating a script

Figure 182  Baseline Report Step Options panel

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 623).

   The Replicate WORKID Step Options panel is displayed, as shown in **Figure 183**.
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.

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.
**Editing a script**

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.
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.

**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 622.

**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 633.
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 602.

   **NOTE**
   In step 4 on page 602, 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 605.

   **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.

---

**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 633.

---

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 delete a script by selecting **List Scripts** on the Script Action Menu panel. For more information, see “Listing scripts” on page 633.
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 410.

   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 639.
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 622 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 184.

**Figure 184   Application Action Menu**

Table 49 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 49   Application tasks**

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Creating an application”</td>
<td>page 642</td>
</tr>
<tr>
<td>“Listing applications”</td>
<td>page 643</td>
</tr>
<tr>
<td>“Editing an application”</td>
<td>page 643</td>
</tr>
<tr>
<td>“Browsing an application”</td>
<td>page 644</td>
</tr>
<tr>
<td>“Deleting an application”</td>
<td>page 645</td>
</tr>
<tr>
<td>“Creating a task ID from an existing application”</td>
<td>page 645</td>
</tr>
</tbody>
</table>
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 603.

   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 643.

---

**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 643.
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 643.

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 602.

**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 646.

### 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 643.
Appendix B Using utilities with ALTER and CHANGE MANAGER

This appendix presents the following topics:

Overview ............................................................... 648
Using the unload utilities ......................................... 648
  Using the BMC BASIC UNLOAD utility .................. 649
  Using the BMC UNLOAD PLUS utility ..................... 650
  Using the IBM UNLOAD utility .............................. 660
  Using the BMC RECOVER PLUS utility .................. 663
Unloading Unicode, LOB, and XML data .......................... 665
Using the load utilities ........................................... 665
  Using the BMC LOADPLUS utility ......................... 666
  Using the IBM LOAD utility ................................. 668
  Loading Unicode, LOB, and XML data .................... 668
Migrating tables with ROWID columns ......................... 669
Using the check utilities ......................................... 669
  Using the BMC CHECK PLUS utility ....................... 669
  Using the IBM CHECK DATA utility ....................... 669
Using the reorg utilities ......................................... 670
  Using the BMC REORG PLUS utility ....................... 671
  Using the IBM REORG utility ............................... 672
  Generating SHRLEVEL reorgs ............................... 672
  Considerations for using the utilities .................... 674
Using the recover and rebuild utilities ....................... 676
  Using the BMC RECOVER PLUS utility ................... 676
  Using the IBM REBUILD INDEX utility ................... 676
  Considerations for using the utilities .................... 677
Using the copy utilities ......................................... 677
  Using the BMC COPY PLUS utility ....................... 678
  Using the IBM COPY utility ................................. 678
  Copying and registering the data sets ..................... 679
  Creating image copies ....................................... 680
  Dynamically allocating copy data sets .................... 683
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 185). 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.
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.

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.”
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 or 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 187).

Migrating tables with ROWID columns

For information about migrating tables with ROWID columns, see “Migrating tables with ROWID columns” on page 650.

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 50.

**Table 50  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>SYLB\textsuperscript{nn}</td>
<td>references a file reference data set that contains LOB data (nn) represents a sequence number for each LOB column.</td>
</tr>
<tr>
<td>SYXC\textsuperscript{nn}</td>
<td>references a file reference data set that contains XML data (nn) 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 186), 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 186 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 actual data in the column is less than 32 MB.
  
  — The LOBGT32M keyword is included in the ALUIN input stream.

  You must manually insert the LOBGT32M keyword into the input stream.

- 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 518.

- 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.

- The BMCSTATS utility does not collect statistics on auxiliary table spaces or indexes. If you select the Use BMCSTATS to update BMC DASD MANAGER PLUS tables and the DB2 Catalog option on the Analysis Statistics Options panel (see Figure 192 on page 686), Analysis uses the IBM RUNSTATS utility to collect statistics on auxiliary objects and to update the DB2 catalog.
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 187), you can select to use the dynamically allocated data sets and to override any JCL Generation options that apply to dynamic allocation.

**Figure 187 Analysis Utility Dataset Options panel**

```
ALUFOASB --------------- Analysis Utility Dataset Options -------------------------
Command ===> 

Type information. Then press Enter to continue or PF12 for previous panel.

Specify the maximum number of work datasets you want BMC LOADPLUS and BMC REORG PLUS to use to build indexes.

MAXSYSUT . . 20   (1 - 9999)

Dynamic Dataset Allocation Options
  _ Use Dynamic SYSRE datasets
  _ Use Dynamic COPY datasets  (Overrides use of utilities to create copies)
  _ Override Defaults for Dynamic Allocation

Commands:  HELP END PREVIOUS
```

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 Getting Started Guide.

**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 multitask 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 652.

- 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 188). 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 188 Analysis Unload / Copy Options panel—UNLOAD PLUS](ALUFAN7)
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 51.

Table 51  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)</td>
<td></td>
</tr>
<tr>
<td>cataloged to an integrated catalog facility (ICF)</td>
<td>DSVOLSER = blank</td>
</tr>
<tr>
<td>cataloged to an integrated catalog facility (ICF)</td>
<td></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</td>
<td>STYPE = blank</td>
</tr>
<tr>
<td>copy)</td>
<td></td>
</tr>
<tr>
<td>image copies that are marked as</td>
<td>SHRLEVEL = R</td>
</tr>
<tr>
<td>SHRLEVEL REFERENCE or SHRLEVEL CHANGE, or SHRLEVEL</td>
<td>or</td>
</tr>
<tr>
<td>only must exist</td>
<td>SHRLEVEL = C</td>
</tr>
<tr>
<td>relative version number of the image copy must exist</td>
<td></td>
</tr>
<tr>
<td></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 188):

- 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 51. 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 187**), 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 Getting Started Guide**.

**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 or 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 52.

<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>SYLBBnn</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>SYXCNnn</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 650.

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 51.

When you select BMC RECOVER PLUS from ICs to analyze your migrate-type work ID (see Figure 185 on page 649), 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 51. 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 189). The relative image copy number of 0 is always the latest version.

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 189 on page 664). 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.
Unloading Unicode, LOB, and XML data

ALTER and CHANGE MANAGER can use the unload and load utilities shown in Table 53 to process Unicode, LOB, and XML data.

### Table 53 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></td>
<td>IBM UNLOAD</td>
<td>IBM LOAD</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 185 on page 649). 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 187 on page 656). 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 SUTnnnn 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.
For information about using the LOB DATA MOVER program to load data contained in LOB columns, see “Unloading data in XML and LOB columns” on page 652.

**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 53 on page 665 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 650.

Using the check utilities

You can select which check data utilities to use from the Analysis Utility Options panel (see Figure 185 on page 649). 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 190). You can choose from the following utilities:

- BMC REORG PLUS
- IBM REORG

**Figure 190  Analysis Reorg Options panel**

![Figure 190 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 187 on page 656). The number of data sets are specified in the MAXSYSUT keyword in the ALUIN input stream.
**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 SUTnnnn0 DD statements in the JCL.

---

**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.
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.

Table 54 summarizes SHRLEVEL options for different versions of DB2.

<table>
<thead>
<tr>
<th>Object</th>
<th>Version of DB2</th>
<th>Reorg utility</th>
<th>SHRLEVEL options</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>8, 9</td>
<td>IBM or BMC</td>
<td>CHANGE, REFERENCE, or NONE</td>
</tr>
<tr>
<td>index table space</td>
<td>10</td>
<td>IBM</td>
<td></td>
</tr>
<tr>
<td>LOB index</td>
<td>8</td>
<td>IBM or BMC</td>
<td>CHANGE or REFERENCE</td>
</tr>
<tr>
<td></td>
<td>9, 10</td>
<td>IBM</td>
<td></td>
</tr>
<tr>
<td>LOB table space</td>
<td>8</td>
<td>not applicable</td>
<td>No reorg. The product generates commands to drop and re-create the table space.</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>IBM or BMC</td>
<td>REFERENCE</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>IBM</td>
<td>CHANGE or REFERENCE</td>
</tr>
</tbody>
</table>

The product can generate the REORG command with the AUX YES parameter on the base table space.
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) parameter in the -BMCR or the -REOR command in the worklist. JCL Generation creates a WRK100n DD statement in the JCL. To specify information about the WRK100n or SYSUT data sets, see the *ALTER and CHANGE MANAGER for DB2 Getting Started Guide*.

- 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.
Considerations for using the utilities

- **(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 NOAMSDELETEI 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 185 on page 649). 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 191). You can choose from the following utilities:

- BMC COPY PLUS
- IBM COPY

Figure 191  Analysis Image Copy Options panel

ALUFDAN5 ---------------- Analysis Image Copy Options --------------------------
Command ===>

Type information. Then press Enter to continue or PF12 for previous panel.

Copy . . . 1 1. IBM COPY 2. BMC COPY PLUS 3. No copy 4. No copy start force
Copy DDN1 . . 1 Copy DDN2 . . 3 1. Copy 2. Copy no register 3. No copy
Recv DDN1 . . 3 Recv DDN2 . . 3 1. Copy 2. Copy no register 3. No copy
Pre-drop Copy 2 1. Build an image copy before drops or reorgs 2. Do not build an image copy before drops or reorgs

Utility Copy
1 1. Use Copy selection specified above to build image copies.
2. Other utilities should build image copies when possible.

Select other copy options
_ PARTCOPY  Make partition level copies using BMC utilities
Using the BMC COPY PLUS utility

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. 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 191), 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.

- 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.

**NOTE**

If the reorg is a two-phase reorg (for example, REORG UNLOAD PAUSE / VALUE / REORG RESTART), the control cards to REORG must match exactly on each phase, including the COPYDDN and RECOVERYDDN cards.

- **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.)

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 55 lists the naming conventions for the primary and backup copies.

**Table 55**  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>Cxxxyyyy</td>
</tr>
<tr>
<td>Local backup</td>
<td>SYCRnnnn</td>
<td>Dxxxyyyy</td>
</tr>
<tr>
<td>Recovery primary</td>
<td>RECLnnnn</td>
<td>Exxxyyyy</td>
</tr>
<tr>
<td>Recovery backup</td>
<td>RECRnnnn</td>
<td>Fxxxyyyy</td>
</tr>
</tbody>
</table>

In Table 55, *nnnn* represents a sequential number for the table space or index. The variable *xxx* represents a sequential number that pertains to the table space or index that is copied, and *yyyy* represents the partition number for copies made by COPY PLUS.

When you select **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.

**NOTE**
If you do not select **Make partition level copies using BMC utilities**, the products use the nonpartitioned object ddname.
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 187 on page 656), 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 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 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 Getting Started Guide.

NOTE
Before you select Override Defaults for Dynamic Allocation, you must select 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 a&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.

- 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.
Dynamically allocating copy data sets

- 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 56 lists the OUTPUT descriptors that the products use for dynamic allocation.

| Table 56  Naming conventions for OUTPUT descriptors |
|------------|-----------------|
| **Type of copy** | **OUTPUT descriptor** |
| Local primary | DCPYLOCP |
| Local backup | DCPYLOCB |
| Recovery primary | DCPYREMP |
| Recovery backup | DCPYREMB |

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 57 lists the TEMPLATE descriptors that are used for dynamic allocation.

| Table 57  Naming conventions for TEMPLATE descriptors |
|------------|-----------------|-----------------|
| **Type of copy** | **TEMPLATE descriptor for primary data set** | **TEMPLATE descriptor for alternate data set** |
| Local primary | DCPYLOCP | SCPYLOCP |
| Local backup | DCPYLOCB | SCPYLOCB |
| Recovery primary | DCPYREMP | SCPYREMP |
| Recovery backup | DCPYREMB | SCPYREMB |
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 192).

Figure 192 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.

- other BMC utilities (such as BMC REORGL 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 BMC 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 BMC 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.
Updating statistics

- 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 **Include IBM RUNSTATS History** 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 ................................................................. 691
Compiling SLIBs ......................................................... 692
Changing SLIBs ......................................................... 693
  Testing changes using ISPF file tailoring ...................... 693
  Compiling changed SLIBs ........................................ 694
Processing SLIBs ..................................................... 695
  Generating the SLIB report ....................................... 695

Overview

You can improve the performance of the JCL construction phase of JCL Generation by using the 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 193 illustrates the processing flow of the SLIB compiler.
This appendix describes the compiler, how to test SLIBs before compiling them, and the compiler’s associated runtime unit.

### Compiling SLIBs

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 changes by using standard ISPF file tailoring before you compile the SLIBs. For more information, see “Testing changes using ISPF file tailoring” on page 693.

### 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.

### 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 695.

**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.

   \[ \text{LRECL}=80, \text{RECFM}=\text{FB}, \text{BLKSIZE}=6160, \text{DSORG}=\text{PS} \]
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 194** shows a sample runtime report.

**Figure 194  Sample runtime report (part 1 of 2)**

<table>
<thead>
<tr>
<th>Skelname</th>
<th>Usage</th>
<th>Compile Type</th>
<th>Compile Date</th>
<th>Compile Time</th>
<th>Usage Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJX$ACMX</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.16</td>
<td>1</td>
</tr>
<tr>
<td>AJXJOB0</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.46</td>
<td>1</td>
</tr>
<tr>
<td>AJX#USRV</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.14</td>
<td>1</td>
</tr>
<tr>
<td>AJXJOB5</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.46</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEP1</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.58</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEP7</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.59</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTEPU</td>
<td></td>
<td>Compiled</td>
<td>08/28/2009</td>
<td>07.08</td>
<td>1</td>
</tr>
<tr>
<td>AJXSYSX$</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>06.02</td>
<td>1</td>
</tr>
<tr>
<td>AJXSTWK0</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>06.00</td>
<td>1</td>
</tr>
<tr>
<td>AJXISPFM</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.45</td>
<td>1</td>
</tr>
<tr>
<td>AJXCCLIBU</td>
<td></td>
<td>Compiled</td>
<td>09/26/2009</td>
<td>18.12</td>
<td>1</td>
</tr>
<tr>
<td>AJXMLIBU</td>
<td></td>
<td>Compiled</td>
<td>09/26/2009</td>
<td>18.12</td>
<td>2</td>
</tr>
<tr>
<td>AJXISPF5</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.45</td>
<td>1</td>
</tr>
<tr>
<td>AJXSLIBU</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.57</td>
<td>1</td>
</tr>
<tr>
<td>AJXTLIBU</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>06.02</td>
<td>1</td>
</tr>
<tr>
<td>AJXPILIBU</td>
<td></td>
<td>Compiled</td>
<td>09/26/2009</td>
<td>18.12</td>
<td>1</td>
</tr>
<tr>
<td>AJX#PRNT</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.12</td>
<td>1</td>
</tr>
<tr>
<td>AJXNOSTS</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.50</td>
<td>1</td>
</tr>
<tr>
<td>AJXWORK0</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>06.06</td>
<td>1</td>
</tr>
<tr>
<td>AJXWORK1</td>
<td></td>
<td>Compiled</td>
<td>10/11/2009</td>
<td>22.11</td>
<td>7</td>
</tr>
<tr>
<td>AJXWKUNT</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>06.06</td>
<td>7</td>
</tr>
<tr>
<td>AJXMTAP</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.12</td>
<td>13</td>
</tr>
<tr>
<td>AJXESTIM</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.43</td>
<td>10</td>
</tr>
<tr>
<td>AJX#DNS</td>
<td></td>
<td>Compiled</td>
<td>10/04/2009</td>
<td>05.10</td>
<td>9</td>
</tr>
</tbody>
</table>
The report summary at the end of Figure 194 provides the information shown in Table 58.

### Table 58  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>
Using worklist parallelism

This appendix presents the following topics:

Overview of worklist parallelism .................................................. 699
Considerations for enabling worklist parallelism .......................... 701
Controlling XIM execution ......................................................... 704
  Determining the status of XIM .................................................. 705
Starting XIM .............................................................................. 706
Inactivating XIM initiators ......................................................... 707
Shutting down XIM ................................................................. 707
Activating XIM initiators ......................................................... 708
Modifying MVS image variables .............................................. 708
Troubleshooting the execution of XIM .................................... 710
Processing a worklist in parallel ............................................. 711

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 195 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.
Considerations for enabling worklist parallelism

You should consider the following items before you enable worklist parallelism:

- XIM must be started 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 //AEXPRINT DD to SYSOUT.

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

- 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. In addition, you must not specify a threshold value for tape for these 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
Considerations for enabling worklist parallelism

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 Getting Started Guide. 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.

For information about installing and customizing the solution and the worklist parallelism feature, see the Database Administration for DB2 Installation Guide. For information about restricting access to the feature, see the ALTER and CHANGE MANAGER for DB2 Getting Started Guide.
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 59 describes authorization for XIM.

Table 59 Authorizing XIM procedures

<table>
<thead>
<tr>
<th>Product</th>
<th>Task</th>
</tr>
</thead>
</table>
| RACF or CA Top Secret | Authorize the procedures for the following subsystems as started tasks in the started procedures table:  
  - XIM performance subsystem  
  - XIM extended job entry subsystem  
  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. |
| CA ACF2             | Authorize the procedures for the following subsystems as started tasks under the started task control:  
  - XIM performance subsystem  
  - XIM extended job entry subsystem  
  For more information, see the appropriate CA ACF2 publication. |
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 196) shows the result of issuing the STATUS command where XIM is active.

**Figure 196  Result of STATUS command**

<table>
<thead>
<tr>
<th>BMC95100I</th>
<th>XIM STATUS Command Accepted, XIM STATUS in progress XIMACM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC95181I</td>
<td>STATUS, 3 XIM Members(s) ACTIVE in XIM Group XIMACM XIMACM</td>
</tr>
<tr>
<td>BMC95159I</td>
<td>Jobname  Jobid  Smfid  Cvtsname  Status</td>
</tr>
<tr>
<td>BMC95184I</td>
<td>XIMACM  STC01000  DB2A  DB2A  Active  16 inits  0 active</td>
</tr>
<tr>
<td>BMC95184I</td>
<td>XIMACM  STC08798  SYSN  SYSN  Active  16 inits  0 active</td>
</tr>
<tr>
<td>BMC95184I</td>
<td>XIMACM  STC08638  DB2B  DB2B  Active  16 inits  0 active</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 Getting Started Guide.

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 197) shows the result of issuing the QUIESCE command.

Figure 197 Result of QUIESCE command

<table>
<thead>
<tr>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC95100I XIM QUIESCE Command Accepted, XIM QUIESCE in progress XIMACM</td>
</tr>
<tr>
<td>BMC95100I XIM STOP Command Accepted, Initiator termination in progress</td>
</tr>
<tr>
<td>BMC98522I Initiator shutdown request received in ASID(01F6). XJS1</td>
</tr>
<tr>
<td>BMC98212I 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 198, the name of the member is XIMACMI.
2 Modify the INITIATORS variable.

In the example shown in Figure 198, 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 Getting Started Guide.*

---

**Figure 198  XIMACMI member (part 1 of 2)**

* 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)
Save the changes to the member.

Inactivate XIM by issuing the QUIESCE command:

/F XIMACM,QUIESCE

Verify the status of XIM by issuing the STATUS command:

/F XIMACM,STATUS

Shut down XIM by issuing the SHUTDOWN command:

/F XIMACM,SHUTDOWN

Start XIM by issuing the start command:

/S XIMACM

The new instance of XIMACM uses the new parameters.

To verify the new parameters, issue the STATUS command:

/F XIMACM,STATUS

Review the values that XIM displays in the system log.

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 DYNUNLD parameters in the installation options module are set to Y, or the DYNCOPY and DYNUNLD keywords are specified in the ALUIN input stream.)
Processing a worklist in parallel

NOTE

If the No copy (NOCOPY ALUIN keyword) or No copy start force (NOCOPYFORCE ALUIN keyword) Analysis image copy option is selected, dynamically allocated copy data sets are not created or used.

If the BMC CHECK PLUS utility is used to ensure the validity of the data, JCL Generation allocates static SYSDISC data sets for use by the utility.

- 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 652.

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 60.

### Table 60 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>ALTER and CHANGE MANAGER for DB2 Getting Started Guide</td>
</tr>
<tr>
<td>Specifying Analysis options</td>
<td>“Overriding the default processing options” on page 62</td>
</tr>
<tr>
<td>Reviewing parallelism worklist commands</td>
<td>ALTER and CHANGE MANAGER for DB2 Reference Manual</td>
</tr>
<tr>
<td>Specifying Execution options</td>
<td>ALTER and CHANGE MANAGER for DB2 Getting Started Guide</td>
</tr>
<tr>
<td>Reviewing Execution JCL</td>
<td>“Execution JCL” on page 92</td>
</tr>
<tr>
<td>Reviewing the output from a parallel worklist</td>
<td>“Output for parallel worklists” on page 104</td>
</tr>
<tr>
<td>Restarting a worklist</td>
<td>“Restart methods” on page 107</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.

constraint
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.
edit 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).

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.

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.

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.

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 thatnonnull 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.

csync 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.

T
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

##### 174, 603
########### 242
$ change-level indicator 312
; (semicolon) as a delimiter 179, 238, 325
<BLANK> keyword 54
<CURRENT> keyword 52
<DEFLT> keyword 53
@ change-level indicator 312
@@@@@@@@ 174, 603

Numerics

-1, used as value for PRIQTY attribute 51
-1, used as value for SECQTY attribute 51

A

A change-level indicator 311
ACM_PARALLEL_WORKLST AJXPOFIN keyword 72
ACMDMLD1 member of HLQ.DBCNTL data set 418
ACMDMLD2 member of HLQ.DBCNTL data set 370
ACMDMLD3 member of HLQ.DBCNTL data set 372
ACMDMLL1 member of HLQ.DBCNTL data set 526
ACMDMLL2 member of HLQ.DBCNTL data set 528, 595
ACMDMLM1 member of HLQ.DBCNTL data set 134
ACMDMLM2 member of HLQ.DBCNTL data set 139
ACMDMLT1 member of HLQ.DBCNTL data set 451
ACMDMLT2 member of HLQ.DBCNTL data set 447
ACMDMLU1 member of HLQ.DBCNTL data set 530, 589
ACMDMLU2 member of HLQ.DBCNTL data set 523, 594
ACMDMLU3 member of HLQ.DBCNTL data set 423
ACMDMLU4 member of HLQ.DBCNTL data set 531, 589
ACMDMLU5 member of HLQ.DBCNTL data set 445
ACMDMLU6 member of HLQ.DBCNTL data set 416
ACMDMLU7 member of HLQ.DBCNTL data set 425
ACMDMLU8 member of HLQ.DBCNTL data set 396, 397
ACMDMLU9 member of HLQ.DBCNTL data set 479
ACMDMLUA member of HLQ.DBCNTL data set 481
ACMDMLUB member of HLQ.DBCNTL data set 473
ACMDMLUC member of HLQ.DBCNTL data set 453
ACMDMLUD member of HLQ.DBCNTL data set 492
ACMDMLUE member of HLQ.DBCNTL data set 422
ACMDMLUF member of HLQ.DBCNTL data set 493
ACTIVATE command 708
activating XIM initiators 708
adding a column to a table 397
additional job steps in JCL, creating 610
advisory REORG-pending (AREOR) status 670
AEXIN input stream, overview 155
AEXPRINT
  CHANGE MANAGER worklist 611
  CM/PILOT worklist 611
data set 104, 701
  used by Execution 103, 348
AEXPRnnn data set 106
AEXTRAC data set 106
AEXSTLOG data set 105
AEXSTnnn data set 106
AEXSYnnn data set 106
AJXIN input stream 84
AJXPOFIN input stream 62, 84
aliases, deleting 417
Allocation Unit, Space Estimation parameter 334
ALLSTATSUPD keyword 689
ALTER
  importing DDL 324, 376
  using to merge subsystems 559
ALTER parameter of INCLUDE keyword 250
alter process, workflow 307
alternate delimiter 179, 238, 325
alternate terminator 179, 238, 325
alter-type work ID
  converting to CDL 291, 324
  creating 352
alter-type worklist
  executing 537
  generating 535
  overview 342
ALUEUSX1 member 96
ALUIN input stream
  building 62
  overriding 62
  used by Analysis 648
ALUIN keywords
  baseline profiles 234
  baselines 237
  displaying change rules 252
  excluding objects 244
  including CDL statements 250
ALUIN keywords (continued)
   including dependent objects 251
   object attributes 246
   object change flags 244
   object owners 240
   outbound migrate profiles 235
   overriding change rules 241
   partition attributes 247
   resolving names 241
   scope types 236

ALUPRINT
   used by Analysis 343
   used by Baseline 187
   used by Baseline Report 189
   used by Compare 257
   used by Import 330

ALUXGRNT REXX executable 81
   -AMS command 65, 102

AMSDELETEI keyword 675

Analysis
   alter-type worklist 342
   ALUPRINT diagnostic output file 343
   analyzing change requests 340
   changing installation options 62
   including command types in a worklist 64
   inputs 59, 99
   locations, generating worklists for multiple 61
   migrate-type worklist 79
   override options 62, 341
   primary key propagation 68
   rebuilding a worklist 60
   requesting a full-recovery baseline 181
   updating catalog statistics 688
   using in change process 340

analyzing
   alter-type work ID 535
   change requests 535
   migrate-type work ID 60

application tables, modifying 584

applications
   browsing 644
   comments, adding 642
   comments, changing 635
   creating 642
   deleting 645
   editing 643
   listing 643
   listing task IDs for an application 646
   maintaining 551
   modifying 643
   overview 640
   planning to establish and maintain 552
   profiles, displaying in a task ID 608
   profiles, specifying 642
   specifying work ID name template 642
   strategies used in maintaining 551
   task ID, specifying 604
   assigning a unique name 129
   ASUTIME attribute 53
   attributes
      ASUTIME 53
      BUFFERPOOL 247
      changing 323, 655
      CHECK 247
      COMPRESS 248
      DEFINE 151, 246, 529, 532
      DROPRESTRICT 247
      DSSIZE 51
      FREEPAGE 51, 248
      IDENTITY 247
      LOCKMAX 53, 246
      LOCKSIZE 246
      NUMPARTS 247
      PATH 247
      PCTFREE 51, 248
      PIECESIZE 53, 240
      PRIQTY 51, 248
      SECQTY 51, 248
      SEGSIZE 51, 246
      specifying a range of values 51
      specifying blank values 54
      specifying default values 53
      specifying new values 52
      specifying nonnumeric values 53
      specifying numeric values 51
      specifying special numeric values 53
      specifying values 50
      START 247
      STOGROUP 247
      TRACKMOD 248

AU (Allocation Unit), Space Estimation parameter 339
   -AUTH command 95, 102, 103, 180

authorization switching
   AUTHSW keyword 95
   delimited identifiers 97
   Global Authorization ID (GLID) 97
   modes 95
   used by Execution 94

authorizations
   browsing in a baseline 185
   commands 81
   DB2 and data set, XIM 704
   DB2 objects 184
   restoring in baselines 178
   shown in a baseline report 325
   shown in a DDL file 189
   AUTHSW installation option 95
   AUTHSW keyword 95
   AUTHSWGLID keyword 95, 98
   AUTHSWOFF keyword 95
   auxiliary indexes
      creating 313
      dropping 314
auxiliary objects
  building stand-alone image copies 680
  creating 511
  creating for a base table 494
  creating for a table space 313
  creating in a nonpartitioned table space 513
  creating in a single partition 506
  creating in partitioned table spaces 500
  creating scope rules 43
  defining attributes 313
  displayed in Specification 313
  dropping 46
  editing 46
  liking 46
  migrating 43, 46, 59, 121
  naming 314
  propagating name changes 58
  reviewing 505, 509, 516
  undoing changes 506, 510, 517
auxiliary table spaces
  creating 313
  dropping 314
auxiliary tables
  creating 313
  dropping 314
Avg Len, Space Estimation parameter 337
Avg Row Length, Space Estimation parameter 333
AX (Auxiliary Object LOB) action code 310

B
backup copy, DD name 682
  -BASE command 182, 186
base table spaces, building stand-alone image copies 680
base tables
  creating 494
  creating a clustering, partitioning index 456
  creating auxiliary objects 494, 511
  liking 511
  migrating 121
  undoing changes 315
baseline report
  authorizations 325
  creating a step in a script 631
  DDL 189
  generating 209
  overview 187
baseline tables 184
baseline to DDL file comparison 288
baselines
  ALUPRINT diagnostic output 187
  application milestone 171
  browsing 185
  checking contents 168
  combining pre-execution baselines 346
  complete 191
  creating a profile 196
  creating a step in a script 624
  creating a structure-only baseline 199
  data sets 187
  definition 168
  deleteage 175
  deleting 191
  deleting, automatically 210
  deleting, explicitly 194, 213
  deleting, using a baseline profile 212
  deleting, using DELETEAGE 192
  deleting, using RETAINMAX 192
  establishing 171, 547
  exception baselines 172
  full recovery, creating 201
  full recovery, overview 173
  full recovery, script to create 205
  full recovery, using 181
  improving performance 194
  incomplete 185, 191
  inputs 168
  managing 190
  migrate profile reference 175
  name template 174, 345, 548
  NoBaseline option 191
  object counts 208
  objects, browsing 185
  output 184
  overview 168
  periodic baselines 172
  post-Execution 610
  pre-Execution 610
  profile 173, 196, 234
  protecting baselines from deletion 190
  recovery point 169, 172, 184
  removing protection from baselines for deletion 190
  report 187, 209
  scope rules 176
  script to create a full recovery baseline 205
  specifying 608
  structure and data recovery points 178
  structure-only, creating 178, 199
  structure-only, overview 172
  structure-only, using 178
  tables 184
  template 174
  types 178
  unload data sets 182, 187
  unprotecting baselines for deletion 190
  using as the scope of a comparison 237
  using in change migration process 547
  using in compare process 226
  using in data structure recovery process 171
  using in old data recovery process 171
  using in structure recovery process 171
  viewing information 210
  work ID 186
baselines (continued)
  workflow 169
  worklist 180
baseline-to-baseline comparison 226, 278
baseline-to-catalog comparison 228, 281
BASICUNLOAD keyword 649
Batch Execution JCL Generation 87
-BIND command 66
BINDAUTH option 66
<BLANK> keyword 54
BLNAME keyword 181
BLPROFILE keyword 181
BLRECOVER keyword 184
BLRECOVERPOINT keyword 181, 184
BLRP data set 187
BLWORKID keyword 186, 345
BMC Software utilities
  BASIC UNLOAD 649
  BMCSTATS 110, 687
  BMCTRIG 110
  CHECK PLUS 110, 669
  COPY PLUS 110, 678
  LOADPLUS 110, 666
  RECOVER PLUS 110, 663, 676
  REORG PLUS 110, 671
  restarting 110
  starting over 112
  UNLOAD PLUS 110, 666
BMC Software, contacting 2
-BMCC command
  processing in parallel 678
  using in a parallel worklist 712
BMCHECK keyword 669
BMCCOPY keyword 678, 680
-BMCD command
  processing in parallel 650
  using fast unload 652
  using in a full-recovery baseline 182, 186
  using in a parallel worklist 712
  using with -UNLB command 653
BMCFASTL installation option 652, 667
BMCFASTLOAD keyword 652, 667
-BMCK command
  processing in parallel 669
  using in a parallel worklist 712
-BMCL command
  FORMAT BMCUNLOAD 667
  processing in parallel 666
  REDEFINE NO parameter 666
  specifying multiple SYSUT data sets 667
  using in a parallel worklist 712
  using with -LOLB command 654
-BMCR command 672
BMCREBUILD keyword 676
BMCREORG keyword 671
-BMCS command
  generating statistics 687
  processing in parallel 687
  using in a parallel worklist 712
BMCSPACE keyword 445
BMCSTATS utility
  collecting statistics 687
  performing space estimation 331
  restarting 110
  using in a parallel worklist 712
BMCSTATSUPD keyword 689
BMCTRIG utility 110
BMCU NLOAD keyword 650
BMW2DB2I8 member of HLQ.UDBCNTL 588
BROWSE (browse diagnostics) command 610
  applications 644
  baselines 185
  DB2 authorizations 185
  DML statements 615
  scripts 636
  task IDs 614
BUFFERPOOL attribute 247

C

CA ACF2, authorizing procedures for XIM 704
CA Top Secret, authorizing procedures for XIM 704
calculating data set size, dynamically allocated unload data sets 658
catalog
  comparing 219
  maintaining catalog information for data modeling tools 557
  merging subsystems 559
catalog baseline 178
catalog baseline profiles
  creating 196
  creating scope rules 176, 197
  used as scope 234
catalog baseline-to-catalog baseline comparison 226, 278
  catalog baseline-to-catalog comparison 228, 281
  catalog baseline-to-DDL file comparison 230
  catalog baseline-to-worklist comparison 230
catalog indirection, overview 60
CATALOG MANAGER Drop Recovery 90
catalog statistics, suppress updating 688
cataloged procedures 94
catalog-to-baseline comparison 285
catalog-to-catalog baseline comparison 229
catalog-to-catalog comparison
  considerations 221
  defining scope 239
  defining scope for a remote location 261
  handling unresolved owner or object 242
  improving performance 245
catalog-to-catalog comparison (continued)
  input 239
  performing 267
  requirements 221
  resolving changes to names 241
  using a script 294
  using an outbound migrate profile 242
catalog-to-DDL file comparison 222, 271
catalog-to-worklist comparison 222
CATAUDIT keyword 100
CATRECOVER keyword 90, 100
CD tables, used by Import 323
CDL (Change Definition Language)
  catalog information 250
  CDLDEP parameter 251
  comparison report 258
  evaluating 551
  excluding command types and catalog information 249
  generating changes 548
  generating from a comparison 253
  generating from a work ID 291
  generating multiple CDL files using locations 249
  handling unresolved owner or object name in CDL 242
  importing in CHANGE MANAGER 374
  including command types and catalog information 249
  override options 251
  overview 324
  recording changes to objects 257
  used by Compare 257
  used by Import 324
  using in change feedback process 555
  using in change migration process 548
-CDL command 250
CDLCHANGERULES keyword 248, 258
CDLDEP parameter of INCLUDE keyword 251
Change Definition Language. See CDL.
change feedback process 554
CHANGE MANAGER
  importing CDL 374
  importing DDL 374
  importing DML 374
  override options 608
change migration process
  acting as receiver 549
  acting as sender 547
  establishing control points 551
  implementing changes 551
  receiving DDL 550
  using inbound migrate profiles 551
change propagation
  dropped table columns in indexes 78
  primary keys 68
  table columns in base views 69
  text 253
  text dependencies 77
change requests
  analyzing 535
  executing 537
change rules
  application order 55
  applying to aliases and synonyms 50
  applying to trigger text 329
  change-type 48
  comparisons 48
  exclude 48
  force 48
  generating a migrate worklist using change rules 154
  import 48
  include 48
  migration 47
  name resolution in a catalog to catalog comparison 242
  object types 48
  specifying values 50
  suppress 48, 248
  tailoring worklists 50
  types 48
  using in migrate-type work IDs 47
  using in outbound migrate profiles 40, 49
  using to set objects to DEFINE NO 151
change validation of text 329
change-level indicators, list of 311
CHANGERULESIN2 keyword
  avoiding use of automatic change rules 252
  determining when objects are renamed 252
  displaying change rules 242
  improving performance of comparison 260
  resolving names of objects and creators 223
CHANGERULESIN2 NULL keyword 242
change-type change rules 48
characters, nonprintable 55
CHECK attribute 247
check constraint text validation
  by Analysis 76
  by Compare 253, 329
check constraint text, change propagation 77
CHECK DATA utility
  checking table spaces 669
  using in a parallel worklist 712
CHECK PLUS utility
  checking table spaces 669
  restarting 110
  using in a parallel worklist 712
-CHEK command
  processing in parallel 669
  using in a parallel worklist 712
child tables 66
CHKSQNUM macro 82
choosing a script 620
CISIZE4K ALUIN keyword 71
cleanup, data set 94
cloning 91
clustering, changing for an index 392
clustering, partitioning index, creating for a base table 456
CM/PILOT scripts 349
CM/PILOT worklist
   creating 608
   executing 606
   failure, determining point of 613
   hash verification numbers 609
   processing 606
   processing status, viewing 613
   restarting execution 616
   starting execution over 607
CMP_STRUCTURE_CHG script 294
CMPIN keyword 237
CMPTYPE keyword 237, 240
collating sequence 43
columns
   adding to a table 397
   changing the data type 400
   changing the length 400
   copying 418
   creating a LOB 496
   creating a ROWID 495, 519
   dropping from a table 399
   editing in a unique constraint 385
   identity 322
   combining pre-execution baselines 346
commands
   BROWSE (browse diagnostics) 610
   DROPALL 46, 311
   MIGALL 46
   PE (propagate estimates) 334, 338
   UNDOALL 46, 311
   VIEW (view task ID execution status) 610
   ZOOM 45
commands, worklist
   -AMS 65, 102
   -AUTH 95, 102, 180
   -BASE 182, 186
   -BEGU 102
   -BIND 66
   -BMCD 186
   -BMCL 667
   -BMCR 672
   -BMCS 687
   -CDL 250
   -DATA 65
   -GLID 95
   -ISMT 70, 183
   -JCLP 187, 667, 672
   -MICR 79
   -REXC 81
   -RNST 687, 688
   -SETP 102
   -SETS 95, 102
-SQL 64, 102
-STOP 103
-SYNC 103
-UNRC 182, 183, 186, 187
-UOWC 103
-WKID 98
comments 179, 238, 325
COMMIT keyword 195
COMMITFREQ keyword 654
COMMITTSIZE keyword 654
comparisons
   ALUPRINT diagnostic output 257
   application versioning 225
   CDL 253
   CDLDEP parameter 251
   creating a step in a script 625
   data modeling tool, using 222
   data structure, post-execution 610
   data structure, pre-execution 610
   displaying a list of objects 240
   displaying old and new values 258
   generating CDL from a work ID 258, 324
   generating multiple CDL files using locations 249
   handling unresolved owner or object in CDL 242
   improving performance 245, 259
   inputs 219
   name resolution 242
   name template 345
   ordering inputs 219
   outputs 256
   overriding options 243
   profiles used 242, 248
   propagating dependent object changes 251
   rename 237
   rename table 550
   renamed object resolution 254, 256
   report 258
   rolling back changes 229
   scope 232, 234
   setting options 243
   specifying a table owner 240
   specifying information included in a report 258
   specifying locations 240
   specifying what to include in the CDL file 250
   synchronizing objects 219
   text validation 253
   using a CM/PILOT script 294
   using a DB2 catalog 220, 239
   using a name template in a multi-step batch job 345
   using a profile to scope 267
   using baselines 226
   using DDL files 238
   using in change migration process 548
   using worklists 235
   work ID, name template 345
   workflow 217
comparisons, baseline
  baseline to baseline 226
  baseline to catalog 228
  baseline to DDL file 288
  baseline to DDL file or worklist 230
  catalog baseline to a catalog 281
  catalog baseline to catalog 228, 281
  catalog baseline to catalog baseline 278
  catalog baseline to DDL file 230
  catalog baseline to worklist 230
  DDL baseline to DDL file 230
  DDL baseline to worklist 230
  local baseline to remote baseline 278
comparisons, catalog
  catalog to baseline comparison 285
  catalog to catalog 219
  catalog to catalog baseline 229
  catalog to DDL file 222, 223, 225, 271
  catalog to worklist 222
  local catalog to a remote catalog 261
comparisons, DDL file
  baseline to DDL file 288
  DDL file to DDL file 224, 274
  DDL file to worklist 224, 274
comparisons, worklist
  baseline to DDL file or migrate-type worklist 288
  worklist to DDL file 224, 274
  worklist to worklist 224, 274
compiling SLIBs 694
COMPRESS attribute 248
Compress, Space Estimation parameter 334, 338
constraints
  creating a primary 379
  creating a unique constraint 381
  editing 385
control characters
  using in change rules 55
  using in DDL files 179, 237
  using in limit keys 459
control interval size, used in VSAM data set creation 71
control system, feeding changes back 554
controlling access to XIM 704
conventions, documentation 24
converting data
  changing attributes 323
  VARCHAR to BLOB or CLOB data types 655
converting table spaces 315
converting to LOB data types 323, 655
converting VCAT-defined to STOGROUP-defined partitions 424
copy data sets, dynamically allocating 683
COPY PLUS utility
  creating partition-level image copies 682
  restarting 110
  specifying 678
  using in a parallel worklist 712
copy utilities 677
COPY utility 678
COPYDDN keyword 679
copying
  BMC-supplied scripts 639
  columns 418
  user-defined scripts 640
CREATE parameter of INCLUDE keyword 250
creating
  alter-type work ID 352
  applications 642
  auxiliary objects 313
  baseline (strong receiver) 564
  data structure for a new application 406
  data structure, using specification 403
database 523
  DDL baseline 406
  inbound migrate profile 356
  JCL for processing CM/PILOT worklist 609
  migrate-type work ID 115
  MQT from a view 426
  primary constraint 379
  receive-type work ID 154
  scripts 622
  synonyms 423
table space 523
task IDs 602
task IDs, from a script 638
task IDs, from an existing application 645
  unique constraint for a table 381
  worklists 608
<CURRENT> keyword 52
CURRENT RULES register 315
customer support 3
Cxxxyyyy DD name 682

D
D (Drop) action code 310
D change-level indicator 311
DASD MANAGER PLUS, updating with BMCSTATS 688
DASDMAN installation option 331
data
  migrating to a different subsystem 160
  migrating within the same subsystem 159
  unloading and loading LOB and XML 65
-DATA command 65
data conversions
  changing a column’s data type 400
  converting to LOB data types 323, 655
  using the UNLOAD PLUS utility with the
  LOADPLUS utility 666
Data Manipulation Language (DML), importing in
CHANGE MANAGER 374
data modeling tools
  importing DDL 556
  maintaining catalog information 557
data set disposition
restart processing 109
startover processing 112
data set sizing 84
data sets
AEXPRINT 104
AEXPRnnn 106
AEXPTRAC 106
AEXSTLOG 105
AEXSTnnn 106
AEXSYnnn 106
allocating 250
baseline, unload 187
CM/PILOT worklist 606
copying 679
JCL for processing CM/PILOT worklist 607
local 679
registering 679
remote 679
sizing 87, 89
SYSnnnnn 107, 113
SYSUDUMP 702
tape 668
UOWTRnnn 106
WRK100n 674
data structures
adding WITH RESTRICT ON DROP 421
altering 306
analyzing changes 340
changing 403
changing, using DML 410
copying a column 418
creating using specification 403
deleting aliases 417
deleting using specification 403
falling back with current data 569
falling back with old data 575
importing 324
importing for a new application 406
migrating 59
migrating changes 546
migrating to a different subsystem 158
migrating within the same subsystem 158
receiving changes 561, 564
receiving DDL 406
recovering structures and data 172
recovering with current data 570
recovering with old data 576
recovery process 171
recovery process, using full-recovery baselines 173
restoring with current data 569
restoring with old data 575
synchronizing 219
updating index parts 419
data structures and data
migrating to a different subsystem 160
migrating within the same subsystem 159
data types, changing for a column 400
Database Administration
password 653
worklist parallelism 699
databases
creating 523
creating implicitly 320
specifying <DEFLT> as the name 320
updating index parts 419
date formats
baseline template 174
work ID name template 603
DB2 catalog
used by Analysis 60
used by Baseline 178
used by Compare 239
used by Import 328
DB2 utilities
CHECK DATA 669
COPY 678
LOAD 110, 668
REBUILD INDEX 676
REOR G 110, 672
REPAIR 71
RUNSTATS 687
UNLOAD 660
DB2STATSUPD keyword 689
DBRMLIB keyword 66
DCPYLOCB OUTPUT descriptor 684
DCPYLOCB TEMPLATE descriptor 684
DCPYLOCP OUTPUT descriptor 684
DCPYLOCP TEMPLATE descriptor 684
DCPYREMB OUTPUT descriptor 684
DCPYREMB TEMPLATE descriptor 684
DCPYREMP OUTPUT descriptor 684
DCPYREMP TEMPLATE descriptor 684
DD name, copies 682
DDL
baseline 179
comment 179, 238, 325
importing 324, 376, 406
importing in ALTER 376
importing in CHANGE MANAGER 374
incremental DDL 325
receiving 406
scope 235
setting columns to NOT NULL 365
used by Baseline 179
used by Import 324
using a script to receive 406
using in a baseline report 189
using in change migration process 550
using in data modeling tools 556
DDL baseline to DDL file comparison 230
DDL baseline to worklist comparison 230
DDL file to DDL file comparison 224, 274
DDL file to worklist comparison 224, 274
<DEFLT> keyword 53
DEFAULTOFF keyword 252
DEFER YES parameter 71
DEFERUNIQUEIX keyword 71
DEFINE attribute
  setting with a change rule 151
  setting with a script 529
  updating with a script 532
  using in comparisons 246
DEFINE NO
  generation of the -ISMT command 70
  used in PeopleSoft environment 589
DELETE_WORKID script 368, 371
DELETEAGE keyword
  deleting baselines 191
  overriding 193
deleting
  aliases 417
  applications 645
  baselines using DELETEAGE 192
  baselines using RETAINMAX 192
  incomplete baselines 191
  sync records for a work ID 103
  sync table entries 371
  task IDs 615
  work IDs 355, 368
delimited identifiers, authorization switching 97
delimiters 179, 238, 325
detail panels, used by Specification 321
DETAIL parameter of REPORT keyword 258
Device Type, Space Estimation parameter for table space 333
Devt, Space Estimation parameter 337
diagnostic output
  for Analysis 343
  for Baseline 187
  for Baseline Report 189
  for CM/PILOT 611
  for Compare 257
  for Execution 348
  for Import 330
disposition of data sets 109
DML
  adding columns to PLAN_TABLE 415
  browsing statements 615
  changing data structures 410
  changing statements 126
  errors 612
  evaluating changes before execution 611
  importing 374
  overview 41
  trial report 611
DML report 611
DML section 612
DML_MIGRATE script 123
DML_STRUCTURE_CHG script 410
documentation information 24
DROP parameter of INCLUDE keyword 250
Drop Recovery option 90
DROP RESTRICT clause 78, 421
DROPALL command 46, 311
dropping a column from a table 399
DROPRESTRICT attribute 247
DSN1COPY utility, using to perform data-only migrations 561
dssize attribute 51
DSSize, Space Estimation parameter 335
dxxxyyyy DD name 682
dynamic allocation
  copy data sets 683
  data sets 93
  requirement for worklist parallelism 711
SYSCOPY data sets 683
SYSREC data sets 656, 661
unload data sets 656, 661
using the COPY PLUS utility 683
using the IBM COPY utility 683
using the IBM UNLOAD utility 661
using the RECOVER PLUS utility 683
using the UNLOAD PLUS utility 656
DYNCOPY installation option 711
DYNCOPY keyword
  dynamically allocated image copy data sets 680, 683
  using with UTILCOPY and NOUTILCOPY 681, 685
worklist parallelism 711
DYNUMUNLD installation option 711
DYNUMUNLD keyword 656, 661, 711
DYWORKUNIT keyword 94

E

E (Edit) action code 310
EBCDIC collating sequence 43
edit macros
  CHKSQNUM 82
  FIXSQNUM 82
editing
  applications 643
  CM/PILOT worklists 609
  DML statements 605
  task IDs 605
  electronic documentation 23
encoding scheme, changing 493
ENFORCE CONSTRAINTS option 670
ENFORCE NO option 670
ENFORCE option 670
enforcing referential constraints on a LOAD 670
ENV keyword 68
environments, PeopleSoft 587
EOJ (end of job) 103
ERP applications  
challenges 584  
changing multi-table table spaces to single-table table spaces 525  
changing table spaces and indexes to DEFINE NO 133  
CM/PILOT solutions 525  
creating a new database and table spaces for tables 523  
creating copies of environments 587  
creating environments 585  
managing 523  
moving empty tables to one table space 527  
optimizing the design 592  
taking baselines 585  
estimating space  
index 335, 434  
table space 332, 428  
ESTROWS parameter, -BMCD worklist command 658  
evaluating CDL 551  
exception baseline 172  
Exclude option, scope rules 147  
exclude-type change rules 48  
Exec Com (execution completed) 101  
Exec Strt (execution started) 101  
Execution return codes 99  
evaluation, worklist  
additional job steps, to specify 539  
AEXPRINT diagnostic output file 103, 348  
alter-type worklist 537  
authorization switching 94  
Batch Execution JCL Generation 87  
cleanup job set for data sets 94  
creating a step in a script 630  
creating a work ID in batch 155  
data set sizing for 87  
deleting sync records 103  
editing JCL 92  
generating JCL to create execution JCL in batch 87  
initial 85  
inputs 90, 344  
LOAD utility 110  
migrate-type worklist 83  
onoptional job steps 112  
options 344  
outputs 103, 348  
override options 344  
overview 90, 344  
preallocation step 93  
REORG utility 110  
replacing a work ID in batch 155  
restart, considerations for multistep jobs 347  
restart, overview 85  
restarting 85  
specifying a multi-step job 345  
start over 86  
start over, overview 86  
status 100  
sync records, deleting 103  
sync table 102  
unload data sets 99  
utilities 99  
View Execution Status 102  
work ID, creating in batch 155  
work ID, replacing in batch 155  
work ID, status 101  
explicit databases, converting to implicit 492  
explicit table spaces 322  
external files, importing 323  
external hexadecimal format 55  
Exxxyyyy DD name 682  
F  
falling back to a previous data structure  
with current data 569  
with old data 575  
feeding back changes 554  
file tailoring 693  
FILEREF parameter 182  
FIND command 45  
FIXSQNUM macro 82  
Force-type change rules 48  
foreign keys  
child table 66  
primary key propagation 68  
FORMAT BMCLOAD 652  
FORMAT BMCSLOAD 667  
Free Page, Space Estimation parameter 339  
FREEPAGE attribute 51, 248  
Freepage, Space Estimation parameter 334  
full recovery, load requirements 183  
full-recovery baseline  
authorizations for dropped objects 173  
-BASE command 182, 186  
-BMCD command 182  
creating 181, 201  
falling back 569, 575  
generating a worklist 201  
-ISMT command 183  
overview 173  
requesting 201  
-UNLI command 182  
using 181  
Fxxyyyyy DD name 682  
G  
GDG (generation data group), using dynamic allocation 685  
GENERATED ALWAYS, defined for identity column 322  
GENERATED ALWAYS, ROWID column 650  
GENERATED BY DEFAULT, ROWID column 650
generating CDL for changes 548

generating JCL
   for generation data groups 83
   to create Execution JCL in batch 87

generating multiple CDL files 249

generating statistics 688

GLID (global authorization ID)
   override option 63
   overview 97
   -GLID command 95, 180
   grants, migrating 81
   group ID 249

hash verification numbers 79, 342, 609
Help, online 23
hexadecimal string 179
High Speed Structure Change (HSSC)
   creating objects for SHRLEVEL CHANGE 447
   modifying and migrating objects for SHRLEVEL
   REFERENCE 450

HISTORY ALL parameter 688
HISTORY NONE parameter 688
HISTORY ALL keyword 688
HLQ.DBCNTL data set
   ACMDMLD1 418
   ACMDMLD2 370
   ACMDMLD3 372
   ACMDMLI1 526
   ACMDMLI2 528, 595
   ACMDMLM1 134
   ACMDMLM2 139
   ACMDMLT1 451
   ACMDMLT2 447
   ACMDMLU1 530, 589
   ACMDMLU2 523, 594
   ACMDMLU3 423
   ACMDMLU4 531, 589
   ACMDMLU5 445
   ACMDMLU6 416
   ACMDMLU7 425
   ACMDMLU8 396, 397
   ACMDMLU9 479
   ACMDMLU1A 481
   ACMDMLU1B 473
   ACMDMLU2C 453
   ACMDMLU1D 492
   ACMDMLU1E 422
   ACMDMLU1F 493

IBM RUNSTATS utility 331
IBM utilities. See DB2 utilities

IBMCHECK keyword 669
IBMCOPY keyword 678, 680
IBMLOAD keyword 668
IBMREBUILD keyword 676
IBMREORG keyword 672
IBMUNLOAD keyword 661
IDENTITY attribute 247
identity column
   generating new values 323
   preserving existing values 322
identity columns 322
IDENTITYOVERRIDE YES parameter, BMC LOADPLUS 322
image copies
   building 680
   registering 679
   SHRLEVEL 660, 664
implicit databases, converting from explicit 492
implicit table spaces
   auxiliary objects 321
   considerations for changing data structures 322
   ROWID GENERATED BY DEFAULT column 321

Import component, CD tables 323
importing
   ALUPRINT diagnostic output 330
   CDL 324, 374
   creating a step in a script 627
   data structure for a new application 406
   DB2 catalog 328
   DDL 324, 374, 406
   DDL in ALTER 376
   DML 374
   external files 323
   implementing changes 551
   inbound migrate profiles 327
   incremental DDL 325
   new structures 324
   outputs 323
   overview 323
   text validation 329
inactivating XIM initiators 707
inbound migrate profiles
   creating 356
   creating change rules 151
   described 327
   using in change feedback process 555
   using in change migration process 551
INCLUDE keyword 64, 250, 712
include option, scope rules 42
include-type change rules 48
index on expression text validation
   by Analysis 76
   by Compare 253, 329
   change propagation 77
index partitions
   setting primary quantities 444
   setting secondary quantities 444, 445
   updating 419
index-controlled partitioned table spaces
   converting to partition-by-growth table spaces 452
   converting to range-partitioned table spaces 480
   converting to table-controlled partitioned table spaces 475, 478
index-controlled partitioning, migrating to range partitioning 135
indexes
   auxiliary 313
   changing the clustering 392
   changing the key columns 392
   changing the key sequence 392
   changing the padding 392
   changing the uniqueness 392
   changing to DEFINE NO 133
   creating a primary constraint 379
   improving performance 194, 259
   propagating dropped table columns 78
   reorganizing 670
   specified with DEFINE NO 532
   unique 71
   updating 422
   updating to DEFINE NO 529
inline LOB columns
   migrating 139
   updating length 533
installation options
   AUTHSW 95
   BMCFASTL 652, 667
   DASDMAN 331
   DYNCOPY 711
   DYNUNLID 711
   REGALL 679
installation, space requirements 702
internal format 55
-ISMT command 70, 183
ISPF interface
   CHKSQNUM macro 82
   file tailoring 693
   file tailoring for JCL generation 84
   FIXSQNUM macro 82
   REXX macro 81
J
JCL
   additional job steps, creating 610
   creating for CM/PILOT worklist processing 607
   editing Execution JCL 92
   editing for CM/PILOT worklist processing 609
   submitting for CHANGE MANAGER worklist processing 616
submitting for CM/PILOT worklist processing 609
symbolic variable use 607
JCL generation
   batch 87
   disposition 109
   input 83
   ISPF file tailoring 84
   output 83
   receive-side JCL 90
   -JCLP command 187, 667, 672
K
katakana characters 55
KEEP-ALL-PRIMARY parameter, OVERRIDE ALUIN
   keyword 252
key columns, changing for an index 392
key sequence, changing for an index 392
KEYCARD ALUIN keyword 688
KEYCARD parameter 688
keywords
   <BLANK> 54
   <CURRENT> 52
   <DEFLT> 53, 320
   ACM_PARALLEL_WORKLST 72
   ALLSTATSUPD 689
   AMSDELETEI 675
   AUTHSW 95
   AUTHSWGLID 95, 98
   AUTHSWOFF 95
   BASICUNLOAD 649
   BLNAME 181
   BLPROFILE 181
   BLRECOVER 184
   BLRECOVERPOINT 181
   BLWORKID 186, 345
   BMCCHECK 669
   BMCCOPY 678, 680
   BMCFASTLOAD 652, 667
   BMCREBUILD 676
   BMCREORG 671
   BMCSPACE 445
   BMCSUPD 689
   BMCUNLOAD 650
   CATAUDIT 100
   CATRECOVER 90, 100
   CDLCHANGE2 248, 258
   CHANGERULESIN2 223, 242, 252, 260
   CHANGERULESIN2 NULL 242
   CISIZE4K 71
   CMPIN 237
   CMPINST 237, 240
   COMMIT 195
   COMMITFREQ 654
   COMMITSIZE 654
   COPYDDN 679
<table>
<thead>
<tr>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>F</td>
</tr>
<tr>
<td>G</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>I</td>
</tr>
<tr>
<td>J</td>
</tr>
<tr>
<td>K</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>O</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>Q</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>U</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>W</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td><strong>keywords (continued)</strong></td>
</tr>
<tr>
<td>DB2STATSUPD 689</td>
</tr>
<tr>
<td>DEFAULTOFF 252</td>
</tr>
<tr>
<td>DEFERUNIQUEIX 71</td>
</tr>
<tr>
<td>DELETEAGE 191</td>
</tr>
<tr>
<td>DYNCOPY 680, 681, 683, 685, 711</td>
</tr>
<tr>
<td>DYNUNLD 656, 661, 711</td>
</tr>
<tr>
<td>DYNWORKUNIT 94</td>
</tr>
<tr>
<td>ENV 68</td>
</tr>
<tr>
<td>HISTORYALL 688</td>
</tr>
<tr>
<td>IBMCHECK 669</td>
</tr>
<tr>
<td>IBMCOPY 678, 680</td>
</tr>
<tr>
<td>IBMLOAD 668</td>
</tr>
<tr>
<td>IBMRREBUILD 676</td>
</tr>
<tr>
<td>IBMREORG 672</td>
</tr>
<tr>
<td>IBMUNLOAD 661</td>
</tr>
<tr>
<td>INCLUDE 64, 250, 712</td>
</tr>
<tr>
<td>LOBGT32M 655</td>
</tr>
<tr>
<td>LOCATION 237, 240</td>
</tr>
<tr>
<td>LOG 668</td>
</tr>
<tr>
<td>MAXSYSUT 666, 671</td>
</tr>
<tr>
<td>MIGLOCATIONS 249</td>
</tr>
<tr>
<td>MIGSCOMP 61</td>
</tr>
<tr>
<td>MULTITBDS 652</td>
</tr>
<tr>
<td>NOALIAS 244</td>
</tr>
<tr>
<td>NOAMSDELETEI 675</td>
</tr>
<tr>
<td>NOAUTHSW 95</td>
</tr>
<tr>
<td>NOAUXILIARY 244</td>
</tr>
<tr>
<td>NOBASELINE 193</td>
</tr>
<tr>
<td>NOBMCFASTLOAD 652, 667</td>
</tr>
<tr>
<td>NOBUFFERPOOL 247</td>
</tr>
<tr>
<td>NOCHECKCNST 244</td>
</tr>
<tr>
<td>NOCOMPRESS 248</td>
</tr>
<tr>
<td>NOCOPY 678, 680, 681, 685</td>
</tr>
<tr>
<td>NOCOPYFORCE 678, 680, 681, 685</td>
</tr>
<tr>
<td>NODATABASE 243, 244</td>
</tr>
<tr>
<td>NODEFINE 246</td>
</tr>
<tr>
<td>NODROPRESTRICT 247</td>
</tr>
<tr>
<td>NOFOREIGNKEY 244</td>
</tr>
<tr>
<td>NOFREEPAGE 248</td>
</tr>
<tr>
<td>NOHISTORY 688</td>
</tr>
<tr>
<td>NOIDENTITY 247</td>
</tr>
<tr>
<td>NOINDEX 244</td>
</tr>
<tr>
<td>NOLOCKSIZE 246</td>
</tr>
<tr>
<td>NOMTASK 651, 662</td>
</tr>
<tr>
<td>NOORDERBY 70</td>
</tr>
<tr>
<td>NOPARTCOPY 682</td>
</tr>
<tr>
<td>NOPARTITION 247</td>
</tr>
<tr>
<td>NOPATH 247</td>
</tr>
<tr>
<td>NOPCTFREE 248</td>
</tr>
<tr>
<td>NOPRIQTY 248</td>
</tr>
<tr>
<td>NOREBUILD 676</td>
</tr>
<tr>
<td>NOREGENIDENTITY 323</td>
</tr>
<tr>
<td>NOREMARKS 246</td>
</tr>
<tr>
<td>NOSECQTY 248</td>
</tr>
<tr>
<td>NOSEGSIZE 246</td>
</tr>
<tr>
<td>NOSTARTVALUE 247</td>
</tr>
<tr>
<td>NOSTATS 686</td>
</tr>
<tr>
<td>NOSTOGROUP 244</td>
</tr>
<tr>
<td>NOSYNONYM 244</td>
</tr>
<tr>
<td>NOTABLE 244</td>
</tr>
<tr>
<td>NOTABLEACCESS 73</td>
</tr>
<tr>
<td>NOTABLEALL 687</td>
</tr>
<tr>
<td>NOTABLESPACE 244</td>
</tr>
<tr>
<td>NOTRACKMOD 248</td>
</tr>
<tr>
<td>NOTRIGGER 244</td>
</tr>
<tr>
<td>NOUNLOADEMPTY 70, 183</td>
</tr>
<tr>
<td>NOUTILCOPY 681, 685</td>
</tr>
<tr>
<td>NOVIEW 244</td>
</tr>
<tr>
<td>NOVIEWCHECK 247</td>
</tr>
<tr>
<td>NOVVALPROP 70</td>
</tr>
<tr>
<td>OBJECTS 188</td>
</tr>
<tr>
<td>PARALLEL 72, 681, 685</td>
</tr>
<tr>
<td>PARTCOPY 682</td>
</tr>
<tr>
<td>PIC 680</td>
</tr>
<tr>
<td>PKEYPROP 68</td>
</tr>
<tr>
<td>PROTECT 190</td>
</tr>
<tr>
<td>REBUILD 60</td>
</tr>
<tr>
<td>REGENIDENTITY 323</td>
</tr>
<tr>
<td>REGISTER 679</td>
</tr>
<tr>
<td>REORGALT 73</td>
</tr>
<tr>
<td>REORGONLINE 673</td>
</tr>
<tr>
<td>REPLACEWORKID 155, 328</td>
</tr>
<tr>
<td>REPORTTYPE 188</td>
</tr>
<tr>
<td>RESMAXIC 663</td>
</tr>
<tr>
<td>RESTARTPARM 109</td>
</tr>
<tr>
<td>RETAINMAX 191</td>
</tr>
<tr>
<td>SCOPE 233, 234, 235</td>
</tr>
<tr>
<td>SCOPERULE 233, 234, 235</td>
</tr>
<tr>
<td>SCOPETYPE 233, 234, 235</td>
</tr>
<tr>
<td>SENDONLY 90</td>
</tr>
<tr>
<td>SEQI 61, 82</td>
</tr>
<tr>
<td>SHWRULES 252</td>
</tr>
<tr>
<td>SHRLEVELCHG 660, 664</td>
</tr>
<tr>
<td>SHRLEVELREF 660, 664</td>
</tr>
<tr>
<td>SINGLEPHASE 72</td>
</tr>
<tr>
<td>STANDALONESTATS 687</td>
</tr>
<tr>
<td>STARTOVER 112</td>
</tr>
<tr>
<td>STOPCOMMIT 71</td>
</tr>
<tr>
<td>SYNCDELETE 103</td>
</tr>
<tr>
<td>TABLEALL 687</td>
</tr>
<tr>
<td>THRESHOLD 195, 260</td>
</tr>
<tr>
<td>UNLDCOLL 658</td>
</tr>
<tr>
<td>UNLOADCOPY 659</td>
</tr>
<tr>
<td>UNLOADCOPYDS 659</td>
</tr>
<tr>
<td>UTILCOPY 675, 681, 685</td>
</tr>
<tr>
<td>UTILSTATS 688</td>
</tr>
<tr>
<td>VALWARN 69</td>
</tr>
<tr>
<td>VVALPROP 70</td>
</tr>
<tr>
<td>VWORKID 186, 328</td>
</tr>
<tr>
<td>VWORKIDMERGE 328</td>
</tr>
<tr>
<td>XIMPCTSURMN 94</td>
</tr>
<tr>
<td>XMRETRYCOUNT 93</td>
</tr>
</tbody>
</table>
keywords (continued)
XIMRETRYMINS 93
XIMRETRYSECS 93

L

L (Like) action code 310
L change-level indicator 311
length, changing for a column 400
limit key 179
listing
   applications 633, 643
   scripts 633
   steps for a script 635
   task IDs 604
literal strings 179
-LOAD command
   requirements for a full recovery 183
   REUSE parameter 668
load utilities 665
LOAD utility 110, 668
LOADPLUS utility
   converting data 666
   creating partition-level image copies 682
   restarting 110
   using in a parallel worklist 712
   using with LOB DATA MOVER program 653
LOB columns
   adding to a table 496
   cancelling 315
   defined as NOT NULL WITHOUT DEFAULT 655
   defined greater than 32 MB 655
LOB data
   converting 323, 655
   loading with IBM LOAD 668
   unloading with BMC UNLOAD PLUS 652
   unloading with IBM UNLOAD 662
LOB DATA MOVER program
   considerations 655
   requirements 653
LOB SYSREC data set 655, 657
LOBGT32M keyword 655
LOCATION keyword 237, 240
locations
   generating worklists 61
   using in outbound migrate profiles 41
LOCKMAX attribute 53, 246
LOCKSIZE attribute 246
LOG keyword 668
logical partition numbers 319
-LLOB command
   used in a parallel worklist 712
   used with -BMCL command 654

M

M (Migrate) action code 46
macros
   CHKSQNUM 82
   FIXSQNUM 82
   REXX 81
maintaining
   applications 551, 640
   scripts 620
   sync tables 355, 368, 615
   task IDs 601
managing baselines 190
matching
   nonnumeric values 53
   numeric values 51
materialized query table (MQT) text validation
   by Analysis 76
   by Compare 253, 329
Maxpartitions, Space Estimation parameter 335
Maxrows, Space Estimation parameter 334
MAXSYSREC ALUIN keyword 651, 657
MAXSYSUT keyword 666, 671
merging subsystems 559
MIGALL command 46
MIGLOCATIONS keyword 249
-MIGR command 79
migrate-type work ID
   change rules 60
   migrate options 60
   specifying change rules 48
   used by Analysis 60
   used by Execution 90
   used in migrate process 39
migrate-type worklist
   generating 151
   overview 79
   used by Execution 83
   work ID 91
migrating
   databases 129
   single-table table space to a partition-by-growth table
      space 137
   tables in explicit databases to implicit databases 138
migrating inline LOB columns 139
migration
   application changes to other subsystems 558
   authorizations 81
   auxiliary objects 43, 46, 59, 121
   change rules 47
   changes to data structures 546
   data only 159, 160
   data structures 59
   data structures and data 159, 160
   data structures only 158
   establishing a baseline 547
migration (continued)

- generating a migrate-type worklist using change rules 154
- generating a worklist using a migrate profile with CHANGE MANAGER 154
- generating CDL 548
- grants 81
- multiple locations 41
- outbound migrate profiles 40, 144
- overview 35
- performing a change migration process 558
- process 35
- receiving DDL or a worklist 550
- receiving subsystem 549
- renaming tables 550
- sending subsystem 547
- stored procedures 47
- tables with ROWID columns 650
- to different subsystems 36
- using an inbound migrate profile 551
- using an outbound migrate profile 553
- using DML 123
- workflow 36

MIGSCOPE keyword 61
Mixed List action codes 310
MO (Migrate Options) action code 46
modifying
- applications 643
- DML statements 126, 413, 605
- MVS image variables 708
- task IDs 605
- worklists 81
- worklists and input streams 103
- MQT (materialized query table), creating 426
- multiple SYSUT data sets 666, 671
- multiple tables, moving 401
- multitasking, unloading and loading of data 651, 662
- MULTITBDS keyword 652
- MVS image variables 708

N

name prefixing 314
name template
- generating baseline names 548
- using in baseline profile 345
- using in multistep job 174
name conventions, DD names 682
NbrRows, Space Estimation parameter 333, 336
NBRSECED UNLOAD PLUS installation option 657
NEWWORKKID ALUIN keyword 116, 353
NOALIAS keyword 244
NOAMSDELETEI keyword 675
NOAUTHSW keyword 95
NOALUXILIARY keyword 244
NOBASELINE keyword 193
NoBaseline option 191
NOBMCFASTLOAD keyword 652, 667
NOBUFFERPOOL keyword 247
NOCHECKCNST keyword 244
NOCOMPRESS keyword 248
NOCOPY keyword 678, 680, 681, 685
NOCOPYFORCE keyword 678, 680, 681, 685
NODATABASE keyword 243, 244
NODEFINE keyword 246
NODROPRESSRICT keyword 247
NOFOREIGNKEY keyword 244
NOFREEPAGE keyword 248
NOHISTORY keyword 688
NOIDENTITY keyword 247
NOINDEX keyword 244
NOKEYCARD ALUIN keyword 688
NOLOCKSIZER keyword 246
NOMTASK keyword 651, 662
NonLf KeyLn, Space Estimation parameter 339
nonpartitioned indexes, PIECESIZE attribute 240
nonpartitioned table spaces
- converting to partition-by-growth table spaces 452, 465
- converting to range-partitioned table spaces 464
- converting to table-controlled partitioned table spaces 460
- creating auxiliary objects 513
nonprintable characters 179, 237, 459
nonstandard characters 55
nonviewable characters 55, 179, 237, 459
NOORDERBY keyword 70
NOPARTCOPY keyword 682
NOPARTITION keyword 247
NOPATH keyword 247
NOPCTFREE keyword 248
NORFQTY keyword 248
NOREBUILD keyword 676
NOREGENIDENTITY ALUIN keyword 322
NOREGENIDENTITY keyword 323
NOREMARKS keyword 246
NOREORGALT ALUIN keyword 73, 675
NOSEQCITY keyword 248
NOSEGSIZE keyword 246
NOSTARTVALUE keyword 247
NOSTATS keyword 686
NOSTOGROUP keyword 244
NOSYNONYM keyword 244
NOTABLE keyword 244
NOTABLEACCESS keyword 73
NOTABLEALL keyword 687
NOTABLESPACE keyword 244
NOTRACKMOD keyword 248
NOTRIGGER keyword 244
NOUNIQUECNST keyword 244
NOUNLOADEMPTY keyword 70, 183
NOUTILCOPY keyword 681, 685
NOVIEW keyword 244
NOVIEWCHECK keyword 247
NOVVALPROP keyword 70
null characters 55, 179, 237, 459
number formats 174, 603
NUMPARTS attribute 247

objects
  baseline, browsing 185
  change flag keywords 244
  comparing 219
  displaying in Compare 240
  synchronizing 219
OBJECTS keyword 188
online Help 23
online reorg 672
online schema changes 672
options
  overriding in Analysis 62
  overriding in Compare 243
  overriding in Execution 89
  overriding in JCL Generation 91
ORDER BY clause 70
outbound migrate profiles
  creating change rules 151
  defining scope 42
  group ID 249
  locations 40
  name resolution in a catalog-to-catalog comparison 242
  scope rules 146
  specifying change rules 49
  used as scope 234
  used by Compare 253
  using change rules 40, 248
  using change rules for multiple production systems 50
  using locations 41, 248
  using scope to select objects 40
OUTPUT descriptors
  naming conventions 684
  using in the COPY PLUS utility 684
  using in the RECOVER PLUS utility 684
  using in the UNLOAD PLUS utility 657
override options
  Analysis, defaults 62, 341
  Execution 89
  JCL Generation 91

parameters
  Allocation Unit 334
  ALTER 250
  AU (Allocation Unit) 339
  Avg Len 337
  Avg Row Length 333
  CDLDEP 251
  Compress 334, 338
  CREATE 250
  DEFER YES 71
  DEFINE NO 70
  DETAIL 258
  Device Type 333
  Devt 337
  DROP 250
  Dssize 335
  Free Page 339
  Freepage 334
  Maxpartitions 335
  Maxrows 334
  NbrRows 333, 336
  NonLf KeyLn 339
  PARM 94
  Pct Compressed 337
  Pct Free 339
  Pctfree 334
  Percent Compressed 333
  Piecesize 338
  PriQty 338
  Priqty 334
  Rows/Key 336
  SecQty 338
  Secqty 334
  Segsize 334
  Unique 338
  UTILID 111
  WORKDDN 667, 672
PARM parameter 94
PART symbolic variable 683
PARTCOPY keyword 682
partition-by-growth table spaces
  converting from index-controlled partitioned table spaces 452
  converting from nonpartitioned table spaces 452
  converting from segmented table spaces 452
  converting from table-controlled partitioned table spaces 452
  converting to index-controlled partitioned table spaces 485
  converting to range-partitioned table spaces 488
  converting to table-controlled partitioned table spaces 487
partitioned table spaces
  converting to nonpartitioned table spaces 467
  converting to partition-by-growth table spaces 482
  creating auxiliary objects 500, 506

padding, changing for an index 392
PARALLEL ALUIN keyword 71, 72
PARALLEL keyword 681, 685
partition-level image copies, creating 682
partitions
  change flag 247
  converting 424
  VCAT defined 65, 674
PATH attribute 247
Pct Compressed, Space Estimation parameter 337
Pct Free, Space Estimation parameter 339
PCTFREE attribute 51, 248
Pctfree, Space Estimation parameter 334
PCTPRIM UNLOAD PLUS installation option 657
PE (propagate estimates) action code 334, 338
PE (propagate estimates) command 338
PEALL (propagate attributes) command 334
PeopleSoft
  challenges 584
  creating copies of environments 587
  creating environments 585
  managing 583
  modifying application tables 584
  modifying PeopleTools tables 584
  optimizing the design 592
  taking baselines 585
  using DEFINE NO 589
PeopleTools tables, modifying 584
Percent Compressed, Space Estimation parameter 333
performance
  catalog-to-catalog comparison 245
  creating indexes for baselines 194
  creating indexes for comparisons 259
  improving in a baseline 194
  improving in a comparison 259
periodic baseline 172
permanent data sets, space required 702
physical partition numbers 319
PIC keyword 680
PIECESIZE attribute 53, 240
Piecesize, Space Estimation parameter 338
PKEYPROP keyword 68
PLAN_TABLE, adding columns 415
post-Execution baseline 87, 344, 610
post-Execution comparison 87, 344, 610
PREALLOC DD 93
pre-Execution baseline 87, 344, 610
pre-Execution comparison 87, 344, 610
preserving limit keys 318
primary constraints
  creating 379
  including in scope 252
primary copy, DD name 682
primary keys
  editing 387
  naming 390
  propagation 68
primary quantity, setting 130, 444
PRIQTY attribute 51, 248
PriQty, Space Estimation parameter 338
processes
  change 306
    change migration 546
  data structure migration 171
  migration 35
    migration within same DB2 subsystem 36
product support 3
profiles
  application, specifying in 635, 642
  catalog baseline 173
  catalog to catalog comparison 242, 248
  change rules 50, 242, 248, 327
  DDL baseline 173
  inbound migrate 327
  locations 41
  outbound migrate 40
  task ID, specifying in 608
propagating
  dropped table columns to indexes 78
  name changes 58
PROTECT keyword 190
Q
QE (quick edit) action code
  Mixed List panel 46
  moving multiple tables 401
QUIESCE command 707
R
RACF (IBM Resource Access Control Facility), authorizing
  procedures for XIM 704
range-partitioned table spaces
  converting from index-controlled partitioned table spaces 480
  converting from table-controlled partitioned table spaces 472
  converting to index-controlled partitioned table spaces 474
RCV_STRUCCUR_DDATA script 570
RCV_STRUCOLD_DDATA script 576
rebind option 66
REBUILD INDEX utility 676
REBUILD keyword 60
rebuild utilities 676
rebuilding a worklist 60
RECV_CHGSRBL script 564
RECV_CHGSS script 561
RECVDDL script 406
receive-type work ID
  creating 155
  creating in batch 155
receive-type work ID (continued)  
- re-creating an existing 155  
- used by Execution 91  
receiving data structure changes 561, 564  
- receiving DDL 550  
- receiving node 90  
- receiving subsystem 549  
- receiving worklists 550  
- RECLnnnn DD name 682  
RECOVER PLUS utility  
- creating partition-level image copies 682  
- rebuilding indexes 676  
- restarting 110  
- unload options 663  
- using 663  
rerecovering data structures  
- establishing a baseline 553  
- with current data 576  
- with old data 576  
recovery baseline with data, creating 201  
recovery point  
- overview 169  
- structure and data 184  
- structure-only 172  
- RECRnmmm DD name 682  
- REDEFINE NO parameter 666  
- referential constraint violations, checking 670  
- REGALL installation option 679  
- REGENIDENTITY keyword 323  
- REGISTER keyword 679  
rename table  
- used by Compare 254, 550  
- used in change migration process 550  
REORG PLUS utility  
- creating partition-level image copies 682  
- restarting 110  
- specifying 671  
- reorg utilities 675  
- REORG utility 110, 672  
- REORGALL ALUIN keyword 73, 670, 675  
- REORGALT ALUIN keyword 675  
- REORGALT keyword 73  
- REORGNONE ALUIN keyword 673  
- REORGONLINE keyword 673  
- REORG-pending (AREO*) advisory status 670  
- REORG-pending (REORP) restrictive status 670  
- REORGPENDONLY ALUIN keyword 73, 670, 675  
- REORGRREF ALUIN keyword 673  
- REPAIR utility 71  
- REPLACEWORKID keyword 155, 328  
- replacing a work ID in batch 155  
- replacing an existing receive-type work ID 155  
- REPCLATE_WORKID script  
- changing column definitions for imported DDL 365  
- modifying a work ID with values in a user-defined table 361  
- replicating multiple work IDs 357  
REPORT keyword 258  
- reports  
- generating a baseline report 187, 209  
- generating a comparison report 258  
REPORTTYPE keyword 188  
- requirements, using worklist parallelism 711  
REMAXIC keyword 663  
- restarting  
- CM/PILOT worklist 607, 616  
- LINEONEPARMS 110  
- multistep jobs 347  
- processing 108  
- -SSID command 111  
- UTILID 111  
- utility with the RESTARTPARM keyword 111  
- utility without the RESTARTPARM keyword 109  
- worklist 107  
RESTARTPARM keyword  
- including 111  
- omitting 109  
- restoring previous versions of data structures 569, 575  
REMAXIC keyword 663  
- deleting baselines 191  
- overriding 193  
return codes, Execution 99  
- REUSE parameter 668  
- reusing a task ID 611  
- reviewing and executing data structure changes 344  
- -REXC command 81  
- REXX executable 81  
- REXX macro 81  
- -RNST command  
- processing in parallel 687  
- used in a parallel worklist 712  
- rotating partitions 319  
- ROWID columns  
- adding to a table 495, 519  
- creating a unique index 497  
- defined as GENERATED ALWAYS 650  
- defined as GENERATED BY DEFAULT 650  
- migrating 650  
- ROWID GENERATED BY DEFAULT column 321  
- ROWID SYSREC data set 655, 657  
- Rows/Key, Space Estimation parameter 336  
rules  
- change 327  
- scope 176  
RUNSTATS utility  
- using 687  
- using in a parallel worklist 712
SBCS (single byte character set) 55
scope
  catalog-to-baseline comparison 234
  catalog-to-DDL comparison 234
  catalog-to-worklist comparison 234
  DDL 235
defining 42, 232
migrate profile 235
MIGSCOPE keyword 61
specifying a table owner 240
specifying information included in a comparison report 258
specifying locations 240
using in outbound migrate profile 40
SCOPE keyword 233, 234, 235
scope rules
  auxiliary object 43, 235
catalog baseline profile 177, 197
Exclude option 147
Include option 42
outbound migrate profile 146
overview 176
used in a baseline 176
used in a comparison 234
scope type
  baseline profile 234
  DDL 235
  migrate profile 235
  worklist 235
SCOPERULE keyword 233, 240
SCOPETYPE keyword 233, 234, 235, 236
SCPYLOCB descriptor 684
SCPYLOC descriptor 684
SCPYREMB descriptor 684
SCPYREMP descriptor 684
script
  baselines, specifying 608
  browsing 636
changing data structures using a DML trigger 410
changing data structures using an existing work ID 403
changing step settings 608
CMP_STRUCTURE_CHG 294
comments, adding 622
creating a Baseline report step 631
creating a Baseline step 624
creating a Compare step 265
creating a full-recovery baseline 205
creating a work ID step 632
creating an Analysis step 629
creating an Execution step 630
creating an Import step 627
data sets for CHANGE MANAGER 608
DELETE_WORKID 368, 371
deleting 637
deleting sync table entries 371
deleting work IDs 368
editing 634
existing CHANGE MANAGER objects required 620
listing 633
listing steps for a script 635
listing task IDs for a script 639
maintaining 620
overview 620
profiles 608
RCV_STRUCT_CUR_DATA 570
RCV_STRUCT_OLD_DATA 576
REC_CHG_SR.BL 564
REC_CHG_SS 561
REC_DDL 406
receiving data structure changes (strong receiver) 564
receiving data structures (strong sender) 561
receiving DDL to create data structure 406
recovering data structure with current data 570
recovering data structure with old data 576
REPLICATE_WORKID 357, 361, 365
scope for a comparison, specifying 608
selecting 620
selecting a script 620
short name 620
specifying information for baselines 608
specifying scope for a comparison 608
steps, changing settings 608
WKID_STRUCTURE_CHG 403
work ID types created 602
secondary quantity, setting 130, 132, 444, 445
SECQTY attribute 51, 248
SecQty, Space Estimation parameter 338
Secqty, Space Estimation parameter 334
security
  specifying for a task ID 605
  specifying for an application 635
  using XIM 704
segmented table spaces, converting to partition-by-growth table spaces 452
SEGSIZE attribute 51, 246
Segsize, Space Estimation parameter 334
sending node 90
sending subsystem 547
SENDONLY keyword 90
SEQI keyword 61, 82
-SETP command 102
-SETS command 95, 102, 180
setting a secondary quantity in a table partition 445
share level on image copies 660, 664
SHOWRULES keyword 252
SHRLEVEL CHANGE 660, 664
SHRLEVEL CHANGE HSSC process, creating objects 447
SHRLEVEL CHANGE online reorg 673
SHRLEVEL NONE option 74
SHRLEVEL NONE reorg 673
SHRLEVEL on image copies 660, 664
SHRLEVEL REFERENCE 660, 664
SHRLEVEL REFERENCE HSSC process, modifying and migrating objects 450
SHRLEVELCHG keyword 660, 664
SHRLEVELREF keyword 660, 664
SHUTDOWN command 707
shutting down XIM 707
single byte character set 55
SINGLEPHASE keyword 72
skeleton library compiler. See SLIB (skeleton library) compiler
SLIB (skeleton library) compiler
changing an ISPF skeleton 693
compiling 692, 694
ISPF file tailoring 693
processing 695
runtime report 695
runtime report summary 697
runtime unit 695
testing changes 693
SMS (Storage Management Subsystem) 65, 702
sort processing log 113
sorting work IDs 354
Space Estimation
BMCSPACE keyword 445
calculation of levels 335
calculation of pages 332
compression percentage 332
estimating primary quantities 444
estimating secondary quantities 444
estimating space requirements 428
index 335
overview 331
setting parameters 441
table space 332
Specification
Auxiliary Objects List panel 312
blank values 54
CD Tables, using 43
copied fields 322
data structure changes 340
default values 53
detail panels 321
Mixed List panel 43
new values based on a current values 52
specifying a blank value for an attribute 54
specifying a current value for an attribute 52
-SQL command 64, 102, 180
SQL terminator 179, 238, 325
-SSID command 111, 180
SSID, specifying in EXEC statement 94
STANDALONESTATS keyword 687
START attribute 247
starting a worklist over. See restarting
starting XIM 706
STARTOVER keyword 112
statistics
generating 688
override options 686
utilities 686
statistics, cardinality 688
STATUS command 705
status of XIM, determining 705
status, work ID 101
STOGROUP attribute 247
-STOP command 103
STOPCOMMIT keyword 71
storage group
specifying default (<DEFLT>) 53
used by Analysis 75
stored procedures, migrating 47
strong receiver
overview 551
receiving data structure changes 564
strong sender
overview 552
receiving data structure changes 561
structure recovery points 178
structure-only baseline 178
submitting JCL
for CHANGE MANAGER worklist processing 616
for CM/PILOT worklist processing 609
subsystem, merging 559
support, customer 3
suppress-type change rules 48
SYCLnnnn DD name 682
SYCRnnnn DD name 682
symbolic variables, PART 683
-SYNC command 103
sync point frequency, override option 63
sync records
conditions for writing 102
deleting for a work ID 103
effect 103
overview 85
sync table
columns 102
deleting records 103
entries, deleting 371
maintaining 355, 368, 615
overview 103
used by Execution 102, 108
viewing task ID entries 613
SYNCAUTH column 102
SYNCCMD column 102
SYNCGLID column 102
SYNCSEQ column 102
SYNCDATA column 102
SYNCSETP column 102
SYNCSETS column 102
synonyms
  applying change rules 50
  creating 423
syntax format for diagrams 27
syntax statement conventions 26
SYSCOPY data set, dynamically allocating 683
SYSnnnnn data set 107, 113
SYSREC data sets
dynamically allocating 656, 661
  LOB 655, 657
  ROWID 655, 657
SYSUDUMP data set 702
SYSUT data set 666, 671
SYSUT work data sets 668

T

table space partitions
  adding 454
  setting primary quantities 444
  setting secondary quantities 132, 444, 445
table spaces
  adding partitions 454
  auxiliary 313
  changing multi-table table spaces 525
  changing the encoding scheme 493
  changing to DEFINE NO 133
  changing VCAT-defined 674
  creating 523
  creating auxiliary objects 500, 506
  creating implicitly 320
  explicit 322
  implicit 321
  reorganizing 670
  specified with DEFINE NO 532
  specifying <DEFLT> as the name 320
  updating to DEFINE NO 529
  with existing data set 422
table spaces, converting 315
  index-controlled partitioned to partition-by-growth 452
  index-controlled partitioned to range-partitioned 480
  index-controlled partitioned to table-controlled partitioned 475, 478
  nonpartitioned to partition-by-growth 452, 465
  nonpartitioned to partitioned 454
  nonpartitioned to range-partitioned 464
  nonpartitioned to table-controlled partitioned 460
  partition-by-growth to indexed-controlled partitioned 485
  partition-by-growth to range-partitioned 488
  partition-by-growth to table-controlled partitioned 487
  partitioned to nonpartitioned 467
  partitioned to partition-by-growth 482
  partitioned to range-partitioned 469
  range-partitioned to index-controlled partitioned 474
  segmented to partition-by-growth 452
  single-table to partition-by-growth 489
  table-controlled partitioned to index-controlled partitioned 471
  table-controlled partitioned to partition-by-growth 452
  table-controlled partitioned to range-partitioned 472
tables
  adding a column 397
  adding a LOB column 496
  adding a ROWID column 495, 519
  adding WITH RESTRICT ON DROP 421
  auxiliary 313
  baseline 184
  creating a base table 494
  creating a primary constraint 379
  creating a unique constraint 381
  dropping a column 399
  moving 527
  moving multiple 401
  propagating dropped columns to indexes 78
  space estimation 422, 523, 532
tapes
data sets, LOADPLUS utility 668
data sets, used in restart processing 109
  stacking 651, 662
task IDs
  browsing 614
  browsing DML statements 615
  changing work ID when executing 606
  CM/PILOT worklist failure 606, 607, 616, 618
  CM/PILOT worklist processing options 607
  comments, adding 604
  comments, changing 605
  creating 602
  deleting 615
  editing 605
  editing CM/PILOT worklists 609
  executing 605, 609
  execution status, viewing 613
  listing 604
task IDs (continued)
   listing for a script 639
   maintaining 601
   modifying 605
   options, processing 607, 609
   overriding CHANGE MANAGER options 607
   overview 601
   profiles, specifying 608
   restart processing for a CM/PILOT worklist 616
   restarting CM/PILOT worklist execution 607
   reusing 611
   starting CM/PILOT worklist execution over 607, 618
   status for restart processing of CM/PILOT worklist 617
   sync table entries, viewing 610, 613
   types of work IDs required 602
   viewing information 604, 614
   viewing sync table entries 610, 613
   viewing the execution status of a task ID 613
   technical support 3
   TEMPLATE descriptors, naming conventions 684
   templates 313
   terminators 179, 238, 325
   THRESHOLD keyword 195, 260
   -TIME command 180
   TRACKMOD attribute 248
   trigger text
      applying change rules to 329
      change propagation 77
      change rules 329
   trigger text validation
      by Analysis 76
      by Compare 253, 329
   troubleshooting, execution of XIM 710

U

U (Undo) action code 46, 311
UNDOALL command 46, 311
Unicode objects
   loading with the LOAD utility 668
   loading with the LOADPLUS utility 666, 667
   unloading with the BASIC UNLOAD utility 649
   unloading with the UNLOAD PLUS utility 652
unique constraints
   creating 381
   editing 385
   excluding from scope 252
unique index, creating for a ROWID column 497
Unique, Space Estimation parameter 338
uniqueness, changing for an index 392
-UNLB command
   used in a parallel worklist 712
   used with -BMCD command 653
UNLDCOLL keyword 658
-UNLI command
   deleting data sets 661
   unloading LOB data 662
   unloading XML data 662
   using in a full-recovery baseline 182
   using multitasking 662
unload data set
   -BMCD command 186
   dynamically allocating 656, 661
   -JCLP command 187
   shown in baseline report 189
   -UNRC command 186, 187
   used by Baseline 182, 187
   used by Execution 99
UNLOAD PLUS installation options
   NBRSECD 657
   PCTPRIM 657
UNLOAD PLUS utility
   converting data 666
   generating single worklist command 652
   restarting 110
   unloading data into separate data sets 652
   used in a parallel worklist 712
   with using LOB DATA MOVER program 653
unload utilities 648
UNLOAD utility 660
UNLOADCOPY keyword 659
UNLOADCOPYDS keyword 659
unloading data 662
unloading data into separate data sets 652
-UNRC command
   DEFINENO parameter 183
   EMPTY parameter 183
   FILEREF parameter 182
   used by Baseline 187
   used by Execution 186
UOWTRnnn data set 106
UPDATEMAXA YES parameter, BMC LOADPLUS 322
updating length of inline LOB columns 533
USE (Use) action code 46
user-defined table, applying changes 361
UTILCOPY keyword 675, 681, 685
UTILID parameter 111
utilities
   BASIC UNLOAD 649
   BMCSTATS 110
   BMCTRIG 110
   CHECK DATA 669
   CHECK PLUS 110, 669
   COPY 678
   COPY PLUS 110
   DSN1COPY 561
   LOAD 668
   LOADPLUS 110
   REBUILD INDEX 676
   RECOVER PLUS 110, 663, 676
   REORG 110, 672
utilities (continued)
   REORG PLUS 110
   REPAIR 71
   restarting 109
   RUNSTATS 687
   starting over 112
   UNLOAD 660
   UNLOAD PLUS 110
   used by Analysis 648
   UTILSTATS keyword 688

V
values
   blank, specifying 54
   default, specifying 53
   hexadecimal 55
   new, specifying based on current values 52
   nonnumeric, matching 53
   nonprintable characters 55
   nonstandard characters 55
   nonviewable characters 55
   numeric, matching 51
   specifying a range 51
   specifying blank 54
   specifying default 53
   specifying nonnumeric 53
   specifying special numeric 53
   VALWARN keyword 69
variables
   XCF_GROUP 709
   XIM_GROUP 709
   VCAT-defined table space, reorganizing 674
   version, establishing 547
   VIEW command 610
view text
   applying change rules to 329
   change propagation 77
   change rules 329
view text validation
   by Analysis 76
   by Compare 253, 329
view, creating an MQT from 426
viewing
   change rules 329
   execution status of a task ID 613
   information about a task ID 604
   sync table entries for a task ID 613
VSAM data sets, creating 71
VVALPROP keyword 69

W
-WKID command 98
WKID_MIGRATE script 141
WKID_STRUCTURE_CHG script 403
WKNAME column 102
WKOWNER column 102
work IDs
   alter-type 352, 549
   automatically creating a new work ID in batch 91
   baseline 186
   CDL conversion 258, 291, 324
   change rules 47, 60
   changing data structure using an existing 403
   changing when executing a task ID 606
   comparing contents 324
   converting to CDL 291
   creating 352
   creating a step in a script 632
   creating in batch 155
   creating in batch mode 116, 353
   deleting 355, 368
   deleting multiple work IDs 368
   deleting sync records 103
   deleting sync table entries 371
   execution 155
   generating CDL from 291
   migrate options 60
   migrate worklist 91
   migrate-type 39, 90
   migrating data structure using an existing 141
   modifying 361
   name template 345, 603, 618, 619, 640, 642
   receive-type 155
   replacing in batch 155
   replicating 632
   replicating multiple work IDs 357
   restart processing 617
   sorting 354
   space estimation, setting 422, 523, 532
   specifying for a task ID 606
   status 101
   status for restart processing of CM/PILOT worklist 617
   status update 101
   types for scripts 602
   types required for task IDs 602
   used by Baseline 186
   used by Execution 91, 101
   using in change migration process 549
   using to change data structures 403
WORKDDN parameter 667, 672
workflow
   alter process 307
   baseline process 169
   comparison process 217
   migrate process 36
WORKID keyword 98, 186, 328
WORKIDMERGE keyword 328
working with applications 640
worklist parallelism
  AEXPRINT data set 701
  allocating space 702
  available in Database Administration 699
  cleanup job set for data sets 94
  creating permanent work data sets 93
  preallocation step 93
  processing utility commands 712
  requirements 711
  restarting 113
  stacked tapes 701
  starting over 113
  using in a PeopleSoft environment 699
worklists
  AEXPRINT diagnostic output 611
  alter-type 342
  analysis 79, 342
  baseline 180
  catalog baseline 180
  CM/PILOT 616
  compare 235
  controlling contents 64
  DDL baseline 180
  editing 609
  executing 537
  generating multiple 40
  generation, alter-type worklist 535
  generation, multiple locations 61
  hash verification numbers 79, 342
  including command types 64
  migrate-type 79, 83
  modifying 81
  multiple 112
  parallel processing 700
  processed by Analysis 340
  rebuilding 60
  restarting 107
  sequence number increment 61, 82
  sequencing 82
  starting over 112
  starting worklist execution over 618
  tailoring with change rule 50
  used by Analysis 342
  used by Baseline 180
  used by Execution 83, 98
  worklist-to-DDL file comparison 224, 274
  worklist-to-worklist comparison 224, 274
  worklog 103, 348
  WRK100n data set 674

X
  X’00’ 179
  XCF_GROUP variable 709

XIM (Cross-System Image Manager)
  controlling execution 704
  determining status 705
  modifying active initiators 708
  shutting down 707
  starting 706
  troubleshooting 710
XIM commands
  ACTIVATE 708
  QUIESCE 707
  SHUTDOWN 707
  STATUS 705
XIM initiators
  activating 708
  inactivating 707
XIM_GROUP variable 709
XIMACM started task 706
XIMCSTSURMN AEXPIN keyword 94
XIMRETRYCOUNT AEXPIN keyword 93
XIMRETRYMIN AEXPIN keyword 93
XIMRETRYSECS AEXPIN keyword 93
XM (Auxiliary Object XML) action code 311
XML data
  loading with IBM LOAD 668
  unloading with BMC UNLOAD PLUS 652
  unloading with IBM UNLOAD 662

Z
  ZOOM command 45