IBM DB2 Path Checker for z/OS
Version 4  Release 2

User's Guide

IBM
Note:
Before using this information and the product it supports, read the information in ‘Notices’ on page 183.
# Contents

## Figures ................................... vii

## Tables .................................... ix

## About this information ............................. xi

- Service updates and support information ........................................... xi
- How to read syntax diagrams ......................................................... xi
- How to send your comments ........................................................ xiii

## Chapter 1. DB2 Path Checker overview .................................................. 1

- What’s new in DB2 Path Checker ..................................................... 1
- Terminology ................................................................................. 2
- What does DB2 Path Checker do? ................................................... 3
- DB2 Path Checker features and benefits .......................................... 6
- Data requirements .......................................................................... 10
- Architecture and process flow ....................................................... 13
- DB2 Path Checker scenarios ........................................................... 14
  - Scenario: Test a package against the previous BIND ..................... 14
  - Scenario: Migrate to a new DB2 version ..................................... 15
  - Scenario: Deploy a new version of an application ....................... 16
- Avoid unnecessary binds by using DB2 Bind Manager .................... 17
- Retrieve hints, recommendations, and expert advice from SQL Performance Analyzer ............................................................. 17
- Service updates and support information ......................................... 18
- DB2 Path Checker documentation and updates .............................. 18
- Accessibility features .................................................................... 19

## Chapter 2. Customizing DB2 Path Checker by using Tools Customizer .......... 21

- Preparing to customize DB2 Path Checker ........................................ 21
  - Set up your environment prior to customization ............................ 23
  - Worksheets: Gathering required data set names ......................... 25
  - Worksheets: Gathering parameter values for Tools Customizer ......... 26
- Starting and preparing Tools Customizer for use .............................. 35
  - Starting Tools Customizer ......................................................... 35
- Modifying Tools Customizer user settings ....................................... 36
- Hiding and displaying panel text .................................................... 39
- Customizing DB2 Path Checker ...................................................... 39
  - Roadmap: Customizing DB2 Path Checker for the first time .......... 40
  - Roadmap: Customizing a new version of DB2 Path Checker from a previous customization ............................... 41
  - Roadmap: Recustomizing DB2 Path Checker ................................. 42
- Specifying the metadata library for the product to customize ............... 43
- Discovering DB2 Path Checker information automatically ................ 45
- Creating and associating DB2 entries ............................................. 47
- Defining parameters ....................................................................... 49
  - Defining DB2 Path Checker parameters .................................... 49
  - Defining DB2 parameters ......................................................... 51
- Generating customization jobs ....................................................... 54
- Submitting customization jobs ....................................................... 55
- Browsing parameters ..................................................................... 57
- Copying DB2 entries ....................................................................... 58
- Removing DB2 entries ................................................................... 59
- Deleting DB2 entries ..................................................................... 60
- Displaying customization jobs ....................................................... 60
- Maintaining customization jobs ..................................................... 61
- Using Tools Customizer in a multiple-LPAR environment .................. 61

© Copyright IBM Corp. 2000, 2014; Copyright HLS Technologies, 2000, 2014
<table>
<thead>
<tr>
<th>Chapter 3. Plan tables</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualifying tables for SQL Explain processing</td>
<td>64</td>
</tr>
<tr>
<td>Creating optimization hints for plan tables</td>
<td>65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4. Running DB2 Path Checker</th>
<th>67</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running DB2 Path Checker by submitting batch jobs</td>
<td>67</td>
</tr>
<tr>
<td>Selecting programs for processing by using name substitution</td>
<td>68</td>
</tr>
<tr>
<td>Generating information about potential changes to access paths for plans and packages</td>
<td>69</td>
</tr>
<tr>
<td>Generating access path information for a DBRM</td>
<td>70</td>
</tr>
<tr>
<td>Generating access path information for a package</td>
<td>71</td>
</tr>
<tr>
<td>Evaluating potential performance degradation of bind or rebind operations</td>
<td>73</td>
</tr>
<tr>
<td>Determining potential access path changes since the previous bind</td>
<td>73</td>
</tr>
<tr>
<td>Determining access path changes for different versions of the same program</td>
<td>75</td>
</tr>
<tr>
<td>Determining access path changes by comparing the results of bind or EXPLAIN operations</td>
<td>78</td>
</tr>
<tr>
<td>Running DB2 Path Checker from the ISPF interface</td>
<td>81</td>
</tr>
<tr>
<td>Setting global options</td>
<td>82</td>
</tr>
<tr>
<td>Reporting on plan table information</td>
<td>83</td>
</tr>
<tr>
<td>Comparing access paths for packages</td>
<td>84</td>
</tr>
<tr>
<td>Comparing access paths for plans</td>
<td>86</td>
</tr>
<tr>
<td>Testing access path data for a DBRM</td>
<td>88</td>
</tr>
<tr>
<td>Generating batch jobs</td>
<td>90</td>
</tr>
<tr>
<td>Step 1. Generating DB2 Path Checker commands</td>
<td>90</td>
</tr>
<tr>
<td>Step 2. Generating and submitting batch jobs</td>
<td>96</td>
</tr>
<tr>
<td>Using utility functions to manage and maintain plan tables</td>
<td>97</td>
</tr>
<tr>
<td>Creating plan tables</td>
<td>97</td>
</tr>
<tr>
<td>Resetting plan tables</td>
<td>98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5. Reporting on access paths</th>
<th>101</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to interpret reports</td>
<td>103</td>
</tr>
<tr>
<td>RPTPKG long report</td>
<td>105</td>
</tr>
<tr>
<td>SYSPRINT report</td>
<td>106</td>
</tr>
<tr>
<td>SYSEXPLN report</td>
<td>107</td>
</tr>
<tr>
<td>SYSCFG report</td>
<td>108</td>
</tr>
<tr>
<td>SYSPRINT report with access path structure change</td>
<td>108</td>
</tr>
<tr>
<td>History table</td>
<td>108</td>
</tr>
<tr>
<td>Compare access paths for two existing binds</td>
<td>110</td>
</tr>
<tr>
<td>REPORT example with MAKE command</td>
<td>112</td>
</tr>
<tr>
<td>TEST example</td>
<td>114</td>
</tr>
<tr>
<td>Command language examples</td>
<td>116</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6. Troubleshooting</th>
<th>119</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery procedures</td>
<td>119</td>
</tr>
<tr>
<td>Messages and codes</td>
<td>119</td>
</tr>
<tr>
<td>Return codes</td>
<td>119</td>
</tr>
<tr>
<td>Messages</td>
<td>119</td>
</tr>
<tr>
<td>How to look up message explanations</td>
<td>135</td>
</tr>
<tr>
<td>Gathering diagnostic information</td>
<td>135</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 7. Reference</th>
<th>139</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools Customizer terminology and data sets</td>
<td>139</td>
</tr>
<tr>
<td>Tools Customizer terminology</td>
<td>139</td>
</tr>
<tr>
<td>Data sets that Tools Customizer uses during customization</td>
<td>139</td>
</tr>
<tr>
<td>DB2 Path Checker command reference</td>
<td>141</td>
</tr>
<tr>
<td>Syntax rules for specifying commands</td>
<td>144</td>
</tr>
<tr>
<td>Wildcard characters in command syntax</td>
<td>145</td>
</tr>
<tr>
<td>Query number considerations</td>
<td>145</td>
</tr>
<tr>
<td>COMPARE command</td>
<td>145</td>
</tr>
<tr>
<td>CONNECT command</td>
<td>148</td>
</tr>
<tr>
<td>EXPLAIN command</td>
<td>149</td>
</tr>
<tr>
<td>EXPLAIN DBRM command</td>
<td>149</td>
</tr>
</tbody>
</table>
Figures

1. DB2 Path Checker processing flow ........................................ 13
2. The Tools Customizer Settings panel (CCQPSET) .................................. 37
3. The Panel Display Options panel (CCQPOPT) ..................................... 39
4. The Specify the Metadata Library panel ........................................ 44
5. The Discover Customized Product Information panel ........................ 46
6. The Associate DB2 Entry for Product panel ...................................... 47
7. The Create a DB2 Entry panel ...................................................... 48
8. The Associate DB2 Entry for Product panel with a new DB2 entry in the master list 48
9. The Product Parameters panel ..................................................... 50
10. The DB2 Parameters panel .......................................................... 53
11. The Finish Product Customization panel ....................................... 56
12. Synonym generation example ....................................................... 64
13. Bind control commands .............................................................. 69
14. DB2 Path Checker commands ....................................................... 69
15. Identified packages ................................................................. 69
16. Generated command ................................................................. 69
17. DB2 Path Checker Interactive main menu ...................................... 82
18. DB2 Path Checker Interactive - SETTINGS panel .............................. 82
19. DB2 Path Checker Interactive - REPORT panel ................................. 83
20. DB2 Path Checker Interactive sample report ................................... 84
21. DB2 Path Checker Interactive - COMPARE (package) panel ................. 85
22. DB2 Path Checker Access Path Differences (Summary) sample report .... 86
23. DB2 Path Checker Interactive - COMPARE (plan) panel .................... 87
24. DB2 Path Checker Access Path Differences (Summary) sample report .... 88
25. Sample TEST report ................................................................. 90
26. The DB2 Path Checker Interactive - BATCH panel ............................ 91
27. DB2 Path Checker Interactive - Utilities panel .................................. 97
28. DB2 Path Checker Interactive - Create PLAN_TABLE panel ................. 98
29. DB2 Path Checker Interactive - Utilities panel .................................. 98
30. DB2 Path Checker Interactive - Clear PLAN_TABLE panel ................. 99
31. RPTPKG sample long report ....................................................... 106
32. JCL for the EXPLAIN sample report ........................................... 106
33. SYSPRINT report example ......................................................... 107
34. SYSEXPLN report example ......................................................... 107
35. SYSCHG report example ........................................................... 108
36. SYSPRINT report example with access path structure change .......... 108
37. History table DDL ................................................................. 109
38. JCL for comparing access paths for two existing binds .................... 110
39. Report for comparing access paths for two existing binds ................ 111
40. REPORT command sample ......................................................... 112
41. MAKE command sample .......................................................... 113
42. JCL for TEST example ............................................................. 114
43. TEST sample long report ........................................................ 115
44. JCL for specifying commands in SYSIN ....................................... 116
45. Sample JCL for DB2 Path Checker SQL PA interface, Part 1 ............... 171
46. Sample JCL for DB2 Path Checker SQL PA interface, Part 2 ............... 172
47. Sample SQL PA report on DB2 Path Checker SQL, Part 1 ................. 173
48. Sample SQL PA report on DB2 Path Checker SQL, Part 2 ................. 174
## Tables

1. Customization roadmaps ........................................... 40
2. Steps for customizing DB2 Path Checker for the first time ........... 40
3. Administrative tasks .................................................. 41
4. Steps for customizing a new version of DB2 Path Checker from a previous customization ........... 41
5. Administrative tasks .................................................. 42
6. Required steps for recustomizing DB2 Path Checker ................... 42
7. Administrative tasks .................................................. 43
8. DB2 Path Checker commands, with related data sets that store the output .................. 101
9. DB2 Path Checker return codes ..................................... 119
10. Status types for the product, the LPAR, and the DB2 entries ........ 141
11. Data set attributes for allocating the Discover output, data store, and customization library data sets 143
About this information

This information provides instructions for using IBM® DB2® Path Checker for z/OS®.

This information is intended for people who understand the DB2 application development process and the various DB2 libraries. It is also useful for application programmers and system analysts who need to understand the functions provided by DB2 Path Checker in the application development process. If you are responsible for installing and supporting DB2 Path Checker, a basic familiarity with MVS™ utilities and JCL is required and knowledge about DB2 BIND is required.

These topics are designed to help database administrators, system programmers, application programmers, and system operators perform the following tasks:

- Plan for the installation of DB2 Path Checker
- Install and operate DB2 Path Checker
- Customize your DB2 Path Checker environment

To use these topics, you should have a working knowledge of:

- The OS/390® or z/OS operating system
- ISPF
- SMP/E

Specific changes since the previous edition of this book are indicated by a vertical bar (|) to the left of a change. Editorial changes that have no technical significance are not noted.

Always check the DB2 Tools Product publications page for the most current version of this information:


Service updates and support information

To find service updates and support information, including software fix packs, PTFs, Frequently Asked Questions (FAQs), technical notes, troubleshooting information, and downloads, refer to the following Web page:

www.ibm.com/software/data/db2imstools/support.html

How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>>>> symbol indicates the beginning of a syntax diagram.
  - The --> symbol indicates that the syntax diagram is continued on the next line.
- The >--- symbol indicates that a syntax diagram is continued from the previous line.
- The ---< symbol indicates the end of a syntax diagram.

- Required items appear on the horizontal line (the main path).

  \[\text{required_item}\]

- Optional items appear below the main path.

  \[\text{required_item} \quad \text{optional_item}\]

  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

  \[\text{required_item} \quad \text{optional_item}\]

- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.

  \[\text{required_item} \quad \text{required_choice1} \quad \text{required_choice2}\]

  If choosing one of the items is optional, the entire stack appears below the main path.

  \[\text{required_item} \quad \text{optional_choice1} \quad \text{optional_choice2}\]

  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.

  \[\text{required_item} \quad \text{default_choice} \quad \text{optional_choice} \quad \text{optional_choice}\]

- An arrow returning to the left, above the main line, indicates an item that can be repeated.

  \[\text{required_item} \quad \text{repeatable_item}\]

  If the repeat arrow contains a comma, you must separate repeated items with a comma.

  \[\text{required_item} \quad \text{repeatable_item}, \quad \text{repeatable_item}\]
A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, column-name). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

---

**How to send your comments**

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other DB2 Path Checker for z/OS documentation:

- Use the online reader comment form, which is located at: [www.ibm.com/software/data/rcf/](http://www.ibm.com/software/data/rcf/)
- Send your comments by e-mail to comments@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of DB2 Path Checker, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).
Chapter 1. DB2 Path Checker overview

IBM DB2 Path Checker for z/OS provides information about potential access path changes before they occur, and can also report changes that occur after you rebind a plan or package.

DB2 Path Checker allows DB2 programmers, system analysts, and database administrators (DBAs) to determine whether a bind of a database request module (DBRM) results in an access path change. DB2 Path Checker can test proposed or actual changes in plans and create a report of changed and unchanged paths, optionally providing cost estimates and DB2 catalog statistics for path changes.

What’s new in DB2 Path Checker

This topic summarizes the technical changes for this edition.

New and changed information is indicated by a vertical bar (|) to the left of a change. Editorial changes that have no technical significance are not noted.

SC19-4141-01

"Up and running" enhancements to customization documentation

Enhancements to the User’s Guide include customization information and checklists to help you get DB2 Path Checker up and running more quickly and easily. New sections provide information about preparing to customizeDB2 Path Checker, setting up your environment, gathering required data set names, and gathering parameter values.

For more information, see Chapter 2, “Customizing DB2 Path Checker by using Tools Customizer,” on page 21.

New keywords for the OPTIONS command

Two new keyword specifications were added to the OPTIONS command to enable you to add the options DESCSTAT and APPLCOMPAT to DB2 BIND commands that DB2 Path Checker generates. The following new messages support the enhancement:

- CKP1145E
- CKP1146E

For more information, see “OPTIONS command” on page 154.

SC19-4141-00

Support for IBM Tools Customizer for z/OS

DB2 Path Checker adds support for Tools Customizer in this version. You can now take advantage of the single, consistent ISPF interface, and standardized customization processes that are common to multiple IBM tools that run on z/OS. A Discover EXEC discovers values for common product, LPAR, and DB2 parameters from previous customization processes that you did manually.

For more information, see Chapter 2, “Customizing DB2 Path Checker by using Tools Customizer,” on page 21.
EXPLAIN PACKAGE command

A new EXPLAIN PACKAGE command is now available to process a specific bind of a package. The command generates EXPLAIN data in your plan table that reflects the current state of your DB2 environment. EXPLAIN PACKAGE is useful in situations where you want to issue more EXPLAIN DBRM or COMPARE commands later on the most accurate access path data available. You can also use it to populate a named plan table with access path data. This plan table can be different from the default owner.PLAN_TABLE.

See “EXPLAIN PACKAGE command” on page 151 for a description of this new command. For task-oriented information and a sample batch job, see “Generating access-path information for a package” on page 71.

APREUSE and APCOMPARE options for generated REBIND commands

New options in this release can specify that DB2 Path Checker automatically adds APREUSE and APCOMPARE rebind options to generated REBIND commands. You must specify a ddname for RBINDOUT or PBINDOUT to use this feature. Set the DB2 option PLANMGMT(EXTENDED) with this feature to preserve access path information from previous binds.

See “OPTIONS command” on page 154 for descriptions of these new options.

Early warning for plans and packages bound prior to DB2 Version 9

DB2 Path Checker issues a warning if DB2 will rebind your plan or package automatically at run time. This feature applies only to programs that you run in DB2 Version 11 and were last bound before DB2 Version 9. For more information, see “What does DB2 Path Checker do?” on page 3.

Terminology

The following terms and definitions help you to understand the documentation for DB2 Path Checker.

Access path

The method that is selected by the database manager for retrieving data from a specific table. For example, an access path can involve the use of an index, a sequential scan, or a combination of the two.

Application

A program or set of programs that performs a task, for example, a payroll application.

Collection

A group of packages that have the same qualifier.

Database request module (DBRM)

In DB2 for z/OS, a data set member that is created by the DB2 precompiler and that contains information about SQL statements. DBRMs are used in the bind process.

Package
An object that contains a set of SQL statements that have been statically bound and that is available for processing. A package is sometimes also called an application package.

Plan
The control structure that is produced during the bind process. DB2 uses the application plan to process SQL statements that it encounters during statement execution.

A plan is sometimes also called an application plan.

Plan table
The plan table, typically owner.PLAN_TABLE, contains information about access paths that is collected from the results of EXPLAIN statements.

Statement table
The statement table, DSN_STATEMNT_TABLE, contains information about the estimated cost of specified SQL statements.

What does DB2 Path Checker do?

DB2 Path Checker helps database administrators and application developers to optimize the performance of SQL statements and maximize the efficiency of application development and implementation on DB2.

Overview
As an application platform and enterprise data repository, DB2 continues to become more mission-critical. Effective and efficient use of DB2 is often vital to the success of an enterprise. The online operation of the database must be robust and reliable. And the process of developing and implementing applications must be as efficient and problem-free as possible.

DB2 Path Checker addresses these requirements by helping you to manage performance issues that can arise as a result of changes to access paths that degrade performance. You can use DB2 Path Checker to identify potential access path changes before they occur. DB2 Path Checker helps you to do what if testing to determine which access paths might change because of development, migration, or system activities. You can use this information to select a course of action without affecting your production DB2 environment.

You typically run DB2 Path Checker when you do the following actions:

- Install a new release or version of DB2
- Apply service maintenance
- Deploy a large application or migrate it from one DB2 subsystem to another

With DB2 Path Checker, you can see the effects of doing a bind before you do the actual bind process. You can also evaluate the effects after you do a rebind. Based on the results, you might choose to rewrite SQL statements or defer rebinding the application.

DB2 Path Checker stores command output in user-specified DB2 plan tables. You can then review the access paths to analyze or predict changes between different points in time.
DB2 Path Checker warns you if DB2 will rebind your plan or package automatically at run time. This feature applies only to programs that you run in DB2 Version 11 and were last bound before DB2 Version 9. The message CKP1142W is displayed when you run REPORT, COMPARE, or TEST commands. The notification helps you to anticipate or avoid performance degradation because of the rebind operation when you run the program in production for the first time.

DB2 Path Checker helps you to analyze changes to access paths in the following ways:

- Through the use of EXPLAIN processing
- By comparing binds and EXPLAIN results
- Through the use of the TEST command

**Explain processing**

You can use the EXPLAIN DBRM and EXPLAIN PACKAGE commands to determine how changes in your DB2 environment might affect access paths over time. EXPLAIN processing is especially helpful in migration scenarios, where you can use DB2 Path Checker to determine how SQL statements will behave after migration to a new version.

You can evaluate these effects without affecting your production environment. You do not have to bind database request modules (DBRMs) or rebind packages. DB2 Path Checker populates your plan table with the access paths that are in effect at the specified point in time. These access paths can reflect the current state of your DB2 environment or a snapshot view at the time of a previous bind. This information can then become the basis for later comparisons and reports.

**Comparisons**

You can use the COMPARE command to determine which access paths changed between binds or between a bind and previously generated EXPLAIN results in a plan table.

The COMPARE command is a good way to generate a list of access paths that changed since the most recent edition or version of your application. For example, if the performance of an application is questionable, you can compare the current version of the package to the previous version.

The COMPARE command compares access path data for two plans or packages. Unlike the TEST command, COMPARE does no EXPLAIN processing. Instead, the command compares existing rows in plan tables. These rows contain access paths that can be the result of binds, rebinds, or EXPLAIN processing. By analyzing the output, you can identify any changes to the SQL statements and determine what caused the access paths to change.

If the SQL did not change, perhaps changes in your DB2 environment caused the access paths to change. If so, closer examination of the specific changes in the access paths is required.

**Tip:**

Consider analyzing your SQL statements in DB2 SQL Performance Analyzer to generate expert advice automatically and help you tune your queries.
The COMPARE command supports the following methods:

**Normal comparison**

This method compares the most recent bind of a specified program to the most recent instance of the same program in a different plan table.

Specify the program that you want to compare with its earlier access paths. The program name can contain wildcard characters. Specify plan names or collection names as qualifiers for the program name. These qualifiers can be the same or different for the two plan tables.

More options help you to select a specific bind time or program name with which to begin processing. These capabilities can be useful in situations where you must restart a long report.

**Compare-to-previous**

This method compares the most recent instance of the program that you specify to the next oldest instance in the same plan table.

Specify the earlier edition or version of the program to compare with the most recent instance by specifying one the following options:

- `PREVIOUS`
  - The previous edition of the current version of the program

- `PREVIOUS VERSION`
  - The most recent bind of the previous software version of the program

The COMPARE command selects members to process from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE (or the equivalent shadow table). The plan tables can have different formats, but the formats must be valid for the version of DB2 that you are running.

**Tests**

You typically use the **TEST** command to evaluate an unbound database request module (DBRM) with the current edition or version of the package or plan. The **TEST** command generates reports that indicate which access paths will change if you rebind the program.

The **TEST** command first generates fresh explain data to determine what the current access paths would be if you rebound the specified program. It then compares these newly generated access paths with a previous bind of the plan or package that you specify. You can test the DBRM against plan table entries before a specified time stamp or begin processing with a specific program name.

The **TEST** command compares access paths from the following sources:

- EXPLAIN data as of the bind time of the program that is running in DB2
- EXPLAIN data that an EXPLAIN command generated into a plan table
- EXPLAIN data at the time of a previous bind
  - That bind can be a previous edition of the current version of the program or a previous program version.

One common scenario for using the **TEST** command is a migration to a new version of DB2. You can create a plan table in the new version, and then load it with access
path information from the previous version. When you issue the TEST command, you specify the newly created plan table that contains the data from the previous version. For more information, see “Scenario: Migrate to a new DB2 version” on page 15.

DB2 Path Checker uses the plan table that you specify as input, regardless of whether it contains EXPLAIN data for the actual bind time of the program. If the EXPLAIN data is not available, the TEST command does any required explain processing and populates the plan table with the required access paths.

The TEST command processes only SQL that can be processed by an EXPLAIN command. During command processing, DB2 Path Checker ignores SQL statements such as OPEN CURSOR, CLOSE CURSOR, and UPDATE CURRENT OF CURSOR.

If you use the DBRMLIB as input instead of the FROM CATALOG option, the TEST command requires a ddname for DBRMIN. The ddname must refer to a valid DBRMLIB or concatenation of DBRMLIBs. If you specify OPTIONS REPORTALL, DB2 Path Checker writes all the access paths instead of writing only those access paths that changed.

Related concepts:
“Scenario: Migrate to a new DB2 version” on page 15
In this scenario, the IT staff is migrating to a new version of DB2. The system DBA wants to test how their applications are likely to perform in the new version and do the migration in stages.

Related reference:
“EXPLAIN command” on page 149
Follow the syntax and descriptions in these topics to use the EXPLAIN commands that are available in DB2 Path Checker.

“COMPARE command” on page 145
Follow the syntax and descriptions in this topic to use the COMPARE command.

“TEST command” on page 168
Follow the syntax and descriptions in this topic to use the TEST command.

“EXPLAIN PACKAGE command” on page 151
The EXPLAIN PACKAGE command extracts the access path information for a package from the DB2 directory.

**DB2 Path Checker features and benefits**

DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.

**Overview**

As your database environment grows in size and complexity, unexpected access path changes often require valuable resources to rectify. DB2 Path Checker helps you to assess changes to the database that might cause access path changes.

Additionally, DB2 Path Checker enables you to input a set of proposed or even hypothetical changes to the database, and then estimate how those changes affect access paths. DB2 Path Checker has many capabilities that help you to predict changes to access paths, anticipate how changes in your DB2 environment affect access paths, and mitigate the unintended performance consequences of change.
When you bind or rebind a program, DB2 reevaluates the access paths that it uses to read or write the data. Often, this reevaluation results in the selection of different access paths. These new paths might or might not be the most efficient for a specific query. The efficiency might depend on changes that are made to the databases since the initial bind, or other changes in your DB2 environment. Sometimes, the first indication that the access paths changed is when the response time of an application worsens for no apparent reason.

DB2 Path Checker helps you to avoid unexpected degradations of performance in your production environment by either proactively anticipating changes to access paths or evaluating them subsequent to a bind.

**Capabilities**

DB2 Path Checker can help you to determine quickly whether a bind of a new DBRM or a rebind of an existing DBRM will result in changed access paths.

You can use the following features of DB2 Path Checker to identify and analyze changes to access paths:

- Determine which access paths change when you migrate between DB2 subsystems.
- Perform EXPLAIN processing on a DBRM to populate owner.PLAN_TABLE.
- Determine which DBRMs will have access path changes if you rebind them.
- Compare the access paths for a bound plan or package with a previous edition or version of the program.
- Create optimization hints in owner.PLAN_TABLE for access paths of specific statements.
- Use lists and pattern matching to process a batch of DBRMs in a single pass.
- Use wildcards for plan and package names to select programs for explain processing, comparison, testing, and reporting.
- Use matching logic to compare two sets of EXPLAIN results that are based on different types of criteria, such as the SQL query number. The SQL can change position in the program or change position in relation to other SQL, and DB2 Path Checker can still match the SQL statements for comparison.
- Find DBRMs that have no matching EXPLAIN results.

The following sections describe some of the advantages of having these capabilities.

**Use DBA time efficiently**

DB2 Path Checker is only one of several DB2 Tools products that help database administrators to work effectively. Database administrators are responsible for making the database environment run smoothly and efficiently. Part of that responsibility is to ensure that applications that are running in the environment perform at a high level.

The following examples describe some of the challenges that they face:

- Determining access path changes when you migrate to the next version of DB2
- Doing access path analysis when an application performs poorly
- Identifying potential access path changes when you deploy a new version of an application
• Determining whether to rebind an existing application

Given the right set of tools, database administrators can become more productive and less dependent on systems staff.

**Identify changes to access paths without affecting your DB2 environment**

DB2 Path Checker can test access paths without binding the associated DBRM or package. The program does not bind or rebind existing plans or packages, so it has no effect on your production environment. Therefore, you can do *what if* analysis with complete transparency.

DB2 Path Checker can test each SQL statement in a DBRM by issuing an EXPLAIN request to determine the access path for that statement. It then compares this newly generated access path to the existing access path in a separate plan table and generates a report. The report shows both the old and new access paths. You can configure this report to show all access paths or just those access paths that changed.

You can also use DB2 Path Checker to evaluate the effects of a bind. You do this by comparing the newly generated access paths to the previous access path information in a separate plan table. DB2 Path Checker generates a report that shows the results of the comparison.

DB2 Path Checker works with any valid DB2 plan table format. If plan tables from different versions of DB2 have a different number of columns, DB2 Path Checker compares as many columns as possible based on the data that is available.

You can use DB2 Path Checker to safely analyze any of the following types of changes:

- Migrations to different environments, such as migrating to a new version of DB2
- Changes to catalog statistics
- Changes that the RUNSTATS online utility created
- Changes to hardware

**Make better-informed decisions**

The choices that DB2 makes during access path selection directly affect your evaluation of performance issues for migrations, service maintenance, or new application versions. DB2 Path Checker helps you to gather the information that you need to decide whether to bind or rebind programs, or to reuse existing access paths. The following examples describe some of the questions that you might consider as you evaluate changed access paths:

- Is a UNION or a JOIN the optimal choice for this query?
- Is a table space scan or an index optimal?
- Was a required index omitted?
- Was an index dropped unexpectedly?
- Is a different choice of index best?
- Is it time to reorganize the database?
- Is it a good idea to run RUNSTATS to update the catalog statistics?
DB2 statistics describe the data characteristics of your table spaces, indexes, and partitions. Factors such as frequency, cardinality, number of table rows, number of index pages, cluster ratios, or index tree levels can affect access plan selection.

For more information, see "Scenario: Test a package against the previous BIND" on page 14.

Streamline the migration process

Path Checker helps you to anticipate the access path changes that occur when you migrate your applications to a new version of DB2. You can anticipate these changes without having to BIND the applications in the new version.

For more information, see "Scenario: Migrate to a new DB2 version" on page 15.

Optimize the performance of new or upgraded application programs

You can use DB2 Path Checker to compare a new version of an application package to the previous version and identify SQL statements that changed. This comparison can help you to narrow down your search for inefficient statements. You can also do explain processing on new versions and compare the results with the active version in DB2.

For more information, see "Scenario: Deploy a new version of an application" on page 16.

Optimize SQL performance and simplify migrations by using other DB2 Tools with DB2 Path Checker

You can use other DB2 Tools products with DB2 Path Checker to further optimize performance, eliminate unnecessary binds, and anticipate problems before they occur. For example, you can use DB2 SQL Performance Analyzer to expand the analysis of the SQL statements that DB2 Path Checker selects when access paths change.

For more information, see "Avoid unnecessary binds by using DB2 Bind Manager" on page 17 and "Retrieve hints, recommendations, and expert advice from SQL Performance Analyzer" on page 17.

Related concepts:

"Scenario: Test a package against the previous BIND" on page 14
In this scenario, the IT staff at the bank is considering rebinding its mortgage application after it adds several new indexes to their DB2 database.

"Scenario: Migrate to a new DB2 version" on page 15
In this scenario, the IT staff is migrating to a new version of DB2. The system DBA wants to test how their applications are likely to perform in the new version and do the migration in stages.

"Scenario: Deploy a new version of an application" on page 16
In this scenario, the bank deploys a new version of its ATM application. The IT staff wants to anticipate any potential performance issues by analyzing the changed access paths before putting the new version into production.

"Retrieve hints, recommendations, and expert advice from SQL Performance Analyzer" on page 17
You can use DB2 Path Checker with IBM DB2 SQL Performance Analyzer for z/OS for more detailed cost analysis of the SQL statements that DB2 Path Checker
selects.

"Avoid unnecessary binds by using DB2 Bind Manager” on page 17

You can use DB2 Path Checker and IBM DB2 Bind Manager for z/OS at different stages of your deployment process to avoid doing unnecessary binds.

Related tasks:

"Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80

Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.

"Comparing access paths for different binds of the same program version” on page 78

Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

Data requirements

DB2 Path Checker relies on information about access paths that is in owner.PLAN_TABLE and the matching DSN_STATEMNT_TABLE.

Examples of operations that can generate this data include the following sources:

- **BIND** commands that use the EXPLAIN(YES) or EXPLAIN(ONLY) options
- SQL EXPLAIN statements
- **EXPLAIN** commands that you issue yourself in DB2 or DB2 Path Checker, such as EXPLAIN PACKAGE

Before you use DB2 Path Checker, you must do one of these operations to populate the plan table with access path data. To ensure that the EXPLAIN data is accurate and complete, use the EXPLAIN(YES) option with your **BIND** commands. For more information, see the section about binding an application in the DB2 for z/OS Application Programming and SQL Guide. See also the descriptions of the EXPLAIN bind option in the DB2 for z/OS Command Reference.

DB2 modifies the plan table definition when it releases a new version. For example, previous versions of DB2 changed the data types and lengths of many of the columns. DB2 Path Checker compares access path information for any valid DB2 plan table. It can also compare plan tables that are from different releases of DB2.

DB2 Path Checker expects the DSN_STATEMNT_TABLE and the PLAN_TABLE that are being processed to have the same creator ID. When a **REPORT** or **TEST** command specifies a target **owner**.PLAN_TABLE, the estimated cost information is retrieved from **owner**.DSN_STATEMNT_TABLE. When a **COMPARE** command references two different plan tables, DB2 Path Checker processes the cost information from the matching DSN_STATEMNT_TABLEs. When a **COMPARE** command references the same plan table but different time stamp values for the binds, the data values match the correct information in DSN_STATEMNT_TABLE. The DSN_STATEMNT_TABLE is not required for normal DB2 Path Checker operation. If the DSN_STATEMNT_TABLE is not available, OPTIONS REPORTCOSTGT does not work for the **COMPARE** and **TEST** commands.

Examples

Problem 1

Multiple programs exist that do not contain access data in the PLAN_TABLE.
Solution 1
Issue the **REBIND EXPLAIN(YES)** command to generate access path data into the PLAN_TABLE and generate new access paths. Issue the **REPORT** command to view the new access paths. Check the run times for the new access paths to determine whether they increase or decrease.

Solution 2
Update SCKPSAMP member CKPEXP01 for plans that do not contain data and run the generated DB2 Path Checker **EXPLAIN** commands to find the probable access paths.

Update the following members for packages that do not have access data:
- SCKPSAMP member CKPEXP02 (no versioning for DBRMs)
- SCKPSAMP member CKPEXP03 (checks for version match in PLAN_TABLE)

Then run the generated DB2 Path Checker **EXPLAIN** commands to find the probable access paths.

Review the access paths for their suitability in your environment. If there are no obvious problems, run the **REBIND EXPLAIN(YES)** and run the **COMPARE** commands to verify that bind processing chooses the same access paths. This solution has the following advantages:
- The DB2 Path Checker **EXPLAIN** commands do not update the access paths. If there are problems with the catalog statistics or the access paths that were chosen, run RUNSTATS and then rerun the **EXPLAIN** commands.
- The explain data establishes a baseline for the expected access paths and requires that you have access to the correct DBRM.

Problem 2
You are migrating a set of programs from a test environment to a production environment. The test database is in the same DB2 subsystem as the production database, but the qualifier IDs are different.

1. Before you begin the migration process, you must determine what access path changes occurred with the new programs.

Solution 1
Compare the access paths in the test database to the existing access paths in the production database by issuing the following commands:

```sql
OPTIONS NOMATCHCREATOR

COMPARE PACKAGE acctpayid.progname IN test.PLAN_TABLE TO prodcollid.* in prod.PLAN_TABLE
```

This command identifies new, changed, and deleted SQL statements and access path changes where the same table is referenced with the same or different SQL.

Solution 2
Specify the DBRMIN ddbname to be the new DBRMLIB and issue the following commands:

```sql
SET CURRENT SQLID = 'prod'
TEST DBRM progname as prodcollid.* in prod.PLAN_TABLE
```
Solution 3

For new programs that do not have any access data in production, issue the following commands:

```
SET CURRENT SQLID = 'prod'
EXPLAIN DBRM dbrmname
```

2. You must determine whether the programs retrieved the same access paths in the production database that were verified in the test database.

Solution

After the binds for the programs in the production database are finished, compare the access paths in the production database to the access paths that were tested. Do this by issuing the following commands:

```
OPTIONS NOMATCHCREATOR

COMPARE PACKAGE collid.progname IN prod.PLAN_TABLE TO acctpayid.* in test.PLAN_TABLE.
```

The output from the `COMPARE` command highlights the access paths that differ between the test and production systems. With this information, you can determine whether changed access paths are likely to decrease performance. You can then take appropriate preventive action before the changes go into effect.

Problem 3

All the packages in a subsystem must be rebound because you are migrating them to a new version for general maintenance. Current data exists in PLAN_TABLE and DSN_STATEMNT_TABLE, but you do not have enough time to review changes to access paths.

Solution

Rebind only packages that are safe to rebind. Set up `TEST` commands for every package in the subsystem, and specify the RBINDOUT ddbname in the execution JCL. Only packages that do not have access path changes have `REBIND` commands in ddbname RBINDOUT. Packages that have access path changes are stored so that they can be reviewed later.

Problem 4

Jobs are scheduled to run RUNSTATS and rebind all the packages, but rebinds disrupt production and take too long.

Solution

Set up `TEST` commands for all packages. Specify `OPTIONS CPUPCT2` `nnn`. The only `REBIND` commands that DB2 Path Checker writes to RBINDOUT are for packages that meet one of the following criteria:

- No access paths changed.
- The access paths have a PROCMS value that is less than or equal to the old PROCMS plus the percentage that is specified in `OPTIONS CPUPCT2`. 

Architecture and process flow

DB2 Path Checker can accept input data from the DB2 catalog, plan tables, or the DBRM library (DBRMLIB) and can send output access path information to flat files, reports, or XML files.

DB2 Path Checker reads existing access paths or does EXPLAIN processing to generate fresh access paths. The access paths can come from any of the following sources:

- Previous binds of a program that you did with the EXPLAIN(YES) or EXPLAIN(ONLY) options
- Programs that you bound in previous versions of DB2
- Unbound DBRMs for which you want to test access paths

The following figure describes the basic process flow for DB2 Path Checker:

![Diagram of DB2 Path Checker processing flow]

Figure 1. DB2 Path Checker processing flow

The following descriptions can help you to understand the inputs and outputs in the figure:

**DB2 catalog**

DB2 Path Checker can read the access path data in the DB2 system catalog directly. The information in the catalog is the most accurate information available about the access paths that DB2 generated at bind time.

**Plan table**

Plan tables are a key source of information about access paths. You can use the default plan table that uses the format user.PLAN_TABLE as a source or a different plan table.

**DBRMLIB**
DB2 Path Checker can read the DBRMs in the DBRM library (DBRMLIB) directly and generate access paths during run time that reflect the current state of your DB2 environment. These access paths are the same ones that DB2 would generate if you bound the program instead of running DB2 Path Checker.

In DB2 Path Checker, you can think of a DBRM as source code that contains the SQL for a bind of a plan or a package. DB2 Path Checker can process a DBRM as a program that is related to a specific business application, such as payroll, accounts receivable, or inventory.

Plan table 2
The second plan table in the figure represents an alternative source of access path information to the default plan table. The following examples describe some of the ways that you can use a different plan table:

- You might want to compare or test against an earlier version of your plan table.
- You might create plan tables of your own to store access plans from a previous version of DB2 during a migration.
- You might issue `EXPLAIN` commands to populate a plan table with access paths for later comparisons or tests.

Flat files
DB2 Path Checker can share information with other DB2 Tools products, such as DB2 SQL Performance Analyzer, by generating flat files.

Reports
DB2 Path Checker produces a one- or two-line report for each row in the plan table. The report provides a summary of access paths that changed. If you specify the ALL option, the report includes all access paths.

XML files
DB2 Path Checker can share information with other DB2 Tools products, such as DB2 SQL Performance Analyzer, by generating XML files.

---

**DB2 Path Checker scenarios**

In the following scenarios, a global banking enterprise uses DB2 Path Checker to anticipate and manage performance issues that result from changed access paths.

**Scenario: Test a package against the previous BIND**
In this scenario, the IT staff at the bank is considering rebinding its mortgage application after it adds several new indexes to their DB2 database.

The system database administrator (DBA) wants to determine how many access paths in the application will change as a result of a new bind in the changed environment.

The DBA creates a job for DB2 Path Checker that tests the source program for the version of the application that is running in the DB2 subsystem. The job has no effect on the DB2 subsystem because DB2 Path Checker reads the DBRM, does `EXPLAIN` processing on the SQL statements, and compares the results. By using DB2 Path Checker, the DBA does not have to do a bind or rebind with the `EXPLAIN(ONLY)` option. The `TEST` command generates the new access path.
information in the plan table. The test compares the newly generated access paths with the access paths for the version that is running.

The job generates a report that shows which access paths will change if the IT staff rebinds the packages that support the application. In addition to providing information about the changed paths, the report adds more details about the changes.

After examining the changed access paths more closely, the DBA learns that some of the paths will use a new index instead of a table scan. The index is likely to improve performance. In other cases, system characteristics changed and the DBA decides to run the DB2 RUNSTATS utility to generate new catalog statistics.

After RUNSTATS populates the DB2 system catalog with new statistics, the DBA runs the job again. The new report reveals that the data distribution within a table does not lend itself to using an index. The DBA removes the index.

This package was last bound in a prior release. The DBA decides to rebind the mortgage application to take advantage of the new indexes and use memory and performance enhancements in the newer version of DB2 that they are running. One package in the application is likely to perform more poorly because the characteristics of an index changed. For this package, the DBA issues a MAKE command in DB2 Path Checker to create a hint in the plan table before rebinding the package. This approach prevents DB2 from generating an updated access path during rebind operations.

Thanks to DB2 Path Checker, the IT staff can rebind the mortgage application with confidence. They know that in most cases the rebound packages perform better. The package that does not run faster uses the access path from the current version of the application.

Related concepts:

“DB2 Path Checker features and benefits” on page 6
DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.

Related reference:

“TEST command” on page 168
Follow the syntax and descriptions in this topic to use the TEST command.

“MAKE command” on page 153
Follow the syntax and descriptions in this topic to use the MAKE command.

**Scenario: Migrate to a new DB2 version**

In this scenario, the IT staff is migrating to a new version of DB2. The system DBA wants to test how their applications are likely to perform in the new version and do the migration in stages.

The DBA needs the ability to do staged migrations in a modular fashion. The DBA wants to create named plan tables that the IT staff can use to stage major migrations according to application boundaries and migration teams. DB2 Path Checker makes this process much easier.

DB2 Path Checker generates access paths for the packages that support each application and tests them against the access paths from the previous version of DB2. To accomplish this task, the DBA must populate plan tables for the new version with access paths from the previous version and then test the DBRM.
The DBA creates an UNLOAD job that unloads the plan table and EXPLAIN tables from the previous release. The DBA then creates a LOAD job to populate the plan table and the EXPLAIN tables in the new release. The DBA then creates jobs in DB2 Path Checker to generate access paths for the packages that support each application. The DBA then compares those access paths with the access paths from the previous version.

By testing the applications proactively, the staff is able to anticipate many potentially harmful changes to access paths before they do the migration. They can partition risk by migrating one application at a time. And they drastically reduce the amount of catalog data that they must migrate.

The system DBA was surprised when the application DBAs added one of the batch jobs that the DBA created to the compile list for their builds. Application developers now understand how to use DB2 Path Checker to anticipate the effects of changes in their applications. The preventive measures that DB2 Path Checker provides are now integrated with the application development process at the bank.

**Related concepts:**
- “DB2 Path Checker features and benefits” on page 6
- DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.
- “What does DB2 Path Checker do?” on page 3
- DB2 Path Checker helps database administrators and application developers to optimize the performance of SQL statements and maximize the efficiency of application development and implementation on DB2.

### Scenario: Deploy a new version of an application

In this scenario, the bank deploys a new version of its ATM application. The IT staff wants to anticipate any potential performance issues by analyzing the changed access paths before putting the new version into production.

The application developers are complaining about poor performance of the new version in the test environment. The DBA first creates a DB2 Path Checker job to compare the application packages against the previous version. The DBA runs a `COMPARE PACKAGE` command with the `NOMATCHCREATOR` option to identify changed SQL statements. After analyzing the output, the DBA is able to identify the SQL statements that changed and give them further scrutiny.

DB2 Path Checker helps the DBA to find the SELECT statement that caused the inefficient query. The access path for the offending SELECT statement was changed to use a table space scan instead of using an index. An application developer corrects the SELECT statement so that DB2 uses an index.

After testing the application again, the developers verify that the query performance is now acceptable. The DBA runs the job again to generate a new report, and then shares the report with the developers to confirm that the application performs as intended.

**Related concepts:**
- “DB2 Path Checker features and benefits” on page 6
- DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.

**Related reference:**
- “COMPARE command” on page 145
  Follow the syntax and descriptions in this topic to use the `COMPARE` command.
Avoid unnecessary binds by using DB2 Bind Manager

You can use DB2 Path Checker and IBM DB2 Bind Manager for z/OS at different stages of your deployment process to avoid doing unnecessary binds.

Tool efficiency to reduce costs

Two tools are available from IBM to help you to avoid costly and unexpected results from unnecessary bind operations. DB2 Path Checker helps you to identify, analyze, and document in advance potential access path changes without initiating bind processing. DB2 Bind Manager automatically identifies packages where the structure of the SQL statements did not change, which also helps you to avoid unnecessary binds.

DB2 Bind Manager

After the DB2 precompiler successfully processes a program with embedded SQL, DB2 Bind Manager analyzes the DBRM in the DBRMLIB. It then compares that DBRM with the DBRM from the precompile step to determine whether the structure of the SQL changed. If not, then a bind is not required. Since DB2 Bind Manager detects only production applications that require a bind, it helps you to concentrate on changes that affect the SQL structure.

DB2 Bind Manager offers these features:

- You can select individual DBRMs from the DBRM library by using an interactive system productivity facility (ISPF) interface. You can then compare them against the subsystem and the load module to verify that you are implementing the correct program. If you have missing DBRMs, you can then recreate them from the system catalog.
- Using Lists and Pattern Matching, you can compare batches of DBRMs against subsystems and load modules in one pass.
- The DBRM Checker feature in DB2 Bind Manager can analyze a load module and determine which programs and which modules within those programs contain static SQL.
- Another function in DB2 Bind Manager does mass comparisons between a DB2 subsystem and different load libraries. Where there is no matching load module, the program creates a FREE command for the package and then generates a backup of the DBRM. This action reduces the amount of catalog data that you must convert and the number of DBRMs that require a rebind.
- You can use the Catalog cleanup feature in DB2 Bind Manager to compare bound packages in a DB2 subsystem to one or more load libraries. Generate FREE commands to remove obsolete or unreferenced packages from the catalog.

Related concepts:

- “DB2 Path Checker features and benefits” on page 6

DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.

Retrieve hints, recommendations, and expert advice from SQL Performance Analyzer

You can use DB2 Path Checker with IBM DB2 SQL Performance Analyzer for z/OS for more detailed cost analysis of the SQL statements that DB2 Path Checker selects.
How DB2 Path Checker interacts with DB2 SQL Performance Analyzer

When DB2 Path Checker finds SQL statements that have changed access paths, you can transfer the SQL statement text to DB2 SQL Performance Analyzer for further analysis. DB2 SQL Performance Analyzer calculates resource consumption, and then generates cost reports that help you to fine-tune the performance of the queries without having to run them in DB2.

When you run DB2 Path Checker, it writes SQL statements that have changed access paths to a data set that you specify in an ANLOUT DD statement. You can then use that data set as input to a DB2 SQL Performance Analyzer job by specifying it as the ddname for ANLIN. You can specify these ddnames in a different step within the same job or at any future time.

The data set that stores the SQL statements is the only requirement to use this feature. DB2 SQL Performance Analyzer processes the SQL statements that DB2 Path Checker selected and generates a report.

For more information, see “Sample job to send SQL statements to SQL Performance Analyzer” on page 170.

Related concepts:
“DB2 Path Checker features and benefits” on page 6
DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.

Related tasks:
“Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80
Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.
“Comparing access paths for different binds of the same program version” on page 78
Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

Service updates and support information

Service updates and support information for this product, including software fix packs, PTFs, Frequently Asked Question (FAQs), technical notes, troubleshooting information, and downloads, are available from the Web.

To find service updates and support information, see the following website:


DB2 Path Checker documentation and updates

This topic explains where to find DB2 Tools information on the web, and explains how to receive information updates automatically.
**DB2 Path Checker information on the web**

The DB2 Tools Library web page provides current product documentation that you can view, print, and download. To locate publications with the most up-to-date information, refer to the following web page:

[http://www.ibm.com/software/data/db2imstools/db2tools-library.html](http://www.ibm.com/software/data/db2imstools/db2tools-library.html)

IBM Redbooks® publications that cover DB2 and IMS™ Tools are available from the following Web page:


The Data Management Tools Solutions website shows how IBM solutions can help IT organizations maximize their investment in DB2 and IMS databases while staying ahead of today's top data management challenges:


**Receiving documentation updates automatically**

To automatically receive a weekly email that notifies you when new technote documents are released, when existing product documentation is updated, and when new product documentation is available, you can register with the IBM My Support service. You can customize the service so that you receive information about only those IBM products that you specify.

To register with the My Support service:

2. Enter your IBM ID and password, or create one by clicking register now.
3. When the My Support page is displayed, click add products to select those products that you want to receive information updates about. The DB2 and IMS Tools category is located under Software > Data and Information Management > Database Tools & Utilities.
4. Click Subscribe to email to specify the types of updates that you would like to receive.
5. Click Update to save your profile.

---

**Accessibility features**

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use a software product successfully.

The major accessibility features in DB2 Path Checker for z/OS enable users to do the following activities:

- Use assistive technologies such as screen readers and screen magnifier software. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.
- Customize display attributes such as color, contrast, and font size.
- Operate specific or equivalent features using only the keyboard. Refer to the following publications for information about accessing ISPF interfaces:
  - z/OS ISPF User’s Guide, Volume 1, SC34-4822
  - z/OS TSO/E Primer, SA22-7787
These guides describe how to use ISPF, including the use of keyboard shortcuts or function keys (PF keys), includes the default settings for the PF keys, and explains how to modify their functions.
Chapter 2. Customizing DB2 Path Checker by using Tools Customizer

Before you begin the customization process, review the steps that are summarized in this topic.

**Procedure**

1. Prepare to customize DB2 Path Checker.
   a. See “Preparing to customize DB2 Path Checker.”
   b. See “Set up your environment prior to customization” on page 23.
   c. See “Worksheets: Gathering required data set names” on page 25.
2. Become familiar with Tools Customizer. See Tools Customizer terminology and data sets.
3. Use Tools Customizer to customize DB2 Path Checker components. See Customizing DB2 Path Checker

**Important:**

You must create sql-id.PLAN_TABLE and sql-id.DSN_STATEMNT_TABLE before you run the installation verification program (IVP) that the customization process creates.

### Preparing to customize DB2 Path Checker

Before you start to customize DB2 Path Checker, determine all of the customization values that you need to specify during the customization process, and familiarize yourself with all of the customization tasks.

The following checklist lists and describes each significant customization step. Use this checklist to guide you through the entire customization process.

**Tip:** Print the following checklist and the data set names and parameter values worksheets. Use the worksheets to record your values, and refer to them during the customization process.

<table>
<thead>
<tr>
<th>Task</th>
<th>Link to detailed instructions</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools Customizer basics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prior to beginning the customization process, familiarize yourself with Tools Customizer terminology and data sets, and other basic information about Tools Customizer.</td>
<td>“Tools Customizer terminology and data sets” on page 139</td>
<td></td>
</tr>
</tbody>
</table>

Software requirements
<table>
<thead>
<tr>
<th>Task</th>
<th>Link to detailed instructions</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify that your environment meets the minimum software requirements.</td>
<td>Set up your environment prior to customization on page 23</td>
<td></td>
</tr>
<tr>
<td>• To install and use DB2 Path Checker, your environment must be running a supported version of the z/OS operating system and of DB2 for z/OS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• You must be familiar with the languages, components, formats, and program sizes that DB2 Path Checker supports.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• You must be familiar with the required authorizations to use DB2 Path Checker after customization.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• To launch DB2 Path Checker from the DB2 Administration Tool Launchpad, you must be using a supported version of DB2 Administration Tool.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• To interface with DB2 SQL Performance Analyzer, you must be using a supported version of this product.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• You must verify that DB2 Path Checker and Tools Customizer for z/OS have been installed successfully.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMP/E installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that DB2 Path Checker has been installed correctly. DB2 Path Checker is installed by using standard SMP/E processing.</td>
<td>Verify that DB2 Path Checker has been installed successfully on page 25</td>
<td></td>
</tr>
<tr>
<td>Verify that Tools Customizer for z/OS has been installed correctly. Tools Customizer for z/OS is installed by using standard SMP/E processing.</td>
<td>Verify that Tools Customizer has been installed successfully on page 25</td>
<td></td>
</tr>
<tr>
<td>Tables for DB2 EXPLAIN output</td>
<td>Data requirements</td>
<td></td>
</tr>
<tr>
<td>Understand the requirements for plan tables and DSN statement tables.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that the required EXPLAIN tables exist before running the IVP program</td>
<td>PLAN_TABLE</td>
<td></td>
</tr>
<tr>
<td>Verify that PLAN_TABLE and DSN_STATEMNT_TABLE exist before you run the installation verification program (IVP) that the customization process creates. If the tables do not exist, you must create them. Tools Customizer for z/OS does not create the tables automatically.</td>
<td>DSN_STATEMNT_TABLE</td>
<td></td>
</tr>
<tr>
<td>Tip: If you are migrating from a previous version of DB2, use member DSNTESC in the SDSNSAMP DB2 library to perform all required migration tasks. If you run all the migration jobs, then the PLAN_TABLE and DSN_STATEMNT_TABLE will be compatible with the DB2 version that you are running.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gather data set names</td>
<td>Worksheets: Gathering required data set names on page 25</td>
<td></td>
</tr>
<tr>
<td>During the customization process, you must specify names for the following types of data sets:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Tools Customizer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DB2 Path Checker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Link to detailed instructions</td>
<td>Status</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>## Task Link to detailed instructions Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gather parameter values</td>
<td>During the customization process, you must specify parameter values for DB2 Path Checker, for DB2, and (if required) for your LPAR.</td>
<td>“Worksheets: Gathering parameter values for Tools Customizer” on page 26</td>
</tr>
<tr>
<td><strong>Customize DB2 Path Checker</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start Tools Customizer by running a REXX EXEC from the ISPF Command Shell panel.</td>
<td>“Starting Tools Customizer” on page 35</td>
<td></td>
</tr>
<tr>
<td>Set up Tools Customizer user settings. If you are running Tools Customizer for the first time, you must modify several user settings to suit your environment. Otherwise, if the user settings that you have already established are still appropriate, skip this step.</td>
<td>“Modifying Tools Customizer user settings” on page 36</td>
<td></td>
</tr>
<tr>
<td>Complete the steps in the appropriate customization roadmap based on the type of customization that you are performing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Customizing DB2 Path Checker for the first time</strong></td>
<td>“Roadmap: Customizing DB2 Path Checker for the first time” on page 40</td>
<td></td>
</tr>
<tr>
<td>Follow this roadmap if you do not have a customized version of DB2 Path Checker and you need to customize it for the first time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Customizing a different version of DB2 Path Checker</strong></td>
<td>“Roadmap: Customizing a new version of DB2 Path Checker from a previous customization” on page 41</td>
<td></td>
</tr>
<tr>
<td>Follow this roadmap if you have already customized a version of DB2 Path Checker and you want to use the same parameter values to customize a different version.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Recustomizing DB2 Path Checker</strong></td>
<td>“Roadmap: Recustomizing DB2 Path Checker” on page 42</td>
<td></td>
</tr>
<tr>
<td>Follow this roadmap if you have a customized version of DB2 Path Checker but you want to change one or more parameter values.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Some customization options require you to manually complete additional tasks after you have used Tools Customizer. If you generated jobs in Tools Customizer that correspond to the following customization options, complete the additional tasks before you submit the jobs. In some cases, an optional task can be completed either by using Tools Customizer or by manually completing tasks without using Tools Customizer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Optional: Qualify tables for SQL EXPLAIN processing</strong></td>
<td>Qualifying tables for SQL Explain processing</td>
<td></td>
</tr>
<tr>
<td>Ensure that your tables are qualified correctly, including your plan tables and the target tables that your SQL statements process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Optional: Set up a history table</strong></td>
<td>History table</td>
<td></td>
</tr>
<tr>
<td>Create a history table to contain a summary of changes that occur during DB2 Path Checker processing.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Set up your environment prior to customization

Prior to beginning the customization process, ensure that your environment meets all requirements, that you have installed all prerequisite software, and that you have considered how you want to customize optional features.
Verify that your environment meets software requirements

Ensure that you are using z/OS V1.12 (5694-A01) or later.

Ensure that you are using one of the following supported versions of DB2 for z/OS:
- DB2 V9.1 (5635-DB2)
- DB2 Value Unit Edition V9.1 (5697-P12)
- DB2 V10 (5605-DB2)
- DB2 Value Unit Edition V10.1 (5697-P31)
- DB2 V11 (5615-DB2)
- DB2 Value Unit Edition V11.1 (5697-P43)

DB2 Path Checker supports:
- Applications in VS COBOL, COBOL II, PL/I, Assembler, and C
- Diagnostic information for DBRMs, DB2 catalog, and DB2 time stamp usage in packages
- Any valid DB2 plan table format
- Large programs (less than 10000 statements)

DB2 Path Checker requires the following authorizations:
- Read access to the DBRMLIB that contains the member to be checked.
- Access to the DB2 load library. DB2 Path Checker uses the DB2 call attachment facility.
- A primary or secondary authorization ID (AUTHID) that has sufficient authority to update the plan table.

The TEST and EXPLAIN commands require a primary or secondary authorization ID that has sufficient authority to issue a dynamic explain for the SQL in the DBRM. Typically, the authority that is required to run the SQL is sufficient.

DB2 Path Checker uses dynamic SQL for everything except the SET CURRENT SQLID command. The DYNAMICRULES (BIND) option of BIND allows public usage if it is bound by an AUTHID with sufficient authority.

- A plan table for the specified owner (AUTHID). A DSN_STATEMNT_TABLE is optional for the authorization ID for TEST and EXPLAIN processing and the information is added to the basic report (SYSPRINT data set) the explain report (SYSEXPLN data set).

To add DB2 Path Checker to the DB2 Administration Tool Launchpad, ensure that you are using one of the following supported versions of DB2 Administration Tool for z/OS:
- DB2 Administration Tool V10.1
- DB2 Administration Tool V10.2
- DB2 Administration Tool V11.1

To set up the interface between DB2 Path Checker and DB2 SQL Performance Analyzer, ensure that you are using one of the following supported versions of DB2 SQL Performance Analyzer:
- DB2 SQL Performance Analyzer V4.2
- DB2 SQL Performance Analyzer V4.1
Verify that DB2 Path Checker has been installed successfully

See the program directory for DB2 Path Checker for z/OS, GI10-8975 for installation instructions.

Verify that Tools Customizer has been installed successfully

Tools Customizer for z/OS is a component of IBM Tools Base for z/OS (5655-V93), which is available free of charge. Tools Customizer for z/OS provides a standard approach to customizing IBM DB2 for z/OS Tools.

See the program directory for IBM Tools Base for z/OS, GI10-8819 for installation instructions.

**Worksheets: Gathering required data set names**

Identify and record the data set names that will be used during the customization process and make sure that requirements for certain data sets are met.

**Tip:** Print the following worksheets and refer to them during the customization process.

**Data set names for Tools Customizer**

Identify and record the following Tools Customizer data set names:

<table>
<thead>
<tr>
<th>Data set name</th>
<th>Description</th>
<th>Special requirements</th>
<th>Your data set name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCCQEXEC</td>
<td>EXEC library for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td>SCCQDENU</td>
<td>Metadata library for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td>SCCQLOAD</td>
<td>Executable load module library for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td>SCCQMENU</td>
<td>ISPF messages for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td>SCCQPENU</td>
<td>ISPF panels for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td>SCCQSAMP</td>
<td>Sample members for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td>SCCQTENU</td>
<td>Table library for Tools Customizer</td>
<td>None.</td>
<td></td>
</tr>
</tbody>
</table>

**Data set names for DB2 Path Checker**

Identify and record the following DB2 Path Checker data set names.

<table>
<thead>
<tr>
<th>Data set name</th>
<th>Description</th>
<th>Special requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCKPCLST</td>
<td>CLIST library for DB2 Path Checker</td>
<td>You must have write access to this data set.</td>
</tr>
<tr>
<td>SCKPDBRM</td>
<td>DBRM library for DB2 Path Checker</td>
<td>None.</td>
</tr>
</tbody>
</table>
### Data set name | Description | Special requirements
--- | --- | ---
SCKPDENU | Metadata library for DB2 Path Checker | None.
SCKLOAD | Executable load module library for DB2 Path Checker | None.
SCKPMENU | ISPF messages for DB2 Path Checker | None.
SCKPPENU | ISPF panels for DB2 Path Checker | None.
SCKPSAMP | Sample members for DB2 Path Checker | You must have write access to this data set.
SCKPSKEL | ISPF JCL skeletons for DB2 Path Checker | None.

### Worksheets: Gathering parameter values for Tools Customizer

During the customization process, you provide parameter values for the product that you are customizing, for DB2, and (in some cases) for your LPAR.

Use the worksheets in this topic to record the appropriate parameter settings for your purposes, and then use these worksheets during the customization process. The worksheets are organized based on the order of the customization panels in Tools Customizer.

**Tip:** Print the following worksheets and refer to them during the customization process.

### Settings for Tools Customizer

**Description**

Use the following worksheet to identify and record the values for Tools Customizer settings. During the customization process, you will enter these values on the Tools Customizer Settings panel (CCQPSET).

For more information about the parameters in this section, see “Data sets that Tools Customizer uses during customization” on page 141.

#### Product Customization Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Sample or default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customization library qualifier</td>
<td>DB2TOOL.PRODUCT.CUST</td>
<td></td>
</tr>
<tr>
<td>Use DB2 group attach</td>
<td>blank</td>
<td></td>
</tr>
</tbody>
</table>
Tools Customizer Library Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Sample or default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata library</td>
<td>DB2TOOL.CCQ110.SCCQDENU</td>
<td></td>
</tr>
<tr>
<td>Discover output data set</td>
<td>DB2TOOL.CCQ110.DISCOVER</td>
<td></td>
</tr>
<tr>
<td>Data store data set</td>
<td>DB2TOOL.CCQ110.DATASTOR</td>
<td></td>
</tr>
</tbody>
</table>

User Job Card Settings for Customization Jobs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Sample or default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The job card information to be inserted into the generated jobs for customizing a product or component.</td>
<td>The job statement information from the ISPF Batch Selection panel.</td>
<td></td>
</tr>
</tbody>
</table>

Metadata library for DB2 Path Checker

Description
Use the following worksheet to identify and record the value of the metadata library for DB2 Path Checker. During the customization process, you will enter this value on the Specify the Metadata Library panel (CCQPHLQ).

DB2 Path Checker metadata library

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Sample or default value</th>
<th>Discovered?</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata library</td>
<td>hlq.SCKPDENU</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Customization values for the Discover EXEC

Complete this worksheet only if you are recustomizing a product that has previously been customized by using Tools Customizer.

Description
Use the following worksheet to identify and record the customization values for the Tools Customizer Discover EXEC. During the customization process, you will enter these values on the Discover Customized Product Information panel (CCQPDSC).
## Discover EXEC for Extracting Information from an Already Customized Product

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Sample or default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover EXEC library</td>
<td>hlq.SCKPCLST</td>
<td></td>
</tr>
<tr>
<td>Discover EXEC name</td>
<td>CKPCDISC</td>
<td></td>
</tr>
<tr>
<td>Discover output data set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information for Discover EXEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path Checker BIND Location</td>
<td>SYSX.CKP420.SCKPSAMP(CKPBIND)</td>
<td></td>
</tr>
<tr>
<td>Path Checker IVP Location</td>
<td>SYSX.CKP420.SCKPSAMP(CKPIVP)</td>
<td></td>
</tr>
</tbody>
</table>

### Product to Customize section

**Description**

The parameters that are listed in the Product to Customize section are read-only. They contain information that was provided on other panels, by Tools Customizer, or by the DB2 Path Checker metadata data set.
### Read-only fields in the Product to Customize section on Tools Customizer panels

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Discovered?</th>
<th>Source of this value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product metadata library</td>
<td>No</td>
<td>This value is specified on the Specify the Metadata Library panel (CCQPHLQ).</td>
</tr>
<tr>
<td>LPAR</td>
<td>No</td>
<td>This value is provided by Tools Customizer.</td>
</tr>
<tr>
<td>Product name</td>
<td>No</td>
<td>This value is provided by the product metadata file.</td>
</tr>
<tr>
<td>Version</td>
<td>No</td>
<td>This value is provided by the product metadata file.</td>
</tr>
<tr>
<td>Product customization library</td>
<td>No</td>
<td>This value is derived from the user-specified customization library qualifier on the Tools Customizer Settings panel (CCQPSET).</td>
</tr>
</tbody>
</table>

### Required parameters section

**Description**
The parameters in this task are required for all customizations. During the customization process, you will enter these values on the Product Parameters panel (CCQPPRD).

**Parameters in the Required parameters section on the Product Parameters panel (CCQPPRD)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLQ of DB2 Path Checker libraries</td>
<td>Yes</td>
<td>Yes</td>
<td>*hlq.*CKP420</td>
<td></td>
</tr>
</tbody>
</table>
Task: Path Checker Customization

Description
Path Checker is a DB2 application that uses SQL. Path Checker must be bound at the subsystem where you intend to use it. This task creates the BIND job.

This task is required.

Jobs generated
The steps in this task generate the following customization job:

- CKPBNDab, where ab are alphanumeric characters assigned by Tools Customizer. This job is based on the CKPBND31 template and is in the A0BND3AA member.

Required authority
The following list indicates the job and the type of authority that is required to run the job:

- CKPBNDab job
  The user ID must have DB2 system administrator authority or valid RACF® or equivalent access to the group to use the customization ID.

Steps and parameters for the Path Checker customization task

<table>
<thead>
<tr>
<th>Step or parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Path Checker BIND job</td>
<td>Yes</td>
<td>No</td>
<td>Selected</td>
<td></td>
</tr>
<tr>
<td>Create the bind job that will BIND the Path Checker packages and plans for a specified subsystem.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task: Invocation EXEC for DB2 Path Checker

Description
Create the TSO invocation EXEC with the name CKP for Path Checker.

This task is required.

Jobs generated
This task generates the following jobs:

- CKPTSOab, where ab are alphanumeric characters assigned by Tools Customizer. This job is based on the CKPTSO33 template and is in the A1TSO33 member.
- CKPCPCab, where ab are alphanumeric characters assigned by Tools Customizer. This job is based on the CKPCPCKP template and is in the A2CPCKP member.

Required authority
The following list indicates the job and the type of authority that is required to run the job:

- CKPTSOab job
  The user ID must have write access to the SCKPCLST data set.

- CKPCPCab job
  The user ID must have write access to the SYSPROC dataset.

Steps and parameters for the Invocation EXEC for DB2 Path Checker task

<table>
<thead>
<tr>
<th>Step or parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the invocation EXEC for DB2 Path Checker</td>
<td>Yes</td>
<td>No</td>
<td>Selected</td>
<td></td>
</tr>
<tr>
<td>Create the CKP EXEC for DB2 Path Checker. CKP EXEC invokes the ISPF interface for DB2 Path Checker.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Steps and parameters for the Invocation EXEC for DB2 Path Checker task

<table>
<thead>
<tr>
<th>Step or parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 load library</td>
<td>Yes</td>
<td>No</td>
<td>DSN.SDSNLOAD</td>
<td></td>
</tr>
<tr>
<td>Copy CKP EXEC to SYSPROC library</td>
<td>No</td>
<td>No</td>
<td>Not selected</td>
<td></td>
</tr>
<tr>
<td>SYSPROC library for CKP</td>
<td>Yes</td>
<td>No</td>
<td>SYSX.SYSPROC</td>
<td></td>
</tr>
</tbody>
</table>

Task: Path Checker IVP jobs

Description
These steps create the BIND job to bind the CKPTDBRM package and create a job that invokes the CKPPTHCK program. This process verifies that DB2 Path Checker is installed correctly.

This task is optional.

Jobs generated
This task generates the following jobs:
- CKPIBN\textit{ab}, where \textit{ab} are alphanumeric characters assigned by Tools Customizer. This job is based on the CKPIBND template and is in the A3IBNDAA member.
- CKPIRU\textit{ab}, where \textit{ab} are alphanumeric characters assigned by Tools Customizer. This job is based on the CKPIRUN template and is in the A4IRUNAA member.

Required authority
The following list indicates the type of authority that is required to run the job:

\textbf{CKPIBN\textit{ab} job}
The user ID must have DB2 system administrator authority or valid RACF or equivalent access to the group to use the customization ID.

\textbf{CKPIRU\textit{ab} job}
The user ID must have DB2 system administrator authority or valid RACF or equivalent access to the group to use the customization ID.

Steps and parameters for the Path Checker IVP jobs task

<table>
<thead>
<tr>
<th>Step or parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the IVP BIND job</td>
<td>Yes</td>
<td>No</td>
<td>Selected</td>
<td></td>
</tr>
</tbody>
</table>
### Steps and parameters for the Path Checker IVP jobs task

<table>
<thead>
<tr>
<th>Step or parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create IVP invocation batch job</td>
<td>No</td>
<td>No</td>
<td>Not selected</td>
<td></td>
</tr>
<tr>
<td>Create the Path Checker IVP job that</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>invokes the CKPPTHCK program</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and runs basic Path Checker commands</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This will verify whether Path Checker</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>has been installed correctly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Task: Customize the DB2 Administration Tool Launchpad for Path Checker

**Description**

Create the job to customize the launchpad for the DB2 Administration Tool.

This task is optional.

**Jobs generated**

The step in this task generates the following customization job:

- CKPLCHab, where ab are alphanumeric characters assigned by Tools Customizer. This job is based on the CKPLCHPD template and is in the A5LCHPD member.

**Required authority**

The following list indicates the job and the type of authority that is required to run the job:

- **CKPLCHab job**
  
  The user ID must have write access to the data sets for DB2 Administration Tool and DB2 Path Checker.

### Steps and parameters for the Customize the DB2 Administration Tool Launchpad for Path Checker task

<table>
<thead>
<tr>
<th>Step or parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Path Checker to DB2 Administration</td>
<td>Yes</td>
<td>No</td>
<td>Selected</td>
<td></td>
</tr>
<tr>
<td>Tool Launchpad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create the job that adds the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>invocation command for DB2 Path Checker</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to the launchpad of the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2 Administration Tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| HLQ of the DB2 Admin libraries           | Yes       | No          | DB2TOOLS.DB2ADMIN   |              |
| Specify the high-level qualifier (HLQ)   |           |             |                     |              |
| of the libraries for the DB2 Administration Tool. Path Checker uses these libraries to update the invocation table for the launchpad of the DB2 Administration Tool. | | | | |
**LPAR Parameters section**

**Description**
This section contains LPAR parameters. They are not required unless you selected the optional task "Customize the DB2 Administration Tool Launchpad for Path Checker" on the Product Parameters panel. If you selected this task, ISPF libraries are required. During the customization process, you will enter these values on the LPAR Parameters panel (CCQPLPR).

**ISPF Libraries**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message library</td>
<td>Yes</td>
<td>No</td>
<td>hlq.SISPMENU</td>
<td></td>
</tr>
<tr>
<td>Panel library</td>
<td>Yes</td>
<td>No</td>
<td>hlq.SISPPENU</td>
<td></td>
</tr>
<tr>
<td>Skeleton library</td>
<td>Yes</td>
<td>No</td>
<td>hlq.SISPSENU</td>
<td></td>
</tr>
<tr>
<td>ISPF table input library</td>
<td>Yes</td>
<td>No</td>
<td>hlq.SISPTENU</td>
<td></td>
</tr>
<tr>
<td>JCL procedure library</td>
<td>Yes</td>
<td>No</td>
<td>SYS1.PROCLIB</td>
<td></td>
</tr>
</tbody>
</table>

**DB2 Parameters section**

**Description**
This section contains DB2 parameters. All parameters are required. During the customization process, you will enter these values on the DB2 Parameters panel (CCQPDB2).
### General DB2 Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>NFM</td>
<td></td>
</tr>
</tbody>
</table>
| The mode in which the DB2 subsystem is running. The following values are valid:  
  - CM is compatibility mode on all listed DB2 versions except DB2 10.  
  - CM8 is conversion mode from DB2 V8 on DB2 10.  
  - CM9 is conversion mode from DB2 Version 9.1 on DB2 10.  
  - NFM is new-function mode on all listed DB2 versions. |

| **Level number** | Yes | Yes | blank |
| The version, release, and modification level of the DB2 subsystem. The following values are valid:  
  - 910 is valid only for CM or NFM.  
  - 101 is valid only for CM8, CM9 or NFM.  
  - 111 is valid only for CM or NFM. |

### DB2 Libraries

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Load library</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>DSN.SDSNLOAD</td>
<td></td>
</tr>
<tr>
<td>The data set name of the DB2 load library. You can specify multiple values for this parameter.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Path Checker BIND parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OWNER option</strong></td>
<td>Yes</td>
<td>No</td>
<td>SYSADM</td>
<td></td>
</tr>
</tbody>
</table>
| This is the DB2 AUTHID that will own the DB2 Path Checker plan.  
This value is specified in the OWNER option of the BIND command that is used to bind the DB2 Path Checker plan. |
## Path Checker BIND parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUALIFIER option</td>
<td>Yes</td>
<td>No</td>
<td>SYSADM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is the qualifier name that will be used for unqualified tables referenced by the DB2 Path Checker plan. This value is specified in the QUALIFIER option of the BIND command that is used to bind the DB2 Path Checker plan. It is also used as the QUALIFIER option value in the BIND command that binds the test packages for the IVP job.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DYNAMICRULES option</td>
<td>Yes</td>
<td>No</td>
<td>RUN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is the value of the DB2 Bind option DYNAMICRULES. Valid values for DB2 Path Checker are RUN and BIND. DYNAMICRULES determines the authorization to be used when running dynamic SQL. DB2 Path Checker users must have the authority to run dynamic SQL.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Path Checker IVP Job parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Discovered?</th>
<th>Default value</th>
<th>Your value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 Auth ID used when running the IVP</td>
<td>Yes</td>
<td>Yes</td>
<td>SYSADM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The DB2 Authorization ID that is used when running the IVP job. During the IVP, the AUTHID is used in a SET CURRENT SQLID statement. The AUTHID must have SELECT authority on the catalog, EXECUTE authority to the DB2 Path Checker plan, and authority to run dynamic SQL.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Starting and preparing Tools Customizer for use**

Use the provided REXX EXEC to start Tools Customizer. The first time that you use Tools Customizer, you must modify the settings that Tools Customizer uses to customize DB2 Path Checker.

**Starting Tools Customizer**

Start Tools Customizer by running a REXX EXEC from the ISPF Command Shell panel.

**Before you begin**

Tools Customizer must be SMP/E installed. You must know the high-level qualifier of where the Tools Customizer libraries reside. The high-level qualifier is considered to be all the segments of the data set name except the lowest-level qualifier, which is SCCQEXEC.
**Attention:** Ensure that Tools Customizer load libraries are not APF authorized. APF authorizing Tools Customizer libraries results in an abend.

**About this task**

To run the REXX EXEC, you must either change the placeholder in the EXEC for the high-level qualifier of the Tools Customizer EXEC library or pass the high-level qualifier as a parameter when you run the EXEC. The REXX EXEC is in the CCQTCZ member of the EXEC library.

**Procedure**

1. Optional: Change the placeholder for the high-level qualifier in the REXX EXEC:
   a. Find the EXEC library data set for Tools Customizer. The name of the data set is `high_level_qualifier.SCCQEXEC`.
   b. Edit data set member CCQTCZ and replace the `<TCZ HLQ>` string with the high-level qualifier of the EXEC library data set. For example, if the name of the Tools Customizer EXEC library is `CCQTCZ.USABSAND.SCCQEXEC`, replace `<TCZ HLQ>` with `CCQTCZ.USABSAND`.

   You have to change the placeholder for the high-level qualifier only once. When you run the REXX EXEC, you do not have to pass the high-level qualifier as a parameter.

2. Run the REXX EXEC (CCQTCZ):
   a. From the ISPF Primary Option Menu, select option 6. The ISPF Command Shell panel is displayed.
   b. Specify the EX command to run the REXX EXEC. For example, if the Tools Customizer EXEC library is `CCQTCZ.USABSAND.SCCQEXEC` and you changed the placeholder for the high-level qualifier in the REXX EXEC, specify: `EX 'CCQTCZ.USABSAND.SCCQEXEC(CCQTCZ)'`

   If you did not change the placeholder for the high-level qualifier in the REXX EXEC, specify: `EX 'CCQTCZ.USABSAND.SCCQEXEC(CCQTCZ)'`

   `CCQTCZ.USABSAND`

**Results**

The IBM Customizer Tools for z/OS main menu panel is displayed.

**What to do next**

If you are running Tools Customizer for the first time, you must modify the Tools Customizer user settings. If you have already set the Tools Customizer user settings, either customize or recustomize DB2 Path Checker.

**Modifying Tools Customizer user settings**

Before you can customize DB2 Path Checker with Tools Customizer, you must review the settings that Tools Customizer uses. You might have to change the default values to suit your environment. In most cases, you can change the Tools Customizer values at any time. For example, after you have customized DB2 Path Checker and are customizing a different product or solution pack, you might have to change the settings.
Procedure

1. On the IBM Tools Customizer for z/OS main panel (CCQPHME), specify option 0, **User settings for Tools Customizer**. The Tools Customizer Settings panel (CCQPSET) is displayed, as shown in the following figure:

   ![Tools Customizer Settings panel (CCQPSET)](image)

   **Figure 2. The Tools Customizer Settings panel (CCQPSET)**

2. Review the values for the following required fields. Use the default value or specify your own value. You must have appropriate read and write access to the data sets that are specified.

   **Customization library qualifier**
   
   The high-level qualifier that is used as the prefix for the customization library. The customization library is a data set in which the generated jobs to customize DB2 Path Checker are stored. Write access to this qualifier is required.

   For each product to be customized, the first value that is specified for the qualifier is always used, even if you change it after you have generated the customization jobs. For example, if you customize a product and then specify a new qualifier for recustomization, although the new qualifier is saved and displayed, the original value is used.

   To maintain multiple instances of Tools Customizer, specify a unique customization library qualifier for each instance of Tools Customizer. Data set names that exceed 42 characters must be enclosed in single quotation marks (').

   **Use DB2 group attach**
   
   DB2 Path Checker does not support DB2 group attach names. You must specify NO in the **Use DB2 group attach** field.

   **Tools Customizer metadata library**
   
   The name of the data set that contains the metadata that is used to display the DB2 parameters. The parameters that are displayed on the DB2 Parameters panel depend on the parameters that you define and the tasks and steps that you select on the Product Parameters panel for the product that you are customizing. For example, the DB2 parameters that are required, based on the selected tasks and steps, are displayed on the DB2 Parameters panel, and you can edit them. If they are not
required, they are not displayed. Read access to this data set is required. Data set names that exceed 42 characters must be enclosed in single quotation marks (').

**Discover output data set**

The name of the data set in which the output from the DB2 Path Checker Discover EXEC is stored. Each product has its own Discover EXEC. The Discover EXEC retrieves the product and DB2 parameters from a previously customized product. Write access to this data set is required. Data set names that exceed 42 characters must be enclosed in single quotation marks (').

**Data store data set**

The name of the data set where Tools Customizer stores information about product and DB2 parameter values. Information about which products are associated with which DB2 entries (DB2 subsystems, DB2 group attach names, and DB2 data sharing members) is also stored in this data set. Data set names that exceed 42 characters must be enclosed in single quotation marks ('). The specified data store data set can be used with only one invocation of Tools Customizer at a time. Data set names that exceed 42 characters must be enclosed in single quotation marks (').

**User job card settings for customization jobs**

The job card information to be inserted into the generated jobs for customizing a product. The default value is the job statement information from the ISPF Batch Selection panel.

The first line of the job card automatically begins with the following information:

```
// JOB
```

where characters 3 - 10 are reserved by Tools Customizer for the job name and includes a blank space after JOB. This name cannot be edited. Information that you specify on the first line of the job card cannot exceed 57 characters. This character limit includes a continuation character. All other lines of the job card cannot exceed 72 characters.

3. Press End to save and exit. If the Discover output data set and the data store data set that you specified do not exist, Tools Customizer creates them.

**Important:** If the ISPF sessions unexpectedly ends before you exit Tools Customizer, the fields on the Tools Customizer Settings panel (CCQPSET) will be repopulated with default values, and you will be required to review them or specify new values again.

**Results**

The values are saved, and the IBM Tools Customizer for z/OS main menu panel (CCQPHME) is displayed again.

**What to do next**

You are ready to customize or recustomize DB2 Path Checker or to change parameter settings.

**Related concepts:**


Using Tools Customizer to customize DB2 Path Checker consists of identifying the product to customize; defining any required DB2 Path Checker and DB2 parameters; generating the customization jobs; and submitting the jobs.

Hiding and displaying panel text

After you are familiar with Tools Customizer, you might want to hide the instructions and some of the basic descriptions that are displayed by default on Tools Customizer panels.

About this task

By using the OPTIONS command, you can choose to show or hide the following information on Tools Customizer panels:

- The instructions on all panels
- The Product to Customize section on the Customizer Workplace panel (CCQPWRK)
- The Usage Notes section on the Product Parameters panel (CCQPPRD), the LPAR Parameters panel (CCQPLPR), and the DB2 Parameters panel (CCQPDB2).

By hiding this information, more data can be displayed on the panels. Later, you can redisplay this information also by using the OPTIONS command.

Procedure

1. On any Tools Customizer panel, issue the OPTIONS command. The Panel Display Options panel (CCQPOPT) is displayed, as shown in the following figure. By default, all options are preselected with a slash (/) to be shown.

   ![Figure 3. The Panel Display Options panel (CCQPOPT)](image)

2. To hide any of the options, remove the slash, and press Enter.

Customizing DB2 Path Checker

Using Tools Customizer to customize DB2 Path Checker consists of identifying the product to customize; defining any required DB2 Path Checker and DB2 parameters; generating the customization jobs; and submitting the jobs.

Customization roadmaps describe the steps that you must complete to customize DB2 Path Checker. Separate roadmaps are provided for the three most common types of customizations.

Use the following table to determine which roadmap corresponds to your environment.
Table 1. Customization roadmaps

<table>
<thead>
<tr>
<th>Environment description</th>
<th>Roadmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>You do not have a customized version of DB2 Path Checker, and you need to customize it for the first time.</td>
<td>“Roadmap: Customizing DB2 Path Checker for the first time”</td>
</tr>
<tr>
<td>You have already customized a version of DB2 Path Checker, and you want to use the same parameter values to customize a different version.</td>
<td>“Roadmap: Customizing a new version of DB2 Path Checker from a previous customization” on page 41</td>
</tr>
<tr>
<td>You have a customized version of of DB2 Path Checker, but you want to change one or more parameter values.</td>
<td>“Roadmap: Recustomizing DB2 Path Checker” on page 42</td>
</tr>
</tbody>
</table>

Roadmap: Customizing DB2 Path Checker for the first time

This roadmap lists and describes the steps that are required to customize DB2 Path Checker for the first time.

If you are customizing a previous version of DB2 Path Checker, see “Roadmap: Customizing a new version of DB2 Path Checker from a previous customization” on page 41.

Before you complete these steps, ensure that the following prerequisites have been met:

- All of the product customization steps that must be done before Tools Customizer is started are complete.

- Tools Customizer is started.

- The Tools Customizer settings have been reviewed or modified, and saved.

Complete the steps in the following table to customize DB2 Path Checker for the first time.

Table 2. Steps for customizing DB2 Path Checker for the first time

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specify the product metadata library for the product that you want to customize. The name of this library is hlq.SCKPDYXIMZDENU.</td>
<td>“Specifying the metadata library for the product to customize” on page 43</td>
</tr>
<tr>
<td>2</td>
<td>Create new DB2 entries and associate them with DB2 Path Checker.</td>
<td>“Creating and associating DB2 entries” on page 47</td>
</tr>
<tr>
<td>3</td>
<td>Define the required parameters.</td>
<td>“Defining parameters” on page 49</td>
</tr>
<tr>
<td>4</td>
<td>Generate the customization jobs for the product or for the DB2 entries on which DB2 Path Checker is ready to be customized.</td>
<td>“Generating customization jobs” on page 54</td>
</tr>
<tr>
<td>5</td>
<td>Submit the generated customization jobs.</td>
<td>“Submitting customization jobs” on page 55</td>
</tr>
</tbody>
</table>

The following table lists some of the common administrative tasks that you might need to do during the customization process.
Table 3. Administrative tasks

<table>
<thead>
<tr>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse the different types of parameters.</td>
<td>“Browsing parameters” on page 57</td>
</tr>
<tr>
<td>Copy an existing DB2 entry to the list of DB2 entries on which DB2 Path Checker can be customized.</td>
<td>“Copying DB2 entries” on page 58</td>
</tr>
<tr>
<td>Remove one or more DB2 entries from the associated list.</td>
<td>“Removing DB2 entries” on page 59</td>
</tr>
<tr>
<td>Delete one or more DB2 entries from the master list.</td>
<td>“Deleting DB2 entries” on page 60</td>
</tr>
<tr>
<td>Display a list of customization jobs that have been previously generated.</td>
<td>“Displaying customization jobs” on page 60</td>
</tr>
<tr>
<td>Maintain the customization jobs in the customization library.</td>
<td>“Maintaining customization jobs” on page 61</td>
</tr>
</tbody>
</table>

Roadmap: Customizing a new version of DB2 Path Checker from a previous customization

This roadmap lists and describes the steps for customizing a new version of DB2 Path Checker based on the existing customization values of a previous version of the same product.

Use this roadmap even if the previous version of DB2 Path Checker was not customized by using Tools Customizer.

Before you complete these steps, ensure that the following prerequisites have been met:

- All of the product customization steps that must be done before Tools Customizer is started are complete.
- Tools Customizer is started.
- The Tools Customizer settings have been reviewed or modified, and saved.

Complete the steps in the following table to customize a new version of DB2 Path Checker from a previous customization.

Table 4. Steps for customizing a new version of DB2 Path Checker from a previous customization

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specify the product metadata library for the product that you want to customize. The name of this library is hlq.SCKPDXIMZDENU.</td>
<td>“Specifying the metadata library for the product to customize” on page 43</td>
</tr>
<tr>
<td>2</td>
<td>Use the DB2 Path Checker Discover EXEC to discover information about the version of DB2 Path Checker that you previously customized manually.</td>
<td>“Discovering DB2 Path Checker information automatically” on page 45</td>
</tr>
<tr>
<td>3</td>
<td>Define the required parameters.</td>
<td>“Defining parameters” on page 49</td>
</tr>
<tr>
<td>4</td>
<td>Generate the customization jobs for the product or for the DB2 entries on which DB2 Path Checker is ready to be customized.</td>
<td>“Generating customization jobs” on page 54</td>
</tr>
</tbody>
</table>
Table 4. Steps for customizing a new version of DB2 Path Checker from a previous customization (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Submit the generated customization jobs.</td>
<td>“Submitting customization jobs” on page 55</td>
</tr>
</tbody>
</table>

The following table lists some of the common administrative tasks that you might need to do during the customization process.

Table 5. Administrative tasks

<table>
<thead>
<tr>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse the different types of parameters.</td>
<td>“Browsing parameters” on page 57</td>
</tr>
<tr>
<td>Copy an existing DB2 entry to the list of DB2 entries on which DB2 Path Checker can be customized.</td>
<td>“Copying DB2 entries” on page 58</td>
</tr>
<tr>
<td>Remove one or more DB2 entries from the associated list.</td>
<td>“Removing DB2 entries” on page 59</td>
</tr>
<tr>
<td>Delete one or more DB2 entries from the master list.</td>
<td>“Deleting DB2 entries” on page 60</td>
</tr>
<tr>
<td>Display a list of customization jobs that have been previously generated.</td>
<td>“Displaying customization jobs” on page 60</td>
</tr>
<tr>
<td>Maintain the customization jobs in the customization library.</td>
<td>“Maintaining customization jobs” on page 61</td>
</tr>
</tbody>
</table>

Roadmap: Recustomizing DB2 Path Checker

This roadmap lists and describes the steps to change parameter values and regenerate customization jobs for DB2 Path Checker after you have customized it for the first time.

The new customization jobs will replace the customization jobs that were previously generated and stored in the customization library. Part of the recustomization process includes selecting or deselecting optional tasks or steps, changing the definitions of parameters that have already been defined, or both. Use the method in this roadmap instead of deleting customization jobs from the customization library.

Before you complete these steps, ensure that the following prerequisites have been met:
- All of the product customization steps that must be done before Tools Customizer is started are complete.
- Tools Customizer is started.

Complete the steps in the following table to recustomize DB2 Path Checker.

Table 6. Required steps for recustomizing DB2 Path Checker

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specify the product metadata library for the product that you want to recustomize. The name of this library is hlq.SCKPDYXIMZDENU.</td>
<td>“Specifying the metadata library for the product to customize” on page 43</td>
</tr>
</tbody>
</table>
Table 6. Required steps for recustomizing DB2 Path Checker (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Edit the specific tasks, steps, or parameters that need to be changed.</td>
<td>• &quot;Defining DB2 Path Checker parameters&quot; on page 49&lt;br&gt;• &quot;Defining DB2 parameters&quot; on page 51</td>
</tr>
<tr>
<td>3</td>
<td>Generate the customization jobs for the product or for the DB2 entries on which DB2 Path Checker is ready to be customized.</td>
<td>&quot;Generating customization jobs&quot; on page 54</td>
</tr>
<tr>
<td>4</td>
<td>Submit the new generated customization jobs.</td>
<td>&quot;Submitting customization jobs&quot; on page 55</td>
</tr>
</tbody>
</table>

The following table lists some of the common administrative tasks that you might need to do during the customization process.

Table 7. Administrative tasks

<table>
<thead>
<tr>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse the different types of parameters.</td>
<td>&quot;Browsing parameters&quot; on page 57</td>
</tr>
<tr>
<td>Copy an existing DB2 entry to the list of DB2 entries on which DB2 Path Checker can be customized.</td>
<td>&quot;Copying DB2 entries&quot; on page 58</td>
</tr>
<tr>
<td>Remove one or more DB2 entries from the associated list.</td>
<td>&quot;Removing DB2 entries&quot; on page 59</td>
</tr>
<tr>
<td>Delete one or more DB2 entries from the master list.</td>
<td>&quot;Deleting DB2 entries&quot; on page 60</td>
</tr>
<tr>
<td>Display a list of customization jobs that have been previously generated.</td>
<td>&quot;Displaying customization jobs&quot; on page 60</td>
</tr>
<tr>
<td>Maintain the customization jobs in the customization library.</td>
<td>&quot;Maintaining customization jobs&quot; on page 61</td>
</tr>
</tbody>
</table>

Specifying the metadata library for the product to customize

You must specify a metadata library for the product that you want to customize.

About this task

The product metadata library contains the information that determines which tasks, steps, and parameters are required to customize DB2 Path Checker. This information controls what is displayed on the Product Parameters panel and the DB2 Parameters panel.

After DB2 Path Checker has been SMP/E installed, the default name of the product metadata library is `high_level_qualifier.SCKPDYXIMZDENU`, where `high_level_qualifier` is all of the segments of the data set name except the lowest-level qualifier.

Procedure

1. Specify option 1 on the Tools Customizer for z/OS panel. The Specify the Metadata Library panel is displayed. This panel contains a list of the product metadata libraries that you specified most recently. If you are using Tools Customizer for the first time, this list is empty, as shown in the following
Use one of the following methods to specify the product metadata library:

- Type the name of a fully qualified partitioned data set (PDS) or an extended partitioned data set (PDSE) in the Metadata library field. Double quotation marks (""") cannot be used around the name. Single quotation marks (') can be used but are not required. If you are customizing DB2 Path Checker for the first time, you must use this method.

- Place the cursor on the library name in the Recent Metadata Libraries list, and press Enter.

The default name of the metadata library after the pack or product has been SMP/E installed is <hlq>.SxxxxDENU, where <hlq> is the high-level qualifier for the pack or the product, and xxx is the 3-character prefix for the pack or the product.

Metadata library: DB2TOOL.PTF420.SCKPDENU

Previously Used Metadata Library:
=>
=>

Metadata library: DYX.DYX150.SDYXDENU

Previously Used Metadata Library:
=>
=>

Figure 4. The Specify the Metadata Library panel

Results

If you are customizing DB2 Path Checker for the first time, the Run Discover EXEC panel is displayed. Otherwise, the Customizer Workplace panel is displayed.

What to do next

- Complete the steps that correspond to your environment:
  
  **Customizing DB2 Path Checker for the first time**
  Do not run the DB2 Path Checker Discover EXEC. Press End. The Customizer Workplace panel is displayed. If your environment requires
associated DB2 entries, ensure that they are created and associated. If your environment does not require associated DB2 entries, skip this step, and edit DB2 Path Checker parameters.

**Customizing DB2 Path Checker from a previous or current customization**

Press Enter to run the DB2 Path Checker Discover EXEC. The Discover Customized Product Information panel is displayed. Specify the required information for running the EXEC.

**Discovering DB2 Path Checker information automatically**

You can use the DB2 Path Checker Discover EXEC to discover information from a previous or current customization of DB2 Path Checker.

**About this task**

**Tip:** Using the DB2 Path Checker Discover EXEC to discover information from a previous or current customization saves time and reduces errors that can occur when parameters are specified manually.

DB2 Path Checker provides the Discover EXEC that you will run. Therefore, the information that can be discovered depends on DB2 Path Checker.

Parameter values that are discovered and parameter values that are specified manually are saved in the data store. If parameter values for the product that you want to customize exist in the data store, Tools Customizer issues a warning before existing values are replaced.

**Procedure**

1. On the Customizer Workplace panel, issue the DISCOVER command. If you chose to run the DB2 Path Checker Discover EXEC on the pop-up panel after you specified the product to customize, skip this step.

   **Tip:** You can run any Tools Customizer primary command by using either of the following methods:
   - Place the cursor on the name of the primary command, and press Enter.
   - Type the primary command name in the command line, and press Enter.

   The Discover Customized Product Information panel is displayed, as shown in the following figure:
2. Either accept the default values for the following input fields that Tools Customizer generates, or replace the default values with your own values:

   **Discover EXEC library**
   - The fully qualified data set name that contains the DB2 Path Checker Discover EXEC.

   **Discover EXEC name**
   - The name of the DB2 Path Checker Discover EXEC.

   **Discover output data set**
   - The fully qualified data set where output from the DB2 Path Checker Discover EXEC is stored.

3. Either accept or change the default values in the **Information for Discover EXEC** fields. These fields are generated by DB2 Path Checker. They show the information that is required to run the DB2 Path Checker Discover EXEC.

4. Issue the RUN command to run the DB2 Path Checker Discover EXEC. Alternatively, save your information without running the DB2 Path Checker Discover EXEC by issuing the SAVE command. If you issue the RUN command to run the DB2 Path Checker Discover EXEC, the parameter information is discovered for DB2 Path Checker, and the Customizer Workplace panel is displayed.

**Results**

The discovered parameter values for DB2 Path Checker replace any existing values.

**What to do next**

The next step depends on your environment:

- If DB2 entries were not discovered, or if you need to customize DB2 Path Checker on new DB2 entries, create and associate the entries.
If DB2 entries were discovered and you want to customize DB2 Path Checker on only these entries, define the parameters.

Related tasks:
- “Creating and associating DB2 entries”
  You can create new DB2 entries and associate them with DB2 Path Checker.
- “Defining parameters” on page 49
  To customize DB2 Path Checker, you must define DB2 Path Checker parameters and DB2 parameters, if your customization requires DB2 entries.

Creating and associating DB2 entries
You can create new DB2 entries and associate them with DB2 Path Checker.

About this task
The list of associated DB2 entries is on the Customizer Workplace panel.

Procedure
1. Issue the ASSOCIATE command on the Customizer Workplace panel. The Associate DB2 Entry for Product panel is displayed, as shown in the following figure:

```
CCQPDAD  Associate DB2 Entry for Product  10:07:28
Command ===>   Scroll ===> CSR

Select any of the following DB2 entries to add them to the Customizer Workplace panel. You use the Customizer Workplace panel to choose the DB2 subsystems, data sharing members, and group attach names on which to customize the product.

Commands: CREATE - Create a new DB2 entry

Product to Customize
Product metadata library : DB2TOOL.PTF420.SCKPDENU > LPAR . . : 3090
Product name . . . . . : DB2 Path Checker
Product version . . . . : 4.2.0

Line commands: A - Associate  C - Copy
Cmd SSID GrpAttch
------------------------------ End of DB2 entries -----------------------------
```

```
CCQPDAD  Associate DB2 Entry for Product  10:07:28
Command ===>   Scroll ===> CSR

Select any of the following DB2 entries to add them to the Customizer Workplace panel. You use the Customizer Workplace panel to choose the DB2 subsystems, data sharing members, and group attach names on which to customize the product.

Commands: CREATE - Create a new DB2 entry

Product to Customize
Product metadata library : DYX.DYX150.SDYXDENU > LPAR . . : 3090
Product name . . . . . : Autonomics Director for DB2
Product version . . . . : 1.5.0

Line commands: A - Associate  C - Copy
Cmd SSID GrpAttch
------------------------------ End of DB2 entries -----------------------------
```

Figure 6. The Associate DB2 Entry for Product panel
2. Create DB2 entries. If you need to associate DB2 entries that are already in the master list, skip this step and go to step 3.
   a. Issue the `CREATE` command. The Create DB2 Entries panel is displayed, as shown in the following figure:

   ![Create a DB2 Entry panel](image)

   **Figure 7. The Create a DB2 Entry panel**

   b. In the appropriate columns, specify a DB2 subsystem ID or DB2 data sharing member name for the DB2 entry that you want to create, and press Enter. Valid values are 1 - 4 characters. You can use symbolic characters. You cannot use blanks.

   **Tips:**
   - To insert multiple DB2 entries, specify the `Inn` line command, where `nn` is the number of DB2 entries to be inserted.
   - You will define specific parameters for these new DB2 entries on the DB2 Parameters panel. This panel is displayed after you select these new DB2 entries and issue the line command to generate the jobs, after you issue the primary command to generate the jobs for all associated DB2 entries, or when you manually edit the DB2 parameters.

   The Associate DB2 Entry for Product panel is displayed, and the new DB2 entry is displayed in the master list, as shown in the following figure:

   ![Associate DB2 Entry for Product panel](image)

   **Figure 8. The Associate DB2 Entry for Product panel with a new DB2 entry in the master list**

   c. Repeat steps b and c for each DB2 entry that you want to create.

d. When you have created all the DB2 entries, associate them with DB2 Path Checker, or press End to display the Customizer Workplace panel.

3. Associate DB2 entries.
a. Specify A against one or more DB2 entries in the master list, and press Enter to associate them with DB2 Path Checker.

Results

The Customizer Workplace panel is displayed with the associated DB2 entries displayed in the associated list.

What to do next

Define the parameters.

Related concepts:

“Tools Customizer terminology” on page 139

Tools Customizer uses several unique terms that you should be familiar with before you begin to use Tools Customizer.

Defining parameters

To customize DB2 Path Checker, you must define DB2 Path Checker parameters and DB2 parameters, if your customization requires DB2 entries.

About this task

You must define the DB2 Path Checker parameters first for the following reasons:

- If you ran the DB2 Path Checker Discover EXEC, you must review the values that were discovered.
- If you select optional tasks and steps on the Product Parameters panel that affect the DB2 entry on which you will customize DB2 Path Checker, additional parameters might be displayed on the DB2 Parameters panel.
- If other steps must be completed in a specific sequence, customization notes on the Product Parameters panel will display the correct sequence.

Defining DB2 Path Checker parameters

DB2 Path Checker parameters are specific to DB2 Path Checker.

About this task

If you ran the DB2 Path Checker Discover EXEC, you must review the parameters that were discovered.

Procedure

1. Specify E next to the Product parameters field on the Customizer Workplace panel, and press Enter. The Product Parameters panel is displayed, as shown in the following figure. If other steps must be completed in a specific sequence before you define the DB2 Path Checker parameters, a note labeled Important will display the correct sequence on this panel.
Browse the following tasks, steps, and parameters. The required and previously selected tasks and steps are indicated by a slash (/). Press End to exit.

Product to Customize

Product metadata library . : DB2TOOL.PTF420.SCKPEN > LPAR . : 3090
Product name . . . . . . . : DB2 Path Checker > Version . : 4.2.0

Product customization library .: CKP.PRODUCT.CUST.3090.$3090$.CKP420

Required parameters
HLQ of the Path Checker libraries . . . : DB2TOOL.CKP420 >

/ Path Checker customization

/ Create the Path Checker BIND job

/ Invocation EXEC for DB2 Path Checker

/ Create the invocation EXEC for Path Checker
DB2 Load Library . . . . . . . . . . : DSN.SDSNLOAD >

Copy CKP EXEC to SYSPROC library
SYSPROC library for CKP . . . . . : SYSX.SYSPROC >

Path Checker IVP jobs

/ Create the IVP BIND job

Create IVP invocation batch job

Customize the DB2 Administration Tool Launchpad for Path Checker

/ Add Path Checker to DB2 Administration Tool Launchpad
HLQ of the DB2 Admin libraries. . . . : DB2TOOLS.DB2ADMIN >

Complete the following tasks to customize the products. The required tasks and steps are preselected. Ensure that all parameters are specified for each selected step within a task. Press End to save and exit.

Commands: SAVE - Save parameter values
Line Commands: / - Select

Required parameters
DB2 Autonomics HLQ . . . : DB2TOOL.DYX150

/ Create or Drop DB2 Autonomics Director tables

Drop DB2 Autonomics tables

/ Create DB2 Autonomics Director tables

/ Bind and free packages and plans

Free packages and plans

/ Bind packages and plans

/ Grant EXECUTE privilege

/ Grant EXECUTE privilege on the plans

/ Set up WLM Environment for DB2 Autonomics Director

/ Define DB2 Autonomics Director WLM address space

Started Task PROCLIB . DYX.PROCLIB

FEC common code HLQ .

Figure 9. The Product Parameters panel
2. Select any required tasks and steps, and specify values for any parameters.
   After you select a task or step with a slash (/), put the cursor in the selected
   field and press Enter. If tasks, steps, and parameters are required, they are
   preselected with a slash (/). Otherwise, they are not preselected.
   All of the required parameters have default values, which you can either accept
   or change.

   Tips:
   - In the command line, specify the KEYS command, and map EXPAND
to one of the function keys.
   - For a detailed description of all input fields, put the cursor in the field,
   and press F1 or the key that is mapped to Help.
   - The following elements apply to specific fields:
     - **Add...** is displayed when parameters can have multiple values but
currently have only one value. To specify multiple values in these
fields, place the cursor on **Add...**, and press Enter. Use the displayed
panel to add or delete additional values.
     - **List...** is displayed when the complete list of valid values for the
fields is too long to be displayed on the panel. To see the complete
list of values, place the cursor on **List...**, and press F1 or the key that
is mapped to Help.
     - **More...** is displayed when input fields contains multiple values. To
see all of the values in the field, place the cursor on **More...**, and
press Enter.

3. Optional: Select other tasks and steps with a slash (/) and press Enter to
   activate the input fields. Either accept or change the default values that are
displayed.

4. Press End to save your changes and exit, or issue the SAVE command to save
   your changes and stay on the Product Parameters panel.

**Results**

The Customizer Workplace panel is displayed, and the status of the product
parameters is Ready to Customize.

**What to do next**

If the status of other parameters on the Customizer Workplace panel is Incomplete
or Discovered, edit these parameters.

**Related tasks:**

- ["Defining DB2 parameters"](defining-db2-parameters)
  DB2 parameters are parameters for a DB2 entry.

**Defining DB2 parameters**

DB2 parameters are parameters for a DB2 entry.

**About this task**

If you did not run the DB2 Path Checker Discover EXEC, you must create and
associate one or more DB2 entries before you can define the DB2 parameters. For
more information, see ["Creating and associating DB2 entries" on page 47](creating-and-associating-db2-entries).
Procedure

1. Specify E next to one or more DB2 entries in the associated list, which is in the Associated DB2 Entries and Parameter Status section on the Customizer Workplace panel, and press Enter. The DB2 Parameters panel is displayed, as shown in the following figure:
Browse the values for all of the DB2 parameters. Press End to exit.

Product to Customize
Product metadata library : DB2TOOL.PTF420.SCKPDEN > LPAR : 3090
Product name ............. : DB2 Path Checker > Version : 4.2.0

DB2 subsystem ID ............ : DB2A
Group attach name ............ :

General DB2 Information
Mode ....................... : NFM (CM, CM8, CM9, NFM)
Level number .............. : (910, 101, 111)

DB2 Libraries
Load library .............. : DSN.SDSNLOAD > Add...

Path Checker BIND parameters
OWNER option .............. : SYSADM >
QUALIFIER option ............ : SYSADM >
Group attach name ............ :

General DB2 Information
Mode ....................... : NFM (CM, CM8, CM9, NFM)
Level number .............. : (910, 101, 111)

DB2 Libraries
Load library .............. : DSN.SDSNLOAD > Add...

Path Checker BIND parameters
OWNER option .............. : SYSADM >
QUALIFIER option ............ : SYSADM >
DYNAMICRULES option .......... : RUN (RUN, BIND)

Enter values for all of the DB2 parameters. Press End to save and exit.

Commands: SAVE - Save parameter values
More: +

DB2 subsystem ID ............ : DB01
Group attach name ............ :

General DB2 Information - common
Mode ....................... : NFM (CM, NFM)
Level number .............. : 101 (101, 111)

DB2 Libraries - common
Load library .............. : DSN.SDSNLOAD Add
Run library .............. : DSN.RUNLIB.LOAD Add
Exit library .............. : DSN.SDSNEXIT Add

DB2 Buffer Pools - common
Name of the 4 KB buffer pool ........ .. BPO
Name of the 8 KB buffer pool ........ .. BP8K0

DB2 Utilities - common
SYSAFF for DB2 utilities ...........
Plan name for the DSNFIAO utility .... DSNFIAO

DB2 Autonomics BIND Parameters
BIND owner ID .............. : DB2USER >
User ID for GRANT statement ....... : PUBLIC

DB2 Autonomics DB2 Parameters
SET CURRENT SQLID ............ : DB2USER
DB2 Autonomics database name ......... : DXYDB
DB2 Autonomics index schema ............ : SYSAUTO >
DB2 Autonomics database STOGROUP ........ SYSAUTO3 >
DB2 Autonomics index STOGROUP ........ SYSAUTO >
DB2 Autonomics index buffer pool ........ .. BPO
Repository table space primary quantity -1
Repository table space secondary quantity -1
2. Specify values for all parameters that are displayed.

   Tips:
   - In the command line, specify the KEYS command, and map EXPAND to one of the function keys.
   - For a detailed description of all input fields, put the cursor in the field, and press F1 or the key that is mapped to Help.
   - The following elements apply to specific fields:
     - **Add...** is displayed when parameters can have multiple values but currently have only one value. To specify multiple values in these fields, place the cursor on **Add...**, and press Enter. Use the displayed panel to add or delete additional values.
     - **List...** is displayed when the complete list of valid values for the fields is too long to be displayed on the panel. To see the complete list of values, place the cursor on **List...**, and press F1 or the key that is mapped to Help.
     - **More...** is displayed when input fields contains multiple values. To see all of the values in the field, place the cursor on **More...**, and press Enter.

   Many parameters have default values, which you can either accept or change.

3. Press End to save your changes and exit, or issue the **SAVE** command to save your changes and stay on the same panel.

Results

The status of the DB2 entries that you selected on the Customizer Workplace panel is Ready to Customize.

What to do next

If the status of other parameters on the Customizer Workplace panel is Incomplete or Discovered, edit these parameters.

Related tasks:

[“Defining DB2 Path Checker parameters” on page 49](#)

DB2 Path Checker parameters are specific to DB2 Path Checker.

Generating customization jobs

To generate customization jobs for DB2 Path Checker and any associated DB2 entries, issue the **GENERATEALL** command, or select one or more DB2 entries on which to customize DB2 Path Checker.

Procedure

Generate the customization jobs by using one of the following methods.

- If you want to generate customization jobs at the product level and for any associated DB2 entries, issue the **GENERATEALL** command, and press Enter.
- If you want to generate customization jobs for specific DB2 entries, select the DB2 entries by specifying the **G** line command against them, and press Enter.

The available DB2 entries are in the associated list in the Associated DB2 Entries and Parameter Status section.
**Important:** Regenerating customization jobs will replace any existing jobs, including jobs that you might have manually modified after they were generated.

**Results**

If the status is Incomplete or Discovered for DB2 Path Checker parameters or DB2 parameters, Tools Customizer automatically starts an editing session for the types of parameters that are required. The session continues until the panel for each type of required parameter has been displayed.

**What to do next**

If an automatic editing session is started, accept the displayed parameter values or define values for the required types of parameters, select optional parameters, tasks, or steps for your environment, and save the parameter values. Otherwise, the customization jobs are generated, and you can submit them.

**Tip:** If the customization jobs are generated, but you are not ready to submit them, you can see them later by issuing the JOBLIST command on the Customizer Workplace panel. The JOBLIST command displays the Finish Product Customization panel, which you can use to submit the jobs.

**Submitting customization jobs**

Submit the customization jobs to customize DB2 Path Checker.

**Before you begin**

Ensure that the correct jobs are generated.

**About this task**

The following figure shows part of the Finish Product Customization panel. The table on this panel shows the customization jobs that are generated by Tools Customizer. They are grouped by job sequence number.
The member-naming conventions depend on whether the customization jobs are for DB2 entries, LPAR, or the product.

Customization jobs for DB2 entries
The members use the following naming convention:

\(<job_sequence_number><job_ID><DB2_entry_ID>\)

where

\(job_sequence_number\)
Two alphanumeric characters, A0 - Z9, that Tools Customizer assigns to a customization job. The number for the first template in the sequence is A0, the number for the second template is A1, and so on.

\(job_ID\)
Characters 4 - 7 of the template name, if the template name contains five or more characters. Otherwise, only character 4 is used. DB2 Path Checker assigns the template name.

\(DB2_entry_ID\)
Two alphanumeric characters, AA - 99, that Tools Customizer assigns to a DB2 entry.
For example, the XYZBNDDB2_entry_ID_1 and XYZBNDDB2_entry_ID_2 jobs are generated from the XYZBNDGR template, and the XYZDB2_entry_ID_1 and XYZDB2_entry_ID_2 jobs are generated from the XYZ template. If the jobs are generated on two DB2 entries, the following member names are listed sequentially: A0BNDGAA, A0BNDGAB, A14AA, A14AB.

**Customization jobs for the product**

The members use the following naming convention:

\<job_sequence_number\>\<job_ID\>

where

- **job_sequence_number** Two alphanumeric characters, A0 - Z9, that Tools Customizer assigns to a customization job. The number for the first template in the sequence is A0, the number for the second template is A1, and so on.

- **job_ID** Characters 4 - 8 of the template name, if the template name contains five or more characters. Otherwise, only character 4 is used. For example, for the XYZMAKE template, the job ID is MAKE. For the XYZM template, the job ID is M. DB2 Path Checker assigns the template name, and it is displayed in the Template column.

For example, the XYZBNDGR job is generated from the XYZBNDGR template, and the XYZ4 job is generated from the XYZ4 template. The following member names are listed sequentially: A0BNDGAA, A14AA.

**Procedure**

1. Submit the generated customization jobs by following the process that you use in your environment or by using the following method:
   a. Specify B against a customization job or the product customization library, and press Enter. An ISPF browsing session is started.
   b. Browse the customization job or each member in the library to ensure that the information is correct.
   c. Run the TSO SUBMIT command.

2. Press End.

**Results**

DB2 Path Checker is customized, and the Customizer Workplace panel is displayed. The status is Customized for the DB2 entries on which DB2 Path Checker was customized.

**What to do next**

You can generate more customization jobs for other DB2 entries, view a list of customization jobs that you previously generated, or recustomize DB2 Path Checker.

**Browsing parameters**

You can browse the product parameters and the DB2 parameters in read-only mode.
**Procedure**

1. On the Customizer Workplace panel, specify B next to the **Product parameters** field or the DB2 entry that you want to browse, and press Enter. The panel that corresponds to your specification is displayed.
2. Press End to exit.

**Copying DB2 entries**

You can copy associated and not associated DB2 entries to other DB2 entries or to new DB2 entries.

**About this task**

Go to the step that applies to your environment:

- To copy an associated DB2 entry to another associated DB2 entry or to an entry that is not associated, go to step 1.
- To copy an associated DB2 entry to a new entry, go to step 2.
- To copy a DB2 entry that is not associated to a new entry, go to step 3.

**Procedure**

1. To copy an associated DB2 entry to another associated DB2 entry or to an entry that is not associated, complete the following steps:
   a. Specify C against a DB2 entry in the associated list of DB2 entries on the Customizer Workplace panel, and press Enter. The Copy Associated DB2 Entry panel is displayed.
   b. Select one or more DB2 entries to which information will be copied by specifying the \ line command, and press Enter. The Associated column indicates whether the DB2 entry is associated.

   **Tip:** To copy information into all of the DB2 Entries in the list, issue the SELECTALL primary command, and press Enter.

   The Copy DB2 Parameter Values panel is displayed.
   c. Specify an option for copying common and product-specific DB2 parameter values. Common DB2 parameter values apply to all DB2 entries for all products that you have customized by using Tools Customizer. Product-specific DB2 parameter values apply only to the product that you are currently customizing.
      - To copy the common DB2 parameter values and the product-specific DB2 parameter values, specify option 1, and press Enter.
      - To copy only the product-specified DB2 parameter values, specify option 2, and press Enter.

   In some cases, the DB2 parameter values might contain the DB2 subsystem ID as an isolated qualifier in data set names. For example, in the DB01.DB01TEST.DB01.SANLOAD, data set name, the DB01 subsystem ID is isolated in the first and third qualifiers but is not isolated in the second qualifier. When the DB2 subsystem ID is an isolated qualifier in data set names, the Change DB2 Subsystem ID in DB2 Parameter Values panel is displayed. Otherwise, the Customizer Workplace panel is displayed.
   d. If the Change DB2 Subsystem ID in DB2 Parameter Values panel is displayed, specify an option for changing the subsystem IDs. Otherwise, skip this step.
      - To change the subsystem ID in isolated qualifiers in data set names, specify option 1, and press Enter.
To use the same subsystem ID in all values, specify option 2, and press Enter.

The Customizer Workplace panel is displayed with the copied associated entry in the list.

2. To copy an associated DB2 entry to a new entry, complete the following steps:
   a. Specify C against a DB2 entry in the associated list of DB2 entries on the Customizer Workplace panel, and press Enter. The Copy Associated DB2 Entry panel is displayed.
   b. Issue the CREATE command. The Create DB2 Entries panel is displayed.
   c. Specify the SSID, the group attach name, or both in the appropriate columns for each new DB2 entry, and press Enter.

   **Tip:** To add rows for additional entries, specify the `nn` line command, where `nn` is the number of entries to be created, and press Enter. The Copy Associated DB2 Entry panel is displayed with the new entries in the list. The new entries are preselected.
   d. Press Enter to complete the copy process. The Customizer Workplace panel is displayed with the copied entries in the list.

3. To copy a DB2 entry that is not associated to a new entry, complete the following steps:
   a. Issue the ASSOCIATE command on the Customizer Workplace panel. The Associate DB2 Entry for Product panel is displayed.
   b. Select one or more DB2 entries by specifying the `/` line command, and press Enter. The Copy a DB2 Entry panel is displayed.
   c. Specify the SSID, the group attach name, or both in the appropriate columns for the new DB2 entry, and press Enter. The Associate DB2 Entry for product panel is displayed with the copied entry in the list.
   d. If you want to associate the copied entry, specify A against it, and press Enter. The Customizer Workplace panel is displayed with the copied entries in the list.

**What to do next**

Edit any of the parameters or generate the jobs.

**Related concepts:**

"Tools Customizer terminology" on page 139

Tools Customizer uses several unique terms that you should be familiar with before you begin to use Tools Customizer.

**Removing DB2 entries**

You can remove DB2 entries from the associated list.

**About this task**

When you remove DB2 entries from the associated list, any customization jobs for the entries are removed from the list of jobs on the Finish Product Customization panel, and they are deleted.

**Procedure**

On the Customizer Workplace panel, specify R next to one or more DB2 entries that you want to remove, and press Enter. The selected DB2 entries are removed.
from the associated list and added to the master list on the Associate DB2 Entry for Product panel, and the customization jobs are deleted.

Related concepts:

“Tools Customizer terminology” on page 139
Tools Customizer uses several unique terms that you should be familiar with before you begin to use Tools Customizer.

Deleting DB2 entries

You can delete DB2 entries from the master list.

About this task

When you delete DB2 entries from the master list, any associations and all customization jobs for products that are customized on the entries will be deleted.

Procedure

1. On the Customizer Workplace panel, issue the ASSOCIATE command. The Associate DB2 Entry for Product panel is displayed.
2. Specify D next to one or more DB2 entries that you want to delete, and press Enter. If the entry is associated with any products, the Delete Associated DB2 Entry panel for the first DB2 entry that you selected is displayed. Otherwise, the Delete DB2 Entry panel is displayed.
3. To delete the DB2 entries, press Enter. If the DB2 entries are associated with any products in the table on the Delete Associated DB2 Entry panel, any associations and all customization jobs for the products that are customized on it are deleted. Otherwise, only the DB2 entries are deleted. If you selected multiple DB2 entries to delete, the next DB2 entry that you selected is displayed on either the Delete Associated DB2 Entry panel or the Delete DB2 Entry panel. Otherwise, the Associate DB2 Entry for Product panel is displayed.

What to do next

If you selected multiple DB2 entries to delete, repeat step 3 until all selected entries are deleted. Then, continue the customization process.

Displaying customization jobs

You can view a list of the members that contain the customization jobs before or after you submit the jobs.

About this task

The customization jobs that you generate for one DB2 entry are also displayed when you customize DB2 Path Checker for another DB2 entry later.

Procedure

On the Customizer Workplace panel, issue the JOBLIST command. The Finish Product Customization panel is displayed. This panel shows the list of jobs that you have previously generated. They are grouped by job sequence number. Use this panel to browse or edit the generated jobs before you submit them.
Maintaining customization jobs

Instead of deleting customization jobs outside of Tools Customizer, you can maintain the correct jobs for DB2 Path Checker by completing the steps for recustomization.

About this task

You cannot delete or rename customization jobs from the customization library by starting an ISPF browse or edit session from the Finish Product Customization panel. If you try to delete customization jobs by using this method, the CCQC034S message is issued. If you try to rename customization jobs, the CCQC035S message is issued.

If you delete or rename customization jobs from the customization library by using ISPF outside of Tools Customizer, Tools Customizer will not recognize that the jobs were deleted, and the Finish Product Customization panel will still display them. If you browse or edit jobs that were deleted from the library outside of Tools Customizer, the CCQC027S message is issued.

Procedure

To maintain the correct customization jobs in the customization library, complete the steps for recustomization.

Using Tools Customizer in a multiple-LPAR environment

Currently, Tools Customizer supports only the local LPAR; however, you can propagate customizations to additional LPARs by using either of two different methods.

About this task

In a multiple-LPAR environment, Tools Customizer identifies the LPAR to which you are logged on. Tools Customizer uses this LPAR name for several different parameter settings, one of which is the data store. When you use the data store during the customization of DB2 Path Checker that is on a different LPAR, Tools Customizer issues message CCQD586S, which indicates that the product has already been customized based on values from the data store on the first LPAR. This message is issued to prevent the data store from becoming corrupted.

This behavior occurs in the following conditions:
- Tools Customizer is installed on a DASD device that is shared by multiple LPARs.
- After a product is customized by using Tools Customizer, the data store is copied to another LPAR.

Procedure

To customize products running against a DB2 subsystem on an LPAR where Tools Customizer is not installed, consider using one of the following methods:

Install one instance of Tools Customizer on one LPAR
- If you intend to reuse the customization values for all the instances of your products on all LPARs, use this method.
  1. Associate all the DB2 entries in this one instance of Tools Customizer. The LPARs on which the DB2 subsystems reside do not matter.
2. Generate the customization jobs for each DB2 entry.

3. Copy the generated customization jobs to the LPAR to run against the specific DB2 entries. Some LPAR-specific edits might be required. You can make these edits in the customized jobs that you copied. Note that this situation is one of the few situations where you might need to make manual changes to the jobs that are customized by Tools Customizer.

**Install one instance of Tools Customizer on each LPAR**

If you do not want to reuse previous customization values and you want to start new customizations, use this method.

**Important:** This method will likely not be the preferred approach for most organizations because most organizations tend to use similar or identical customization values for each product instance on all LPARs.
Chapter 3. Plan tables

A plan table (user.PLAN_TABLE) must exist to use the EXPLAIN statement, including the dynamic EXPLAIN that is issued by DB2 Path Checker.

When you test or compare programs by using the TEST and EXPLAIN commands, the plan table that is used by dynamic EXPLAIN in DB2 Path Checker and the plan table that contains the current path information from a bind with EXPLAIN(YES) can be the same table.

DB2 Path Checker works with any valid DB2 plan table or data type in any supported release of DB2. When DB2 Path Checker works with two plan tables, it does not require that both tables have the same format. It will compare data that is common to the two formats. However, both tables must be valid plan tables with column structures that are supported by their respective DB2 releases.

The easiest way to use DB2 Path Checker is to make backups of plan table data and create periodic archives, depending on migration frequency. These backups can be created with simple SQL, as shown in the following example:

```sql
DELETE FROM PLAN_TABLE2;
COMMIT;
INSERT INTO PLAN_TABLE2 SELECT * FROM PLAN_TABLE
```

You can also use options such as DSNTIAUL or REORG UNLOAD ONLY EXTERNAL to create backups of plan table data. You can use the previous versions of these tables to make multiple comparisons and identify instances of access path changes.

If the plan table rows are from a bind for a plan, DB2 uses APPLNAME (plan) and PROGNAME (DBRM) to identify the data. If the plan table rows are from a bind for a package, DB2 uses COLLID (collection identifier) and PROGNAME (DBRM).

If the plan table rows are from a dynamic SQL EXPLAIN, the following conditions apply:

- If the EXPLAIN program was bound as a package collection, COLLID and PROGNAME will specify the active program and the collection that it came from.
- If the EXPLAIN program was bound as a PLAN or DBRM, only PROGNAME is used.

DB2 Path Checker issues multiple SQL requests against both plan tables, so indexes on APPLNAME, COLLID, PROGNAME, TIMESTAMP (as descending), QUERYNO (as descending), QBLOCKNO, PLANNO, and MIXOPSEQ will improve performance.

If a DSN_STATEMNT_TABLE is associated with the PLAN_TABLE, DB2 Path Checker shows the DB2 cost estimates for the SQL in the report.

If a DSN_STATEMNT_TABLE is defined, DB2 Path Checker reports on the cost estimates from the DB2 Optimizer. This is more efficient if an index is defined on the DSN_STATEMNT_TABLE with columns, APPLNAME, COLLID, PROGNAME, TIMESTAMP (as descending), and QUERYNO.
Qualifying tables for SQL Explain processing

In bind processing, the target PLAN_TABLE, which is the current $sqlid$.PLAN_TABLE, and the target tables that will be processed by the SQL statements are qualified separately. However, SQL EXPLAIN processing requires that you combine these processes when tables are qualified. If the qualifying process is not done correctly, the tables will be created with the wrong qualifier.

About this task

The PLAN_TABLE is qualified by the user ID in the owner parameter, and the application tables are qualified by the user ID that is specified for the qualifier parameter.

To ensure that the tables are qualified correctly, use one of the following methods:

- Create a PLAN_TABLE under the ID that was used for the qualifier parameter and set the current SQLID to that value before issuing the TEST or EXPLAIN command.
- Use the PLAN_TABLE from the owner parameter and create aliases or synonyms to the application tables using the SPUFI that is shown in the following figure to generate CREATE SYNONYM commands. This SPUFI only allows for one set of applications to be mapped to the OWNER.PLAN_TABLE at a time.
- Specify OPTIONS CREATE TABLES before the TEST or EXPLAIN command. Specifying OPTIONS CREATE TABLES first creates the PLAN_TABLE and DSN_STATEMNT_TABLE that are required to process the TEST and EXPLAIN commands. These tables are dropped at the end of the process. They are created as implicit tables where the CREATE statement does not specify a table space.

The SPUFI shown in the following figure generates the synonyms that are required for a single set of application tables.

```
-- THIS WILL EXAMINE ALL THE TABLES FOR CREATOR ID = PUBLIC01
-- AND CREATE A SYNONYM FOR THAT TABLE UNDER P390H IF
-- THERE ISN'T A SYNONYM OR ALIAS FOR THAT TABLE NAME ALREADY
-- NOTICE THAT THE SUBSELECT FROM SYSTABLES DOESN'T QUALIFY
-- THE TYPE VALUE BECAUSE EITHER AN ALIAS, TABLE OR VIEW
-- WILL PREVENT THE CREATION OF A NEW SYNONYM
SELECT 'CREATE SYNONYM ' CONCAT RTRIM(NAME) CONCAT ' FOR ' CONCAT RTRIM(CREATOR) CONCAT '.' CONCAT RTRIM(NAME) CONCAT ';'
FROM SYSIBM.SYSTABLES A
WHERE CREATOR = 'PUBLIC01'
AND NOT EXISTS (SELECT 1 FROM SYSIBM.SYSSYNONYMS B
WHERE B.CREATOR = 'P390H'
AND B.NAME = A.NAME)
AND NOT EXISTS (SELECT 1 FROM SYSIBM.SYSTABLES C
WHERE C.CREATOR = 'P390H'
AND C.NAME = A.NAME)
ORDER BY 1
```

Figure 12. Synonym generation example
Creating optimization hints for plan tables

Use the MAKE command to influence the access paths that DB2 selects for processing SQL statements.

About this task

Use the MAKE command to mark an existing access path in a plan table as a hint. A bind is required to make the hint active. The bind accepts a hint only if the correct DSNZPARM has been set to activate hints. For more information about OPTHINTS, see DB2 Installation Guide.

You can also create hints that specify subsystem parameters that are related to optimization. DB2 can then use these parameters for particular SQL statements.

A bind is required to make the hint active. The BIND accepts a hint only if the correct DSNZPARM has been set to activate hints. For more information about optimization hints, see the DB2 for z/OS Managing Performance.

To create optimization hints for plan tables:

Procedure

1. Customize the sample job to reflect site standards.
2. Specify ddnames that point to the required libraries.
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
3. Specify a CONNECT command to establish a session with the DB2 subsystem.
4. Specify an OPTIONS command to set processing behavior.
5. Optional: Specify a SET CURRENT SQLID command.
6. Specify a MAKE command.
   a. Provide the query number (QUERYNO) of the SQL statement.
   b. Specify the value of the TIMESTAMP column that, combined with the query number, uniquely identifies the row that you want to mark.
   c. Specify whether the object type of the plan table entry is a plan or package.
   d. Specify the name of the plan or collection that contains the program.
   e. Provide the name of the program.
   f. Provide the name of the plan table that contains the row that you want to mark.
   g. Provide the hint name.

Example

This sample job marks an access path with the name of an optimization hint for SQL statement 200. A bind generated this access path on December 12, 2012. The bind was for the program MYPROG in the collection MYCOLL, and the access path is in the plan table ADMF001.PLAN_TABLE.

//SAMPLE01 JOB ,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//*
//STEP001 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP.LOADLIB
// DD DISP=SHR,DSN=DB2A.SDSNLOAD
//SYSPRINT DD SYSOUT=*
CONNECT TO DB2A
SET CURRENT SQLID = 'ADMF001'
MAKE STATEMENT 200 WITH TIMESTAMP='2012121222421165'
IN PACKAGE MYCOLL.MYPROG IN ADMF001.PLAN_TABLE
BE HINT MYHINT
/*
Chapter 4. Running DB2 Path Checker

You can run DB2 Path Checker by submitting batch jobs that you create yourself, or you can use the ISPF interface to create the jobs that you require.

About this task

By running DB2 Path Checker as a batch job, you can take advantage of all the available features. You can run many different commands in a single run, such as tests, comparisons, or EXPLAIN operations. You can create scripts to automate your analysis of access path changes, parse reports, and find the information that you need.

DB2 Path Checker Interactive is an ISPF interface that provides online access to a subset of DB2 Path Checker functions. You can use it to display information about your plan table, compare access paths for plans and packages, test DBRMs, and generate batch jobs automatically. By using the utility functions, you can also create new plan tables or delete all rows in an existing plan table.

Restriction:

If you use the SYSIN data set as an input data set for commands and a long name does not fit on one 80-character line, DB2 Path Checker cannot process the command. The following list provides examples of names that cannot exceed 80 characters:

- Package names
- Collection names
- Plan table names
- Version names

Related information:

See the DB2 Universal Database for z/OS Administration Guide or the DB2 Universal Database for z/OS Application Programming and SQL Guide for more information about DSN_STATEMENT_TABLEs.

Running DB2 Path Checker by submitting batch jobs

Run DB2 Path Checker by submitting a batch job that contains the required load libraries, DD statements, options, and commands.

About this task

You can use the full range of features that are available in DB2 Path Checker when you run the tool as a batch job. The main advantage of running DB2 Path Checker in this way is that you can automate many commands and run them in consecutive steps. You can run multiple tests, comparisons, or EXPLAIN operations in sequence.

The following examples demonstrate only some of the possibilities:

- Test a DBRM and then run another test on a different DBRM.
• Generate one or more reports.
• Compare a package with the previous bind.

You can even use automation to create the jobs themselves, to read the resulting reports, and to elicit the information that is most important to you. For example, suppose that you want to generate a list of access paths that reference a deprecated index or table. You can search the output of TEST commands that use the REPORTALL option to generate the list.

You can also provide input commands to DB2 Path Checker by supplying them in SYSIN or by specifying them in the PARM field of the EXEC statement. If you do not include commands in your batch job, however, supply them in SYSIN; DB2 Path Checker supports the PARM field only for compatibility with previous releases.

The information that DB2 Path Checker requires depends on the type of analysis that you request. You must specify the DB2 load library in a STEPLIB or JOBLIB DD statement because DB2 Path Checker uses the DB2 call attachment facility.

For more information about DB2 Path Checker command syntax and job control statements, see Chapter 7, “Reference,” on page 139.

Procedure

You can perform most DB2 Path Checker tasks by modifying the sample job, which is provided in the samples library. Use one of the following sample jobs, depending on the version of DB2 that you are running:

• DB2 V8 to V10: high-level-qualifier.SCKPSAMP(CKPIVP)
• DB2 V11: high-level-qualifier.SCKPSAMP(CKPIVP11)

The following sections describe how to modify the sample job to perform these tasks.

Related reference:

“DB2 Path Checker command reference” on page 144
Use this information to learn how to use the commands that are available in DB2 Path Checker. Refer to the topics about individual commands to find the correct command syntax.

“Job control statements” on page 174
Job control statements specify DB2 Path Checker functions and what options to use.

Selecting programs for processing by using name substitution

You can specify the programs that DB2 Path Checker commands process by specifying a ddname that references a data set that contains bind control cards.

About this task

Name substitution is specific to //BINDIN processing. The optional BINDIN DD statement generates commands from bind control cards. When you issue this command, DB2 Path Checker parses the bind control cards and issues a command for each plan or package that is specified in the control cards. DB2 Path Checker
ignores the plan and package name in the commands and identifies the plan or package to be processed from the BIND control commands.

The input command can specify a real name or just a single asterisk (*). The plan name or collection ID and program name are generated from the BIND control cards. You can process only one command with BINDIN controlling the plan, collection, and program name.

For example, suppose that you combine the bind control commands that are shown in the following figure with the DB2 Path Checker commands (Figure 14). DB2 Path Checker selects the two packages (Figure 15) to be processed as though you specified the command in (Figure 16).

The message CKP245I displays the BINDIN ddname to identify the source of the program names that DB2 Path Checker is processing.

```sql
//BINDIN DD *
  DSN SYSTEM(DB7G)
  BIND PACKAGE(ACCTPAYX) MEMBER(TEST01) -
    ACTION(REPLACE) VALIDATE(BIND) ISOLATION(CS) EXPLAIN(YES)
  BIND PACKAGE (ACCTPAYX) MEMBER (TEST02) -
    ACTION(REPLACE) VALIDATE(BIND) ISOLATION(CS) EXPLAIN(YES)
END
```

_Figure 13. Bind control commands_

```sql
CONNECT TO DB7G
  COMPARE PACKAGE *.* IN P390H.PLAN_TABLE TO PREVIOUS VERSION
```

_Figure 14. DB2 Path Checker commands_

```sql
collid=ACCTPAYX and program name TEST01
collid=ACCTPAYX and program name TEST02
```

_Figure 15. Identified packages_

```sql
CONNECT TO DB7G
  COMPARE PACKAGE ACCTPAYX.TEST01 IN P390H.PLAN_TABLE TO PREVIOUS VERSION
  COMPARE PACKAGE ACCTPAYX.TEST02 IN P390H.PLAN_TABLE TO PREVIOUS VERSION
```

_Figure 16. Generated command_

Each program name that is selected from BINDIN generates message CKP246I to identify the program name that DB2 Path Checker selected from the BIND command for processing.

Related reference:

- “Job control statements” on page 174

Job control statements specify DB2 Path Checker functions and what options to use.

---

**Generating information about potential changes to access paths for plans and packages**

Generate information about access paths at different points in time to determine how changes in your DB2 environment might affect the performance of your SQL statements.
**About this task**

DB2 Path Checker helps you to evaluate hardware changes or data-related changes in your DB2 environment that can affect performance.

The following examples describe some of these possible changes:
- Large influxes of data
- New indexes
- Addition of parallel processing
- Introduction of zIIP or zAAP technologies

The `EXPLAIN` commands that are available in DB2 Path Checker generate access path information in your plan table that you can use later for testing, comparisons, and reports.

**Related reference:**

"EXPLAIN PACKAGE command” on page 151

The `EXPLAIN PACKAGE` command extracts the access path information for a package from the DB2 directory.

**Generating access path information for a DBRM**

Use the `EXPLAIN DBRM` command to populate your plan table with access paths that reflect the current state of your DB2 environment.

**About this task**

Complete this task to generate access paths for a DBRM that you did not bind yet. In this example, the DBRM type is a package rather than a plan.

These access paths are the same ones that a DB2 `BIND` command would generate if you bound the package instead of issuing the `EXPLAIN DBRM` command. Therefore, you do not have to bind the package to place the information in your plan table.

DB2 Path Checker uses the DBRM itself or the DB2 catalog as a data source, depending on the options that you specify.

**Procedure**

1. Customize the sample job to reflect your site’s standards.
2. Specify ddnames that point to the required libraries:
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
   - The DBRMLIB for DB2 for z/OS
3. Specify a `CONNECT` command to establish a session with the DB2 subsystem.
4. Specify an `OPTIONS` command to set processing behavior.
5. Optional: Specify a `SET` command to set the current SQLID, catalog qualifier, or schema.
6. Specify an `EXPLAIN DBRM` command.
   a. Provide the name of the DBRM for which you want to generate fresh access paths.
   b. Specify that the type of DBRM is a package.
   c. Provide the collection ID for the collection that references the package.
   d. Provide the name of the plan table that DB2 Path Checker reads for input.
Example

The following example generates access plan information for a DBRM.

```
//SAMPLE01 JOB ,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//*
//STEP001 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP.LOADLIB
// DD DISP=SHR,DSN=DB2A.SDSNLOAD
//DBRMIN DD DISP=SHR,DSN=MY.DBRMLIB
//SYSPRINT DD SYSOUT=* 
//ANLOUT DD SYSOUT=* 
//SYSEXPLN DD SYSOUT=* 
//SYSCCHG DD SYSOUT=* 
//RBindOUT DD SYSOUT=* 
//PBindOUT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSIN DD *
CONNECT TO DB2A
OPTIONS REPORTCHG
SET CURRENT SQLID = 'ADMF001'
EXPLAIN DBRM CKPTDBRM TO PACKAGE ACCTPAY
   IN ADMF001.PLAN_TABLE
/*

Related reference:

“Job control statements” on page 174
Job control statements specify DB2 Path Checker functions and what options to use.

“EXPLAIN command” on page 149
Follow the syntax and descriptions in these topics to use the EXPLAIN commands that are available in DB2 Path Checker.

“CONNECT command” on page 148
Follow the syntax and descriptions in this topic to use the CONNECT command.

“OPTIONS command” on page 154
Follow the syntax and descriptions in this topic to use the OPTIONS command.

“SET command” on page 166
Follow the syntax and descriptions in this topic to use the SET command.

Generating access path information for a package

Use the EXPLAIN PACKAGE command to populate your plan table with access paths that reflect the state of your DB2 environment at the point in time when DB2 bound a package.

About this task

Complete this task to generate access plan information for a package.

The EXPLAIN PACKAGE command reads data from the DB2 system catalog, which contains the most accurate EXPLAIN information that is available for previous binds or rebinds of a package. Any path analyses or comparisons that DB2 Path Checker might do later are only as good as the accuracy of the information in the plan table.

In this task, you use the EXPLAIN command to process the most recent bind of the package and write the access paths to owner.PLAN_TABLE. This task uses the default options to select the package to process. DB2 Path Checker selects the active edition or version of the program that is running in DB2. You can use more
options for the EXPLAIN PACKAGE command that are not described in this task to specify either a specific version name or the current, previous, or original version or edition.

Restrictions:

- The EXPLAIN PACKAGE command is available only in DB2 V10 NFM or later. If you run EXPLAIN PACKAGE against an earlier version of DB2, DB2 Path Checker issues an error message and does not do EXPLAIN processing.
- The EXPLAIN PACKAGE command updates the plan table that you specify. It does not update the DSN_STATEMENT_TABLE with cost information.

Procedure

1. Customize the sample job to reflect your site's standards.
2. Specify ddnames that point to the required libraries:
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
3. Specify a CONNECT command to establish a session with the DB2 subsystem.
4. Specify an EXPLAIN PACKAGE command.
   a. Specify the package and collection for which you want to generate access path information.
   b. Specify the name of the plan table where DB2 Path Checker writes the access paths.

Example

The following example job generates access plan information in the plan table ADMF001.PLAN_TABLE for the most recent bind of the package.

```
//SAMPLE02 JOB ,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//STEP002 EXEC PGM=CKPTPATHCK
//STEP11 DD DISP SHR,DSN=CKP.LOADLIB
// DD DISP=SHR,DSN=DB2A.SDSNLOAD
//SYSPRINT DD SYSOUT=* 
//ANLOUT DD SYSOUT=* 
//SYSEXPLN DD SYSOUT=* 
//SYSCHG DD SYSOUT=* 
//RBINDOUT DD SYSOUT=* 
//PBINDOUT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSDIAGN DD SYSOUT=* 
//SYSIN DD *
   CONNECT TO DB2A 
   EXPLAIN PACKAGE CKPTDBRM COLLECTION ACCTPAY
   TO ADMF001.PLAN_TABLE
/*
```

Related reference:

- "Job control statements" on page 174
  Job control statements specify DB2 Path Checker functions and what options to use.
- "EXPLAIN command" on page 149
  Follow the syntax and descriptions in these topics to use the EXPLAIN commands that are available in DB2 Path Checker.
- "CONNECT command" on page 148
  Follow the syntax and descriptions in this topic to use the CONNECT command.
Evaluating potential performance degradation of bind or rebind operations

Use DB2 Path Checker to learn about potential effects on program performance as a result of bind or rebind operations.

About this task

You can test the differences that occur in access paths if you bind or rebind a program. You can also test the differences between existing binds, or between binds and EXPLAIN results.

DB2 Path Checker can test or compare access paths in the following ways:
- Test the consequences of rebinding the active version of a DBRM
- Test the consequences of binding an unbound DBRM, typically a new program version
- Compare the differences between the current version of a program and the previous version
- Compare the differences between binds of the same version of a program
- Compare differences between a bound program and the results of EXPLAIN operations

Attention:

Do not rely on cost estimates in plan tables when you decide whether to rebind a plan or package. Access path data is transient and provides only a snapshot of your DB2 environment at a specific point in time. With the passage of time, cost estimates in your plan table can become increasingly inaccurate and misleading.

Determining potential access path changes since the previous bind

Use the TEST command to determine which access paths change if you rebind the current version of a DBRM.

About this task

Complete this task to determine the access paths that will change if you rebind the current version of a DBRM.

In this task, you use the FROM CATALOG option to specify the DB2 system catalog as the source of input data instead of a DBRM name. The program is a package.

The TEST command in this example compares the following set of access paths:
- The access paths for the current point in time
  - The DBRM name in the command specifies the source of this data. These access paths are the same ones that DB2 would generate if you bound the package at the time you issue the command.
- The access paths as of the bind time of the active program that is running in DB2
The DB2 system catalog is the source of this data.

When you issue the TEST command runs, DB2 Path Checker compares access paths for the current point in time with the access paths that reflect the actual bind time of the program.

Procedure
1. Customize the sample job to reflect your site's standards.
2. Specify ddnames that point to the required libraries and data sets:
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
   - A ddname for DBRMIN
3. Specify a CONNECT command to establish a session with the DB2 subsystem.
4. Optional: Specify an OPTIONS command to set processing behavior, such as the level of detail in reports or the method of matching SQL statements.
5. Optional: Specify a SET command to set the current SQLID, catalog qualifier, or schema.
6. Specify a TEST command.
   a. Provide the name of the DBRM that you want to test.
   b. Specify that the type of DBRM is a package.
   c. Provide a plan name or collection ID in qualifier.* format.
   d. Provide the name of the plan table that DB2 Path Checker reads for input.

Example

The following example job tests the package CKPTDBRM in collections that match the specification ACCTPAY.* for access path changes since the previous bind. The system catalog provides the input data for the previous bind.

In the example, you set CPUPCT 10 to determine whether the package is safe to rebind. If it is, DB2 Path Checker generates a REBIND command automatically in RBINDOUT. DB2 Path Checker generates the REBIND command under either of the following conditions:
   - The access path did not change
   - The PROCMS for the new access path is less than the PROCMS for the old access path plus 10 percent

//SAMPLE01 JOB,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
/**
//STEP003 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP.LOADLIB
//DD DISP=SHR,DSN=DB2A.SDSNLOAD
//DBRMOUT DD DISP=SHR,DSN=CKP.DBRMOUT
//SYSPRINT DD SYSOUT=* 
//ANLOUT DD SYSOUT=* 
//SYSEXPLN DD SYSOUT=* 
//SYSCHG DD SYSOUT=* 
//RBINDOUT DD SYSOUT=* 
//PBINDOUT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSDIAGN DD SYSOUT=* 
//SYSIN DD *
CONNECT TO DB2A
OPTIONS REPORTCOSTGT MATCHSQL2 REPORTCHG CPUPCT 10
SET CURRENT SQLID = 'ADMF001'
TEST DBRM CKPDBRM AS PACKAGE ACCTPAY.*
   IN ADMF001.PLAN_TABLE FROM CATALOG
/

Related tasks:
“Testing access path data for a DBRM” on page 88
You can use the TEST function to test access path data for a DBRM.

Related reference:
“TEST command” on page 168
Follow the syntax and descriptions in this topic to use the TEST command.
“Job control statements” on page 174
Job control statements specify DB2 Path Checker functions and what options to use.
“CONNECT command” on page 148
Follow the syntax and descriptions in this topic to use the CONNECT command.
“OPTIONS command” on page 154
Follow the syntax and descriptions in this topic to use the OPTIONS command.
“SET command” on page 166
Follow the syntax and descriptions in this topic to use the SET command.

Determining access path changes for different versions of the same program
Use the TEST and COMPARE commands to learn which access paths change if you bind a new version of a program.

About this task
The command that you choose depends on whether you bound the new version of the DBRM yet. If you did not yet bind the package, use the TEST command. If you already did bind the package, use the COMPARE command.

Testing an unbound DBRM against the previous program version:
Use the TEST command to compare a DBRM that you did not bind yet against the previous version of the package.

About this task
Complete this task to test an unbound DBRM against a previous program version.

In this task, you use the EXPLAIN command to process a DBRM and generates access paths that reflect the current state of your DB2 environment. It then compares them with the access paths from the previous version in your plan table. In this task, the type of program is a package.

The resulting report shows you which access paths would change if you bound the DBRM.

Procedure
1. Customize the sample job to reflect your site’s standards.
2. Specify ddnames that point to the required libraries:
   • The load library for DB2 Path Checker
   • The load library for DB2 for z/OS
• A ddname for DBRMIN

3. Specify a CONNECT command to establish a session with the DB2 subsystem.

4. Optional: Specify an OPTIONS command to set processing behavior, such as the level of detail in reports or the method of matching SQL statements.

5. Optional: Specify a SET command to set the current SQLID, catalog qualifier, or schema.

6. Specify a TEST command.
   a. Provide the name of the DBRM for which you want to generate fresh access paths.
   b. Specify that the type of DBRM is a package.
   c. Provide a plan name or collection ID in qualifier.* format.
   d. Provide the name of the plan table that contains the access paths from the previous version of the package.

Example

The following example job tests the package CKPTDBRM in all collections that match the specification ACCTPAY.* against the previous version that is stored in the plan table SYSADM.PLAN_TABLE.

```plaintext
//SAMPLE05 JOB ,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//STEP005 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP..LOADLIB
//       DD DISP=SHR,DSN=DB2A.SDSNLOAD
//DBRMIN DD DISP=SHR,DSN=CKP..DBRM
//SYSPRINT DD SYSOUT=*
//ANLOUT DD SYSOUT=*
//SYSEXPLN DD SYSOUT=*
//SYSCHG DD SYSOUT=*
///RBINDOUT DD SYSOUT=*
//PBINDOUT DD SYSOUT=*
//SYSDIAGN DD SYSOUT=*
//SYSIN DD *
   CONNECT TO DB2A
   OPTIONS REPORTCOSTGT MATCHSQL2 REPORTCHG CPUPCT 10
   SET CURRENT SQLID = 'SYSADM'
   TEST DBRM CKPTDBRM AS PACKAGE ACCTPAY.*
      IN SYSADM.PLAN_TABLE
/*

Related reference:
"TEST command" on page 168
Follow the syntax and descriptions in this topic to use the TEST command.

Comparing two bound versions of a program:

Use the COMPARE command to compare the access paths for the current version of a program with the access paths for the previous version.

About this task

Complete this task to compare two bound versions of a program.

In this task, the program type is a package. You already bound both versions with EXPLAIN(YES), so the access path information exists in the plan table. The resulting report shows you which access paths changed between the previous version and the current version.
Use the **TO PREVIOUS VERSION** option for this task to ensure that DB2 Path Checker selects the previous version of the package. The **TO PREVIOUS** option does not apply to this task because it might select a previous bind of the current version instead of the previous version.

**Procedure**

1. Customize the sample job to reflect your site's standards.
2. Specify ddnames that point to the required libraries:
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
   - A ddname for DBRMIN
3. Specify a **CONNECT** command to establish a session with the DB2 subsystem.
4. Optional: Specify an **OPTIONS** command to set processing behavior, such as the level of detail in reports or the method of matching SQL statements.
5. Optional: Specify a **SET** command to set the current SQLID, catalog qualifier, or schema.
6. Specify a **COMPARE** command.
   a. Specify that the object type is a package.
   b. Provide a plan name or collection ID in `qualifier.*` format.
   c. Provide the name of the plan table that contains the access paths for both versions of the package.
   d. Specify the **TO PREVIOUS VERSION** option.

**Example**

The following example job compares each package in the plan table `SYSADM.PLAN_TABLE` that matches the collection specification `ACCTPAY.*` against the previous version.

```
//SAMPLE04 JOB MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//STEP004 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP..LOADLIB
// DD DISP=SHR,DSN=DB2A.SDSNLOAD
//DBRMIN DD DISP=SHR,DSN=CKP..DBRM
//SYSPRINT DD SYSOUT=* ANLOUT DD SYSOUT=* SYSEXPLN DD SYSOUT=*
//SYSCRG DD SYSOUT=* //BINDOUT DD SYSOUT=* SYSDIAGN DD SYSOUT=* //SYSLIB DD *
//CONNECT TO DB2A OPTIONS REPORTCOSTGT MATCHSQL2 REPORTCHG CPUPCT 10
//SET CURRENT SQLID = 'ADMF001'
//COMPARE PACKAGE ACCTPAY.* IN SYSADM.PLAN_TABLE TO PREVIOUS VERSION
/*

**Related reference:**

- "Job control statements" on page 174
  Job control statements specify DB2 Path Checker functions and what options to use.
- "COMPARE command" on page 145
  Follow the syntax and descriptions in this topic to use the **COMPARE** command.
Determining access path changes by comparing the results of bind or EXPLAIN operations

Use the **COMPARE** command to learn which access paths changed between binds or between a bind and previously generated EXPLAIN results in a plan table.

Comparing access paths for different binds of the same program version:

Use the **COMPARE** command to determine which access paths changed between
binds of the same version of a plan or package.

Before you begin

More than one bind must exist in the plan table for the current version of the
package.

About this task

Complete the following task to compare a package with the previous bind of the
same program version.

This task uses the TO PREVIOUS option to ensure that DB2 Path Checker selects
the previous bind of the current version of the package. The TO PREVIOUS
VERSION option does not apply to this task. If you specify TO PREVIOUS
VERSION, DB2 Path Checker skips the previous bind of the current version and
selects the previous version for comparison instead.

The resulting report shows you which access paths changed between binds.

Procedure

1. Customize the sample job to reflect your site's standards.
2. Specify ddnames that point to the required libraries:
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
   - The DBRMLIB for DB2 for z/OS
3. Specify a **CONNECT** command to establish a session with the DB2 subsystem.
4. Optional: Specify an **OPTIONS** command to set processing behavior, such as the
   level of detail in reports or the method of matching SQL statements.
5. Optional: Specify a **SET** command to set the current SQLID, catalog qualifier, or
   schema.
6. Specify a **COMPARE** command.
   a. Specify that the object type is a package.
   b. Provide a plan name or collection ID in qualifier.* format.
   c. Provide the name of the plan table that contains the access paths for both
      binds of this package version.
   d. Specify the **TO PREVIOUS** option.
Example

The following example job compares the current version of each package in the plan table SYSADM.PLAN_TABLE whose collection ID matches the specification ACCTPAY.* with the most recent version of the program.

```objectivec
//SAMPLE04 JOB ,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//STEP004 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP..LOADLIB
// DD DISP=SHR,DSN=DB2A.SDSNLOAD
//DBRMIN DD DISP=SHR,DSN=CKP..DBRM
//SYSPRINT DD SYSOUT=* 
//ANLOUT DD SYSOUT=* 
//SYSEXPLN DD SYSOUT=* 
//SYSCHG DD SYSOUT=* 
//RBindOUT DD SYSOUT=* 
//PBINDOUT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSDIAGN DD SYSOUT=* 
//SYSIN DD *
   CONNECT TO DB2A
   OPTIONS REPORTCOSTGT MATCHSQL2 REPORTCHG CPUPCT 10 
   SET CURRENT SQLID = 'ADMF001' 
   COMPARE PACKAGE ACCTPAY.* IN SYSADM.PLAN_TABLE 
   TO PREVIOUS 
/*
Related concepts:

"DB2 Path Checker features and benefits" on page 6
DB2 Path Checker helps you to identify and address performance issues that are related to the SQL statements in your applications.

"Retrieve hints, recommendations, and expert advice from SQL Performance Analyzer" on page 17
You can use DB2 Path Checker with IBM DB2 SQL Performance Analyzer for z/OS for more detailed cost analysis of the SQL statements that DB2 Path Checker selects.

Related tasks:

"Comparing access paths for packages" on page 84
You can use the COMPARE function to compare two packages. DB2 Path Checker supports two comparison methods: normal and compare-to-previous.

"Comparing access paths for plans" on page 86
You can use the COMPARE function to compare two plans. DB2 Path Checker supports two comparison methods: normal and compare-to-previous.

Related reference:

"Job control statements" on page 174
Job control statements specify DB2 Path Checker functions and what options to use.

"COMPARE command" on page 145
Follow the syntax and descriptions in this topic to use the COMPARE command.

"CONNECT command" on page 148
Follow the syntax and descriptions in this topic to use the CONNECT command.

"OPTIONS command" on page 154
Follow the syntax and descriptions in this topic to use the OPTIONS command.

"SET command" on page 166
Follow the syntax and descriptions in this topic to use the SET command.
Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands:

Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.

About this task

Complete this task to compare access paths for the current edition of a program with access paths that were generated by EXPLAIN processing.

In this task, no access path information for the most recent bind of the package is available in the plan table. No data is available because the previous bind ran with the EXPLAIN(NO) option. You first generate fresh EXPLAIN data in the plan table by issuing an EXPLAIN PACKAGE PREVIOUS command. You now have the required access path information to test the current edition of the program and to determine which access paths change if you rebind.

Procedure

1. Customize the sample job to reflect your site's standards.
2. Specify ddnames that point to the required libraries and data sets:
   - The load library for DB2 Path Checker
   - The load library for DB2 for z/OS
   - A ddname for DBRMIN
3. Specify a CONNECT command to establish a session with the DB2 subsystem.
4. Optional: Specify an OPTIONS command to set processing behavior, such as the level of detail in reports or the method of matching SQL statements.
5. Optional: Specify a SET command to set the current SQLID, catalog qualifier, or schema.
6. Specify a TEST command.
   a. Provide the name of the DBRM that you want to test.
   b. Specify that the type of DBRM is a package.
   c. Provide a plan name or collection ID in qualifier.* format.
   d. Provide the name of the plan table that DB2 Path Checker reads for input.

Example

The following example job tests the current edition of the package CKPTDBRM in the collection that matches the specification ACCTPAY.* to determine which access paths changed since the most recent bind. The plan table that contains the access path information is SYSADM.PLAN_TABLE.

```
//SAMPLE06 JOB ,MSGCLASS=H,MSGLEVEL=(1,1),TIME=(,4),REGION=144M
//STEP006 EXEC PGM=CKPPTHCK
//STEPLIB DD DISP=SHR,DSN=CKP..LOADLIB
// DD DISP=SHR,DSN=DB2A.SDSNLOAD
//DBRMIN DD DISP=SHR,DSN=CKP..DBRM
//SYSPRINT DD SYSOUT=* 
//ANLOUT DD SYSOUT=* 
//SYSEXPLN DD SYSOUT=* 
//SYSCFG DD SYSOUT=* 
//RBINDOUT DD SYSOUT=* 
//PBINDOUT DD SYSOUT=* 
//SYSDIAGN DD SYSOUT=* 
//SYSIN DD * 
```
CONNECT to DB2A
OPTIONS REPORTCOSTGT REPORTCHG CPUPCT 10
SET CURRENT SQLID = 'SYSADM'
TEST DBRM CKPTDBRM AS PACKAGE ACCTPAY.*
      IN SYSADM.PLAN_TABLE
/

Related concepts:
"DB2 Path Checker features and benefits" on page 6
DB2 Path Checker helps you to identify and address performance issues that are
related to the SQL statements in your applications.
"Retrieve hints, recommendations, and expert advice from SQL Performance
Analyzer" on page 17
You can use DB2 Path Checker with IBM DB2 SQL Performance Analyzer for z/OS
for more detailed cost analysis of the SQL statements that DB2 Path Checker
selects.

Related reference:
"TEST command" on page 168
Follow the syntax and descriptions in this topic to use the TEST command.
"Job control statements" on page 174
Job control statements specify DB2 Path Checker functions and what options to
use.
"CONNECT command" on page 148
Follow the syntax and descriptions in this topic to use the CONNECT command.
"OPTIONS command" on page 154
Follow the syntax and descriptions in this topic to use the OPTIONS command.
"SET command" on page 166
Follow the syntax and descriptions in this topic to use the SET command.

Running DB2 Path Checker from the ISPF interface

DB2 Path Checker Interactive is an ISPF interface that provides online access to a
subset of DB2 Path Checker functions.

About this task

Invoke DB2 Path Checker Interactive by using the following methods:
• Running the CLIST CKP
• Selecting DB2 Path Checker Interactive from a menu

The person who installs DB2 Path Checker or the system programmer during
installation determines the method that is used at your site. When DB2 Path
Checker Interactive is invoked, the DB2 Path Checker Interactive main menu is
displayed, as shown the following figure.
Setting global options

Each time you use DB2 Path Checker Interactive, you must first specify default values for the DB2 subsystem and the authorization ID.

About this task

To specify the default values:

Procedure

1. Invoke DB2 Path Checker Interactive by using the method that is configured for your site. The DB2 Path Checker Interactive main menu is displayed.

2. Select option 0 SETTINGS to access the DB2 Path Checker Interactive - SETTINGS panel, as shown in the following figure.

3. Specify the required information in the following fields.

   **DB2 subsystem**
   Specify the default DB2 subsystem to perform DB2 Path Checker functions against.

   **Current SQLID**
   Specify the current SQLID.
Default AUTHID for PLAN_TABLEs
Specify the authorization ID to be prefixed to plan tables that are not fully qualified.

4. Press F3 to return to the main menu.

Reporting on plan table information
The DB2 Path Checker Directory report contains the timestamps for each bind of the selected plan or package (when EXPLAIN is specified for the bind).

About this task
The report provides the beginning and ending timestamps for each bind. This report is useful to help determine appropriate timestamp values to specify with the BEFORE keyword when preparing to submit a batch DB2 Path Checker job.

To generate a DB2 Path Checker Directory report:

Procedure
1. Specify option 1 REPORT on the main menu to access the DB2 Path Checker Interactive - REPORT panel, as shown in the following figure.

   ![Figure 19. DB2 Path Checker Interactive - REPORT panel](image)

   - Menu Utilities Options Help
   - Path Checker Interactive - REPORT
   - COMMAND ===> Scroll ===> CSR
   - +PLAN_TABLE .... P390H.PLAN_TABLE >
   - +Mode (PLAN or PKG) .. PKG
   - +Plan/collection .. ACCTPAYX >
   - +DBRM/package .. TEST01 >
   - Version ........ >

   Notes:
   1. Fields marked with an asterisk (*) are required.

2. If the report is for a plan DBRM or a package, specify the name of the plan table on which to run the report. Specify the plan or collection name and the DBRM or package name. These values are required. If you are running the report for a package, specifying the package version is optional.

3. Press Enter. A report using the values that were specified in Step 2 is generated, as shown in the following figure.
4. Press PF3 twice to return to the main menu.

Comparing access paths for packages
You can use the COMPARE function to compare two packages. DB2 Path Checker supports two comparison methods: normal and compare-to-previous.

About this task

A normal comparison compares the most recent bind of the package to the newest instance of the same package in the new table. A compare-to-previous comparison compares the newest instance of the specified package in the old table to the next oldest instance in the same table.

To compare packages:

Procedure
1. Select option 3 COMPPK on the main menu. The DB2 Path Checker Interactive - COMPARE (package) panel is displayed, as shown in the following figure.
2. Specify values for the COMPARE command parameters. Specify the collection name, the package name, and the old plan table name. The following fields are required:

**Collection**
Specify the name of the collection that contains the package that you want to generate a report about.

**Package**
Specify the name of the package that identifies the package that you want to generate a report about.

**Old plan table**
Specify the name of the table that contains the Explain information for the collection to be compared against.

The following fields are optional:

**New plan table**
Specify a new plan table for a normal comparison.

**Compare to previous VERSION**
Select this field to compare to a previous version.

**No match on Creator**
Select this field to exclude the creator from the access path comparison.

3. Press Enter. A report that contains the comparison results in generated, as shown in the following example:

```
Menu Utilities Options Help
-----------------------------------------------
Path Checker Interactive - COMPARE (package)
COMMAND ==>>

*Collection .................. ___________________ >
*Package .................. ______________________ >
*Old plan table ....... ____________________ >
New plan table ....... ____________________ >
Options:
  - Compare to previous VERSION
  - No match on Creator

Notes:
1. Leave "New Plan Table" blank for compare-to-previous
2. "Compare to previous VERSION" is ignored if "New Plan Table" is not blank.
3. Fields marked with an asterisk (*) are required.
```

Figure 21. DB2 Path Checker Interactive - COMPARE (package) panel
4. Press PF3 twice to return to the main menu.

Related tasks:

“Comparing access paths for different binds of the same program version” on page 78

Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

Comparing access paths for plans

You can use the COMPARE function to compare two plans. DB2 Path Checker supports two comparison methods: normal and compare-to-previous.

About this task

A normal comparison compares the most recent bind of the specified plan DBRM to the newest instance of the same plan DBRM in the new table. A compare-to-previous comparison compares the newest instance of the specified plan DBRM in the old table to the next oldest instance in the same table.

To compare plans:

Procedure

1. Select option 2 COMPPL on the main menu. The DB2 Path Checker Interactive - COMPARE (plan) panel is displayed, as shown in the following figure.
2. Specify values for the COMPARE command parameters. The following fields are required.

   **Plan** Specify the name of the plan that you want to generate a report about.

   **DBRM** Specify the name of the program that contains the SQL to be analyzed.

   **Old plan table** Specify the name of the plan table that contains the Explain information for the plan.

   The following fields are optional:

   **New plan table** Specify a new plan table for a normal comparison. Leave this field blank for a compare-to-previous comparison.

3. Press Enter. A report is generated that contains the comparison results, as shown in the following figure:

![DB2 Path Checker Interactive - COMPARE (plan) panel](image)

Figure 23. DB2 Path Checker Interactive - COMPARE (plan) panel

2. Specify values for the COMPARE command parameters. The following fields are required.

   **Plan** Specify the name of the plan that you want to generate a report about.

   **DBRM** Specify the name of the program that contains the SQL to be analyzed.

   **Old plan table** Specify the name of the plan table that contains the Explain information for the plan.

   The following fields are optional:

   **New plan table** Specify a new plan table for a normal comparison. Leave this field blank for a compare-to-previous comparison.

3. Press Enter. A report is generated that contains the comparison results, as shown in the following figure:
The report consists of a Summary section and a Detail section. The Summary section contains the access path differences. The Detail section contains the specific differences in the plan table entries, column by column.

4. Press PF3 twice to return to the main menu.

**Related tasks:**

“Comparing access paths for different binds of the same program version” on page **78**

Use the **COMPARE** command to determine which access paths changed between binds of the same version of a plan or package.

**Testing access path data for a DBRM**

You can use the **TEST** function to test access path data for a DBRM.

**Before you begin**

You must specify an SQLID in the **SETTINGS** option before you can use the **TEST** function.
About this task

To test access path data for a DBRM:

Procedure

1. Specify option 4 Test on the main menu, and press Enter. The DB2 Path Checker Interactive - TEST panel is displayed.

2. Specify values for the TEST command parameters, and press Enter. The following fields are required:

   **DBRM**
   Specify the program name that contains the SQL to be analyzed. This program name must be the same in the DBRM and in the plan or the collection that you are using for comparison. If you use wildcards, the name of the program to be tested is selected from those programs that are available in the DBRMIN concatenation. Standard TSO wildcard rules unless FROM CATALOG is specified.

   **AS parameters**
   - **Object type**
     To run the TEST command against a plan, specify P. To run the TEST command against a package, specify K.
   - **Plan/collection**
     The plan/collection is the name of the plan or the collection ID that qualifies the object.

   **IN Plan table**
   Specify the name of the plan table that contains the TEST information to compare against. If you do not specify an owner ID (high-level qualifier), the table name is automatically prefixed with the current SQLID that you specified for the SET command.

   **FROM CATALOG**
   To retrieve SQL from the catalog instead of a partitioned data set, specify Y.

The following fields are optional:

   **START WITH**
   To restart a large report from a specific program, specify a program name.

   **BEFORE**
   To limit processing to a specific subset of the rows in the plan table, specify a timestamp.

The TEST report is generated and displayed, as shown in the following example.
Generating batch jobs

You can use DB2 Path Checker Interactive to simplify the process of generating DB2 Path Checker commands and the JCL that is required to run them in batch.

The following procedures show how to generate DB2 Path Checker commands before you specify options to generate and submit batch jobs.

**Step 1. Generating DB2 Path Checker commands**

You must generate DB2 Path Checker commands before you can generate and submit the job.

**About this task**

**Important:** The command line is for TSO or ISPF commands. It is not for DB2 Path Checker commands.

To generate DB2 Path Checker commands:
Procedure

1. Specify option B \textbf{BATCH} on the main menu to access the DB2 Path Checker Interactive - BATCH panel, as shown in the following figure. The commands on the DB2 Path Checker Interactive - BATCH panel are displayed in the sequence that is required for processing. Each command is generated based on values that you specify for its parameters. Before each command is added to the command buffer, it is validated.

![DB2 Path Checker Interactive - BATCH panel](image)

Figure 26. The DB2 Path Checker Interactive - BATCH panel

2. In the Connect to SSID field, specify the ID for the DB2 subsystem that you want to connect to, and press Enter. The CONNECT command is generated.

3. Select the DB2 Path Checker commands that you want to generate, and press Enter. You can select any combination of DB2 Path Checker commands, but you must generate at least the COMPARE, EXPLAIN, REPORT, or TEST command. The corresponding DB2 Path Checker Interactive - BATCH panel for each command that you selected is displayed in sequential order.

4. Specify values for any of the following DB2 Path Checker commands that you selected, and press Enter to generate the command. Press F3 to save each command and continue, or press F12 to cancel without saving.

\textbf{OPTIONS}

For any options that you do not specify, the DB2 Path Checker default values apply.

\textbf{SET}

To generate the TEST or EXPLAIN commands, you must specify an SQLID in the \textbf{SET CURRENT SQLID} field. This SQLID must be the creator of the PLAN_TABLE that is used for the TEST or EXPLAIN commands and the creator of the application tables that are referenced by the program. An SQLID is not used for the COMPARE and REPORT commands. If you do not use the SYSIBM tables for catalog information, you must specify a catalog qualifier in the \textbf{SET CATALOG QUALIFIER} field. Setting the catalog qualifier allows Select access on the following tables:

- SYSDBRM
- SYSINDEXES
- SYSPACKAGE
- SYSPACKLIST
- SYSPACKSTMT
The following fields are required:

**Object type**
To compare against a plan, specify P. To compare against a package, specify K.

**DBRM/package**
Specify the name of the DBRM for the COMPARE command to process. TSO or ISPF rules apply to wildcard characters, where * is any number of characters and % is a single character. If you use wildcard characters, the program to be compared is selected from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE to match wildcards.

**Plan/collection**
Specify the name of the plan or the collection ID that qualifies the object.

**IN Plan table**
Specify the name of the plan table that contains the EXPLAIN information for the DBRM/package. If you do not specify an owner ID (high-level qualifier), the table name is automatically prefixed with the current SQLID that you specified for the SET command.

The following fields are optional. However, you must specify a value for the **TO Plan/collection** or the **TO Plan table** fields, or you must specify a value for the **PREVIOUS** field.

**TO parameters**

**Plan/collection**
To compare against a different plan or collection in the same or different PLAN_TABLE, specify a different plan or collection ID. DBRMs must be compared to DBRMs, and packages must be compared to packages.

**Plan table**
To compare against the same plan or collection in a different PLAN_TABLE, specify the same plan or collection ID. If you do not specify an owner ID (high-level qualifier), the table name is automatically prefixed with the CURRENT SQLID that you specified for the SET command.

**PREVIOUS**
To compare against a previous BIND_DATE in the same PLAN_TABLE as the current access path, specify Y (PREVIOUS). To compare against a previous VERSION and BIND_DATE in the same PLAN_TABLE or another combination, specify V (PREVIOUS VERSION).

**EXPLAIN DBRM**
If you specify 'D' in the field, the values that you enter in the panel apply to the **EXPLAIN DBRM** command. The following fields are required:
**DBRM**

Specify the name of the DBRM. The DBRM can be read from a PDS that has been kept from the precompiler, or it can be read from the DB2 catalog. TSO or ISPF rules apply to wildcard characters, where * is any number of characters and % is a single character.

**TO parameters**

You can use these parameters to save the output of the EXPLAIN process to a PLAN_TABLE and DSN_STATEMNT_TABLE. Specifying these parameters is equivalent to performing a BIND EXPLAIN(YES) operation without the dangers of doing the actual bind.

**Plan/collection**

Specify a different plan or collection ID to run the EXPLAIN command against a different plan or collection in the same or different PLAN_TABLE.

**Plan table**

Specify the same plan or collection ID to run the EXPLAIN command against the same plan or collection in a different PLAN_TABLE.

If you want to use the DB2 catalog as the source for the DBRM, you must specify values for the FROM CATALOG parameters, and you must specify DBRMOUT in the JCL Options.

**EXPLAIN PACKAGE**

If you specify 'K' in the field, the values that you enter in the panel apply to the EXPLAIN PACKAGE command.

**package-name**

Indicates the name of the package to explain. Wildcard characters are allowed in the name. If you use wildcard characters, standard TSO wildcard rules apply. DB2 Path Checker selects the packages to be tested from the catalog table SYSPACKAGE.

**collection-name**

Indicates the name of the collection that references the package. DB2 Path Checker selects the package to be tested from the collection in the system catalog.

**TO**

This optional clause copies the EXPLAIN data to a different plan table and statement table that you specify here. Both tables must exist under a separate schema name from the default owner, for example, qualifier.PLAN_TABLE and qualifier.DSN_STATEMNT_TABLE.

This option can be useful in situations where you want to save access path data to a plan table other than the default owner.PLAN_TABLE. The EXPLAIN PACKAGE command writes access path data to the default plan table as part of command processing. The owner of the default plan table is the owner that you specify by issuing the SET CURRENT SQLID command.

**qualifier**
Indicates the name of the schema for the specified plan table.

table-name

The name of the plan table where DB2 Path Checker copies the access path data. To prevent DB2 Path Checker from copying the data in the default plan table onto itself, this value must specify a different plan table than the default plan table sql-id.PLAN_TABLE.

VERSION

This optional clause indicates the version of the package that you want to explain. If you do not specify a version, DB2 Path Checker writes explain data for all versions of the package.

version-name

Specifies the version of the package.

If you did not specify VERSION(AUTO) as an input option to the DB2 pre-processor or co-processor, this value is a user-specified character sequence. If you used VERSION(AUTO), then the value is an automatically generated consistency token in character format (typically formatted as a time stamp, such as ’2012122222165’). Command syntax requires that you surround this value with single quotation marks.

COPYID

Specifies the bind (edition or version) for which DB2 Path Checker does explain processing. Any option for COPYID other than CURRENT produces no data unless both of the following conditions are met:

- EXTENDED PLAN MANAGEMENT is active
- EXTENDED PLAN MANAGEMENT was in effect at the time of the previous two or three binds of the package

CURRENT

Indicates the most recent bind that used any EXPLAIN option other than ONLY.

PREVIOUS

Indicates the bind immediately previous to the most recent bind that used any EXPLAIN option other than ONLY. This bind can be a previous edition of the current version or a previous version.

ORIGINAL

Indicates the oldest recorded bind that used any EXPLAIN option other than ONLY.

REPORT

The following fields are required:
Object type
To run the REPORT command against a plan, specify P. To run the REPORT command against a package, specify K.

DBRM/package
Specify the name of the DBRM. The DBRM/package is the name of the DBRM for the REPORT command to process. TSO or ISPF rules apply to wildcard characters, where * is any number of characters and % is a single character. If you use wildcard characters, the program to be tested is selected from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE to match wildcards.

Plan/collection
Specify the name of the plan or collection ID that qualifies the object.

Plan table
Specify the name of the plan table that contains the EXPLAIN information for the DBRM/package. If you do not specify an owner ID (high-level qualifier), the table name is automatically prefixed with the CURRENT SQLID that you specified for the SET command.

The following fields are optional:

START WITH
To start a report with a specific program, specify a program name.

BEFORE
To limit processing to a specific subset of the rows in the plan table, specify a timestamp in this format: 200212222222165.

DIRECTORY
To identify all the beginning and ending timestamp values for a given program, specify Y.

TEST
Before you can use the TEST command, you must successfully generate a SET command with a current SQLID. The TEST command combines the COMPARE and EXPLAIN command processes. The resulting access path report does not change the existing access path. The TEST command uses the PLAN_TABLE that is identified by the SET CURRENT SQLID value to process the EXPLAIN logic and compare the access path to the PLAN_TABLE that is specified in the IN Plan table field.

The following fields are required:

DBRM
Specify the program name that contains the SQL to be analyzed. This program name must be the same in the DBRM and in the plan or the collection that you are using for comparison. If you use wildcards, the name of the program to be tested is selected from those programs that are available in the DBRMIN concatenation. Standard TSO wildcard rules unless FROM CATALOG is specified.

AS parameters
Object type
To run the TEST command against a plan, specify P. To run the TEST command against a package, specify K.

Plan/collection
Specify the name of the plan or the collection ID that qualifies the object.

IN Plan table
Specify the name of the plan table that contains the TEST information to compare against. If you do not specify an owner ID (high-level qualifier), the table name is automatically prefixed with the current SQLID that you specified for the SET command.

The following fields are optional:

START WITH
To restart a large report from a specific program, specify a program name.

BEFORE
To limit processing to a specific subset of the rows in the plan table, specify a timestamp in this format: 2002121222421165.

FROM CATALOG
To retrieve SQL from the catalog instead of a partitioned data set, specify Y.

Results
The commands that you updated are generated, and the DB2 Path Checker Interactive - BATCH panel is displayed.

What to do next
Specify job options to generate and submit the job.

Step 2. Generating and submitting batch jobs
By using the DB2 Path Checker commands that you generated, you can generate and submit batch jobs.

About this task
To generate and submit batch jobs:

Procedure
1. Specify the data set name and the member name for the job that will be generated in the Job DSN field. The data set name must be a partitioned data set (PDS) that exists. The member will contain the job.
2. Optional: Update jobcard information, STEPLIB information, or both, and specify output data sets.
   a. Select JCL Options, and press Enter. The DB2 Path Checker Interactive - (JCL Options) panel is displayed.
   b. To update the jobcard and STEPLIB information, select the Jobcard/Steplib field. The DB2 Path Checker Interactive - (Jobcard/Steplib) panel is displayed. You can specify 1 - 3 job statements and two STEPLIB DSNs. The DB2 Path Checker loadlib is included in the job.
c. Specify output data sets for DD statements. You can specify either SYSOUT or physical data sets. PDSs must exist before you can generate the job. If a sequential data set does not exist when the JCL is created, JCL creates the data set with normal JCL options for DISP=(NEW,CATLG). Only data sets that are required for command input are validated and generated.

d. Press Enter to save your changes, and press F3 to continue. To cancel, press F12.

3. Select Generate job, and press Enter. The job is generated.

4. Optional: To edit the job, select the Edit job field, and press Enter. Normal ISPF rules apply when you edit the job. The JCL and the associated control cards are displayed.

5. Submit the job. You can submit the job in the following ways:
   - If you edited the job, issue the SUBMIT command from the command line on the EDIT panel.
   - On the DB2 Path Checker Interactive - BATCH panel, select Submit job, and press Enter.

Results

The job is submitted.

What to do next

If you want to generate a new job with different commands, select Clear for the next job, and press Enter to reset the generated commands from the previous job.

Using utility functions to manage and maintain plan tables

The Utility function provides access to the CREATE and RESET utility functions that manage and maintain plan tables.

Creating plan tables

You can use the CREATE function to create new plan tables in the current DB2 subsystem.

About this task

To reset plan tables:

Procedure

1. Specify option U UTILITY from the main menu to access the DB2 Path Checker Interactive - Utilities panel, as shown in the following figure.
2. Specify option 1 CREATE. The DB2 Path Checker Interactive - Create PLAN_TABLE panel is displayed, as shown in the following figure.

![Figure 28. DB2 Path Checker Interactive - Create PLAN_TABLE panel](image)

3. Specify the required information in the following fields.

   **Database**
   Specify an existing database.

   **Tablespace**
   Specify an existing tablespace.

   **Plan table name**
   Specify the name of the new plan table.

4. Press Enter.

**Resetting plan tables**
You can use the RESET function to delete all rows in an existing plan table and leave the table empty.

**About this task**

To reset plan tables:

**Procedure**

1. Specify option U UTILITY from the main menu to access the DB2 Path Checker Interactive - Utilities panel, as shown in the following figure.

![Figure 29. DB2 Path Checker Interactive - Utilities panel](image)

2. Specify option 2 RESET. The DB2 Path Checker Interactive - Clear PLAN_TABLE panel is displayed, as shown in the following figure.
3. In the **Plan table name** field, specify the name of the plan table to be cleared, and press Enter. All the rows in the specified plan table are deleted. A status message is displayed when the plan table is successfully cleared.

4. Press F3 twice to return to the main menu.
Chapter 5. Reporting on access paths

Many useful reports are available in DB2 Path Checker to help you identify, analyze, and take appropriate action on changed access paths.

About this task

Complete this task to generate a report by issuing a command.

The following commands generate reports:

• REPORT
• TEST
• COMPARE
• EXPLAIN
• OPTIONS

You can generate reports with the following levels of detail:

Basic reports

Basic reports write data to the SYSPRINT data set, and provide general information about input commands, summaries of whether differences were encountered, and minimal command output.

Explain reports

Explain reports write data to the SYSEXPLN data set. They include exhaustive detail about the access paths and the differences between the old and new access paths.

Change reports

Change reports write data to the SYSCHG data set. They include details about only those access paths that changed.

To write output for any but the basic reports, you must specify a ddname for SYSEXPLN (explain reports) or SYSCHG (change reports) in the JCL for the job.

The following table describes the data sets that can contain output from specific commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORT</td>
<td>SYSPRINT, SYSEXPLN data sets</td>
</tr>
<tr>
<td>TEST</td>
<td>SYSPRINT, SYSEXPLN, SYSCHG data set</td>
</tr>
<tr>
<td>COMPARE</td>
<td>SYSPRINT, SYSEXPLN, SYSCHG data set</td>
</tr>
<tr>
<td>EXPLAIN</td>
<td>SYSPRINT, SYSEXPLN, SYSCHG data set</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>SYSPRINT data set</td>
</tr>
</tbody>
</table>

Commands that examine a single plan or package (EXPLAIN and REPORT)

The REPORT command generates the following information about changed access paths:
A default summary report and a directory report in the SYSPRINT data set.

The directory report is a summary in directory format of the bind and EXPLAIN data in the specified plan table, which can be any valid DB2 plan table.

An optional detail report in the SYSEXPLN data set.

Commands that compare access paths (TEST and COMPARE)

DB2 Path Checker produces a one- or two-line report for each plan table row. If the access path for a query does not match, the data for the old access path has the numeral 1 in column 2. Column 1 contains a carriage control. The access path data for the new access path for that query has the numeral 2 in column 2.

If a query in one plan table does not have a corresponding query in the other plan table, DB2 Path Checker produces a single row with the numeral 2 in column 2.

Only one line of path information is shown when the access paths match in the two plan tables. Table rows with matching access paths are marked with an asterisk (*) and are listed only if you specify OPTIONS REPORTALL.

If the SQL is not identical, DB2 Path Checker attempts to match the query numbers. If the query numbers do not match, the mismatch is reported as an access path change.

Procedure

1. Customize the sample job to reflect your site’s standards.
2. Specify ddnames that point to the required libraries and data sets:
   - The DBRM library for DB2 Path Checker
   - The load library for DB2 for z/OS
   - A ddname for DBRMIN
3. Specify a CONNECT command to establish a session with the DB2 subsystem.
4. Optional: Specify an OPTIONS command to set processing behavior, such as the level of detail in reports or the method of matching SQL statements.
5. Optional: Specify a SET command to set the current SQLID, catalog qualifier, or schema.
6. Specify the command that you want to use to generate report information.
   For example, if you want to compare access paths for two existing binds of a package, specify a COMPARE command.

Example

The following example shows the output of the OPTIONS command in message CKP531I, which displays in the SYSPRINT data set when the command finishes processing.

<table>
<thead>
<tr>
<th>APREUSE</th>
<th>APCOMPARE</th>
<th>MATCHSQL2</th>
<th>REPORTCHG</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>NONE</td>
<td>NOSQLEQVER</td>
<td>NOSQLEQVER</td>
</tr>
<tr>
<td>MATCHCREATOR</td>
<td>NOSEQBYVERSION</td>
<td>CURRENTVERSIONONLY</td>
<td>NOMATCHVERSION</td>
</tr>
<tr>
<td>MATCHSQL2</td>
<td>DEGREE1</td>
<td>CURRENTVERSIONONLY</td>
<td>NOMATCHVERSION</td>
</tr>
<tr>
<td>CREATE TABLES</td>
<td>PKGSTABIL</td>
<td>IGNORE</td>
<td></td>
</tr>
<tr>
<td>CPUPCT*</td>
<td>N/S</td>
<td>DNUMER = 0</td>
<td>NOMATCHRC = 4</td>
</tr>
<tr>
<td>NOTFOUNDRC = 0</td>
<td>CCSID</td>
<td>37</td>
<td>COMMITCOUNT = 0</td>
</tr>
</tbody>
</table>
### How to interpret reports

The reports that DB2 Path Checker produces use column headings that match fields in the default plan table (`owner.PLAN_TABLE`).

The following descriptions of the column headings in DB2 Path Checker reports can help you to analyze changed access paths and take appropriate action. For more information about the structure of plan tables, see the *DB2 for z/OS SQL Reference*.

**AC**

Indicates the value of the ACCESSTYPE field in `owner.PLAN_TABLE`. The value describes the method of accessing the table.

**ACCESSNAME**

Indicates the name of the index when the value of the ACCESSTYPE column is 'P'.

**CREATOR**

The first column with this label indicates the SQLID of the creator of the table that the SQL statement is accessing.

**CREATOR**

The second column with this label indicates the value of the ACCESSCREATOR field from the default plan table (`owner.PLAN_TABLE`). This value indicates the creator of the index.

**FN**

Indicates the value of the COLUMN_FN_EVAL field in `owner.PLAN_TABLE`. This value indicates the status of an SQL aggregate function at different stages of its evaluation.

**IN**

Indicates whether the SQL statement was inserted or deleted since the previous edition or version of the program. Possible values are INS or DEL.

**IO**

Indicates the value of the INDEXONLY field in `owner.PLAN_TABLE`. A value of ‘Y’ indicates that the query does not have to access data, and access to an index alone is sufficient to do the step.

**JN**

Indicates the value of the JOIN_TYPE field in `owner.PLAN_TABLE`, which specifies the type of join.

**LK**

Indicates the value of the TSLOCKMOD field in `owner.PLAN_TABLE`. This value is the mode of lock that is acquired on either the new table or on its table space or table space partitions.

**M**

Indicates the value of the METHOD field from the default plan table (`owner.PLAN_TABLE`). The value is a number that indicates the JOIN method that the statement is using.
MC
Indicates the value of the MATCHCOLS field in owner.PLAN_TABLE. The value specifies the number of index keys that an index scan uses.

MJN
Indicates the value of the MERGE_JOIN_COLS field in owner.PLAN_TABLE. The value specifies the number of columns that are joined during a merge scan join (Method=2).

MTD
Indicates the value of the METHOD field in owner.PLAN_TABLE. The value is a number that indicates the join method that is used for the step.

MXSQ
Indicates the value of the MIXOPSEQ field in owner.PLAN_TABLE. The value specifies the sequence number of a step in a multiple index operation.

OP
Indicates the value of the WHEN_OPTIMIZE field in owner.PLAN_TABLE. This value provides information about when DB2 determined the access path and the role of any host variables, parameter markers, special registers, or reoptimizations.

PF
Indicates the value of the PREFETCH field in owner.PLAN_TABLE. The value determines whether prefetch reads data pages in advance.

PG
Indicates the value of the PAGE_RANGE field in owner.PLAN_TABLE. The value indicates whether the table qualifies for page range screening so that plans scan only those partitions that the query requires.

PLAN
Indicates the value of the APPLNAME field in owner.PLAN_TABLE. The value indicates the name of the application plan for the row.

PLNO
Indicates the numeric ID of the application plan for the row in owner.PLAN_TABLE.

QBNO
Indicates the value of the QBLOCKNO field in owner.PLAN_TABLE. The value is a number that identifies each query block within a query. The numbers are not in any particular order, nor are they necessarily consecutive.

QRYNO
Indicates the sequential ID of the SQL statement in the DBRM. If you hardcoded the SQL identifiers, the number might not be sequential.

SORTUJGO
This value consists of eight "Yes" or "No" flags in Y/N format, representing the contents of the following fields in owner.PLAN_TABLE:

SORTN_UNIQ
Indicates whether the new table is sorted to remove duplicate rows.

**SORTN_JOIN**

Indicates whether the new table is sorted for join method 2 or 4.

**SORTN_ORDERBY**

Indicates whether the new table is sorted for ORDER BY.

**SORTN_GROUPBY**

Indicates whether the new table is sorted for GROUP BY.

**SORTC_UNIQ**

Indicates whether the composite table is sorted to remove duplicate rows.

**SORTC_JOIN**

Indicates whether the composite table is sorted for join method 1, 2, or 4.

**SORTC_ORDERBY**

Indicates whether the composite table is sorted for an ORDER BY clause or a quantified predicate.

**SORTC_GROUPBY**

Indicates whether the composite table is sorted for a GROUP BY clause.

**TBNO**

Indicates the value of the TABNO field from the default plan table (owner.PLAN_TABLE). The value specifies the ID of the table that the SQL statement is accessing.

**TNAME**

Indicates the value of the TNAME field from the default plan table (owner.PLAN_TABLE). The value indicates the name of the table that the SQL statement is accessing.

**TNO**

Indicates the value of the TABNO field from the default plan table (owner.PLAN_TABLE). The value specifies the ID of the table that the SQL statement is accessing.

---

**RPTPKG long report**

This section includes samples of a RPTPKG long report and the JCL for the EXPLAIN sample report.

An example of a RPTPKG long report is shown in the following figure:
This example identifies the access path data for DBRM CKPTDBRM and saves the data as package ACCTPAYX.CKPTDBRM. The JCL for the EXPLAIN sample report is shown in the following figure:

SYSPRINT report

The SYSPRINT report provides the access path data from the PLAN_TABLE, estimated service unit costs, and the estimated CPU (milliseconds) from DSN_STATEMENT_TABLE, if available, and the changed data values from PLAN_TABLE.

The example SYSPRINT report in the following figure shows the difference between the most current access path and the previous access path.

The DBRM was precompiled with VERSION(AUTO) and is a single table SELECT statement.
The SYSEXPLN report provides the SQL text where available.

The example of the SYSEXPLN report in the following figure includes an English-like description of the access path, detailed list of access changes, and estimated costs.

**SYSEXPLN report**

The example of the SYSEXPLN report in the following figure includes an English-like description of the access path, detailed list of access changes, and estimated costs.

## Figure 33. SYSPRINT report example

```sql
PCK031I EXECUTING LICENSED PATH CHECK V244
11 OPTIONS CATALOGSQL REPORTCOSTGT
11 CONNECT TO DB7G
11 COMPARE PACKAGE ACCTPAYX.TEST01 IN PUBLIC.PLAN_TABLE
TO PREVIOUS
11 ACCESS PATH FOR COLLID - ACCTPAYX.TEST01 SQL ID- OLD PLAN_TABLE - RUN DATE 2006/03/16
PCK232I TIMESTAMP USED FOR PREVIOUS COMMAND = 2006022112404535 VERSION = 2005-04-11-18.20.42.913225
IN QRYNO M CREATOR TNAME TBNO AC MC CREATOR ACCESSNAME IO SORTUJOG LK PF FN QBDN PUNO MXXQ MN PG JN OP
+ 327 0 PUBLIC SYSCOLUMNS 1 I 2 PUBLIC DSNDCX01 N NNNNNNNN IS L 1 1 0
PCK220I DATA CHANGE FOR COLUMN ACCESSTYPE WAS R
PCK220I DATA CHANGE FOR COLUMN MATCHCOLS WAS 0
PCK220I DATA CHANGE FOR COLUMN ACCESSCREATOR WAS
PCK220I DATA CHANGE FOR COLUMN ACCESSNAME WAS
PCK220I DATA CHANGE FOR COLUMN PREFETCH = WAS S
PCK221I DATA CHANGE FOR COLUMN ACCESSTYPE IS NOW R
PCK221I DATA CHANGE FOR COLUMN MATCHCOLS IS NOW 0
PCK221I DATA CHANGE FOR COLUMN ACCESSCREATOR IS NOW
PCK221I DATA CHANGE FOR COLUMN ACCESSNAME IS NOW
PCK221I DATA CHANGE FOR COLUMN PREFETCH = IS NOW S
PCK224I DSN_STATEMNT_TABLE ESTIMATED COST CHANGE - OLD EST SVC UNITS 209
PCK223I DSN_STATEMNT_TABLE ESTIMATED COST CHANGE - NEW EST SVC UNITS 139
PCK206I COMPARE COMPLETE FOR PROGRAM = TEST01 VERSION = 2005-04-11-18.34.53.754785
PREVIOUS VERSION = 2005-04-11-18.20.42.913225
PCK203I STATEMENTS WITH SAME ACCESS PATH 1 STATEMENTS WITH DIFFERENT ACCESS PATH 1 FOR PROGRAM TEST01
PCK204I QUERIES WITH MATCHING EXPLAIN 2 QUERIES WITHOUT MATCHING EXPLAIN 0 FOR PROGRAM TEST01 COMPLETE RET CODE=04

Figure 34. SYSEXPLN report example

---

Chapter 5. Reporting on access paths  107
SYSCHG report

Displays only the access paths that have changed. Use this report to quickly identify which versions were compared and to check the bind time of each version.

You can also use this report to identify a package or DBRM and issue a FIND statement in the SYSPRINT report to view details of the access path changes. The reported data for this option includes the package or DBRM name, version ID, number of statements with changed access paths, bind time, explain time, and collection ID.

<table>
<thead>
<tr>
<th>DBRM</th>
<th>Version</th>
<th>No of stmts</th>
<th>Bind time</th>
<th>Explain time</th>
<th>Collid</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNDA009</td>
<td>2012-08-16-15.51.37.9154</td>
<td>1</td>
<td>2015-09-24-12.29.10.210361</td>
<td>2015-09-24-12.29.10.210361</td>
<td>BATCH</td>
</tr>
<tr>
<td>TO</td>
<td>2012-08-16-15.51.37.9154</td>
<td></td>
<td>2015-09-24-12.29.10.210361</td>
<td>2015-09-24-12.29.10.210361</td>
<td>BATCH</td>
</tr>
<tr>
<td>BNDA019</td>
<td>2012-08-16-15.51.43.9235</td>
<td>1</td>
<td>2015-09-24-12.29.10.265951</td>
<td>2015-09-24-12.29.10.265951</td>
<td>BATCH</td>
</tr>
<tr>
<td>TO</td>
<td>2012-08-16-15.51.43.9235</td>
<td></td>
<td>2015-09-24-12.29.10.265951</td>
<td>2015-09-24-12.29.10.265951</td>
<td>BATCH</td>
</tr>
<tr>
<td>BNDB009</td>
<td>2009-07-20-19.28.56.9579</td>
<td>1</td>
<td>2015-09-24-12.29.10.314899</td>
<td>2015-09-24-12.29.10.314899</td>
<td>BATCH</td>
</tr>
<tr>
<td>TO</td>
<td>2009-07-20-19.28.56.9579</td>
<td></td>
<td>2015-09-24-12.29.10.314899</td>
<td>2015-09-24-12.29.10.314899</td>
<td>BATCH</td>
</tr>
</tbody>
</table>

Figure 35. SYSCHG report example

SYSPRINT report with access path structure change

An access path structure change occurs when the tables are not referenced in the same sequence. The SYSPRINT report example, shown in the following figure, illustrates this access path structure. The report has three sections. The first four lines comprise the first section and are a summary of the structure of the old access path and the new access path. The next section identifies the old access path. The last section identifies the new access path.

There is no comparison of the data items from the access path because the structure of the access path has changed. The summary shows the different sequence for the tables accessed. The old access path referenced the SYSTABLES table first and the new access path references SYSCOLUMNS first.

<table>
<thead>
<tr>
<th>DBRM</th>
<th>Version</th>
<th>No of stmts</th>
<th>Bind time</th>
<th>Explain time</th>
<th>Collid</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNDA009</td>
<td>2012-08-16-15.51.37.9154</td>
<td>1</td>
<td>2015-09-24-12.29.10.210361</td>
<td>2015-09-24-12.29.10.210361</td>
<td>BATCH</td>
</tr>
<tr>
<td>TO</td>
<td>2012-08-16-15.51.37.9154</td>
<td></td>
<td>2015-09-24-12.29.10.210361</td>
<td>2015-09-24-12.29.10.210361</td>
<td>BATCH</td>
</tr>
<tr>
<td>BNDA019</td>
<td>2012-08-16-15.51.43.9235</td>
<td>1</td>
<td>2015-09-24-12.29.10.265951</td>
<td>2015-09-24-12.29.10.265951</td>
<td>BATCH</td>
</tr>
<tr>
<td>TO</td>
<td>2012-08-16-15.51.43.9235</td>
<td></td>
<td>2015-09-24-12.29.10.265951</td>
<td>2015-09-24-12.29.10.265951</td>
<td>BATCH</td>
</tr>
<tr>
<td>BNDB009</td>
<td>2009-07-20-19.28.56.9579</td>
<td>1</td>
<td>2015-09-24-12.29.10.314899</td>
<td>2015-09-24-12.29.10.314899</td>
<td>BATCH</td>
</tr>
<tr>
<td>TO</td>
<td>2009-07-20-19.28.56.9579</td>
<td></td>
<td>2015-09-24-12.29.10.314899</td>
<td>2015-09-24-12.29.10.314899</td>
<td>BATCH</td>
</tr>
</tbody>
</table>

Figure 36. SYSPRINT report example with access path structure change

History table

The History table for a summary of changes detected in DB2 Path Checker processing contains the same information as the SYSCHG ddname report.
The DDL for the History table is shown in the following figure:

```sql
CREATE TABLE HIST_TABLE

(APPLNAME_OLD CHAR(8) NOT NULL,
 APPLNAME_NEW CHAR(8) NOT NULL,
 COLLID_OLD VARCHAR(128) NOT NULL WITH DEFAULT,
 COLLID_NEW VARCHAR(128) NOT NULL WITH DEFAULT,
 VERSION_OLD VARCHAR(64) NOT NULL WITH DEFAULT,
 VERSION_NEW VARCHAR(64) NOT NULL WITH DEFAULT,
 QBLOCK_TYPE_OLD CHAR(6) NOT NULL WITH DEFAULT,
 QBLOCK_TYPE_NEW CHAR(6) NOT NULL WITH DEFAULT,
 QUERYNO_OLD INTEGER NOT NULL WITH DEFAULT,
 QUERYNO_NEW INTEGER NOT NULL WITH DEFAULT,
 PROCSU_OLD INTEGER NOT NULL WITH DEFAULT,
 PROCSU_NEW INTEGER NOT NULL WITH DEFAULT,
 PROCMS_OLD INTEGER NOT NULL WITH DEFAULT,
 PROCMS_NEW INTEGER NOT NULL WITH DEFAULT,
 CHANGE_REASON CHAR(20) NOT NULL WITH DEFAULT,
 COMPARE_DATE TIMESTAMP NOT NULL WITH DEFAULT,
 COMPARE_USER CHAR(8) NOT NULL WITH DEFAULT
) IN DSNDB04.HIST9H01;
```

Figure 37. History table DDL

The following example shows the Control Card for the History table:

```sql
OPTIONS HISTORY TABLE P390H.HIST_TABLE
```

The following example shows the work file to collect the data:

```sql
//HISTWORK DD DSN=P390H.TEST.HISTWORK,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(TRK,(30,30)),
// DCB=(RECFM=VB,LRECL=862,BLKSIZE=0)
```

If the HISTWORK work file ddname exists and the OPTIONS HISTORY TABLE is not specified, the data is written to the work file and is not inserted to the table.

The data can be loaded with the following information:

```sql
LOAD DATA LOG YES RESUME YES INDDN SYSREC00 INTO TABLE
 P390H.HIST_TABLE

(APIPLNAME_OLD CHAR(8),
 APPLNAME_NEW CHAR(8),
 COLLID_OLD VARCHAR,
 COLLID_NEW VARCHAR,
 VERSION_OLD VARCHAR(64),
 VERSION_NEW VARCHAR(64),
 QBLOCK_TYPE_OLD CHAR(6),
 QBLOCK_TYPE_NEW CHAR(6),
 QUERYNO_OLD INTEGER,
 QUERYNO_NEW INTEGER,
 PROCSU_OLD INTEGER,
 PROCSU_NEW INTEGER,
 PROCMS_OLD INTEGER,
 PROCMS_NEW INTEGER,
 CHANGE_REASON CHAR(20),
 COMPARE_DATE TIMESTAMP,
 COMPARE_USER CHAR(8)
) IN DSNDB04.HIST9H01;
```
Remember: The user ID from the run and the timestamp for the comparison are defaulted from the SQL insert so the load file does not contain these.

Compare access paths for two existing binds

The example JCL shown in the following figure compares current access paths for all the programs in CKPTCOLL collection.

CKPTCOLL is a test collection that only contains one package CKPTDBRM. The report shows one new SQL statement (363) and one changed access path where the QUERYNO changed from 369 to 379. The report contains both the SYSPRINT report and the detailed SYSEXPLN report.

```sql
/* ALL LINES THAT START WITH */
/* ARE COMMENTS */
/* THIS TEST WILL REPORT ON */
/* ALL THE PACKAGES IN */
/* COLLECTION */
/* ACCTPAK AND ALL DBRMS IN PLAN CUSTOMER */
/* CONNECT TO DSN1 */
/* THIS TEST WILL COMPARE ON */
/* ALL THE PACKAGES IN */
/* COLLECTION */
/* ACCTPAK WITH SAVED DATA IN PLAN_TABLE2 */
/* COMPARE PACKAGE CKPTCOLL.CKPTDBRM IN P390H.PLAN_TABLE */
/* TO CKPTCOLL.* IN P390H.PLAN_TABLE2 */
```

Figure 38. JCL for comparing access paths for two existing binds

The report for this example is shown in the following figure.
--- EXECUTING LICENSED PATH CHECKER V2R1 ---

-- ALL LINES THAT START WITH -- ARE COMMENTS

-- THIS TEST WILL REPORT ON ALL THE PACKAGES IN COLLECTION

-- ACCTPAY AND ALL DBRMS IN PLAN CUSTOMER

CONNECT TO DSN1

-- THIS TEST WILL COMPARE ON ALL THE PACKAGES IN COLLECTION

-- ACCTPAY WITH SAVED DATA IN PLAN_TABLE2

COMPARE PACKAGE CKPTCOLL.CKPTDBRM IN P390H.PLAN_TABLE

TO CKPTCOLL.* IN P390H.PLAN_TABLE2

2003/08/08 SQL ID - ACCESS PATH FOR CKPTDBRM COLLID - CKPTCOLL OLD PLAN_TABLE -

STATEMENT 363

STEP 1 ACCESSES TABLE SYSIBM.SYSTABLES USING INDEX SYSIBM.DSNDTX01 USING 1 COLUMN(S).

THE ACCESS WILL USE LIST PREFETCH

THE INTENT LOCK FOR THE TABLE IS IS

THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260923

THE TABLE HAS 332 ROWS OF 570 BYTES.

THE UNIQUE INDEX HAS 0 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 61

STATEMENT 369

THIS IS THE OLD ACCESS PATH.

STEP 1 ACCESSES TABLE SYSIBM.SYSINDEXES USING MULTIPLE INDEX ACCESS (MX).

THE ACCESS WILL USE LIST PREFETCH

THE INTENT LOCK FOR THE TABLE IS IS

THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260937

THE TABLE HAS 309 ROWS OF 490 BYTES.

MX STEP 1 USES INDEX SYSIBM.DSNDXX02 USING 1 COLUMN(S).

THE ACCESS ONLY READS THE INDEX NOT THE TABLE

THE INTENT LOCK FOR THE TABLE IS IS

THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260937

THE TABLE HAS 309 ROWS OF 490 BYTES.

THE NON-UNIQUE INDEX HAS 0 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 65

MX STEP 2 USES INDEX SYSIBM.DSNDXX01 USING 1 COLUMN(S).

THE ACCESS ONLY READS THE INDEX NOT THE TABLE

THE INTENT LOCK FOR THE TABLE IS IS

THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260937

THE TABLE HAS 309 ROWS OF 490 BYTES.

THE UNIQUE INDEX HAS 0 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 73

MX STEP 3 WILL UNION THE RID LISTS

THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260937

THE TABLE HAS 309 ROWS OF 490 BYTES.

STEP 2 ACCESSES TABLE SYSIBM.SYSTABLES USING INDEX SYSIBM.DSNDTX01 USING 2 COLUMN(S).

THE JOIN METHOD IS NESTED

Figure 39. Report for comparing access paths for two existing binds
The example in the following figure shows the SYSPRINT report for the current access path for program CKPTDBRM in collection CKPTCOLL. The program was not compiled with a version identifier.

As shown in the following figure, the MAKE command sets the current access path for QUERYNO 312 as HINT with the name of PROD. The HINT must be processed with a bind to make it active.
CKP031I EXECUTING LICENSED PATH CHECKER V2R1

-- ALL LINES THAT START WITH -- ARE COMMENTS
-- THIS TEST WILL REPORT ON DBRM CKPTDBRM IN COLLECTION
-- CKPTCOLL
-- CONNECT TO DSN1

CONNECT TO DSN1
SET CATALOG QUALIFIER = 'SYSIBM'
REPORT ON PACKAGE CKPTCOLL.CKPTDBRM IN P390H.PLAN_TABLE2
-- MAKE STATEMENT 312 FROM PLAN_TABLE2 BE HINT

2003/08/09 SQL ID - ACCESS PATH FOR CKPTDBRM COLLID - CKPTCOLL OLD PLAN_TABLE -

| IN QRYNO M | CREATOR TNAME | TBN0 AC MC CREATOR ACCESSNAME ID SORTUJGO LK PF Q9RO PLNO MXXQ |
|-----------|---------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| 287 0     | SYSIBM        | SYSCOLUMNS        | 1 I 0 SYSIBM      | DSNDCX01          | N NNNNNNNN IS S 1 1 |
| 293 0     | SYSIBM        | SYSCOLUMNS        | 1 I 2 SYSIBM      | DSNDCX01          | N NNNNNNNN IS S 1 1 |
| 293 1     | SYSIBM        | SYSKEYS           | 4 I 3 SYSIBM      | DSNDCX01          | Y NNNNNNNN IS S 3 1 |
| 293 1     | SYSIBM        | SYSTABLES         | 2 I 2 SYSIBM      | DSNDCX01          | N NNNNNNNN IS S 4 1 |
| 312 0     | SYSIBM        | SYSTABLES         | 2 I 0 SYSIBM      | DSNDCX01          | N NNNNNNNN IS S 1 1 |
| 312 1     | SYSIBM        | SYSTABLES         | 2 I 2 SYSIBM      | DSNDCX01          | N NNNNNNNN IS L 1 2 |
| 323 0     | SYSIBM        | SYSTABLES         | 1 I 0 SYSIBM      | DSNDCX01          | N NNNNNNNN N S 1 1 |
| 338 0     | SYSIBM        | SYSTABLES         | 1 I 1 SYSIBM      | DSNDCX01          | N NNNNNNNN L S 1 1 |
| 344 0     | SYSIBM        | SYSTABLES         | 1 I 1 SYSIBM      | DSNDCX01          | N NNNNNNNN L S 1 1 |
| 344 1     | SYSIBM        | SYSCOLUMNS        | 1 I 2 SYSIBM      | DSNDCX01          | Y NNNNNNNN IS S 1 2 |
| 344 1     | SYSIBM        | SYSCOLUMNS        | 2 I 2 SYSIBM      | DSNDCX01          | Y NNNNNNNN IS S 1 2 |
| 354 1     | SYSIBM        | SYSCOLUMNS        | 2 I 0 SYSIBM      | DSNDCX01          | N NNNNNNNN IS S 1 1 |
| 354 1     | SYSIBM        | SYSCOLUMNS        | 2 I 2 SYSIBM      | DSNDCX01          | N NNNNNNNN IS S 1 2 |
| 369 0     | SYSIBM        | SYSINDEXES        | 1 M 0             | N NNNNNNNN IS L 1 1 |
| 369 0     | SYSIBM        | SYSINDEXES        | 1 MX 1 SYSIBM     | DSNDCX02          | Y NNNNNNNN IS S 1 1 1 |
| 369 0     | SYSIBM        | SYSINDEXES        | 1 MX 1 SYSIBM     | DSNDCX01          | Y NNNNNNNN IS S 1 1 2 |
| 369 0     | SYSIBM        | SYSINDEXES        | 1 MX 1 SYSIBM     | DSNDCX01          | Y NNNNNNNN IS S 1 1 3 |

CKP206I REPORT COMPLETE FOR PROGRAM = CKPTDBRM VERSION =

MAKE STATEMENT 312 WITH TIMESTAMP '2003080821360561'
IN PACKAGE CKPTCOLL.CKPTDBRM IN P390H.PLAN_TABLE2
BE HINT PROD

NUMBER OF ROWS UPD 02
COMPLETE RET CODE=00

2003/08/09 SQL ID - ACCESS PATH FOR CKPTDBRM COLLID - CKPTCOLL OLD PLAN_TABLE -

STATEMENT 287 FROM REPORT.
STEP 1 ACCESSES TABLE SYSIBM .SYSCOLUMNS
USING INDEX SYSIBM .DSNDCX01 USING 0 COLUMN(S).
THE ACCESS WILL USE SEQUENTIAL PREFETCH
THE INTENT LOCK FOR THE TABLE IS IS
THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260561
THE TABLE HAS 4,669 ROWS OF 981 BYTES.
THE UNIQUE INDEX HAS 0 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 83

STATEMENT 293 FROM REPORT.
STEP 1 ACCESSES TABLE SYSIBM .SYSINDEXES
USING INDEX SYSIBM .DSNDCX03 USING 0 COLUMN(S).
THE ACCESS WILL USE SEQUENTIAL PREFETCH
THE INTENT LOCK FOR THE TABLE IS IS
THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260803
THE TABLE HAS 309 ROWS OF 490 BYTES.
THE UNIQUE INDEX HAS 0 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 74

STATEMENT 293 FROM REPORT.
STEP 2 ACCESSES TABLE SYSIBM .SYSCOLUMNS
USING INDEX SYSIBM .DSNDCX01 USING 2 COLUMN(S).
THE JOIN METHOD IS NESTED LOOP
THE INTENT LOCK FOR THE TABLE IS IS
THE TIMESTAMP FOR THIS EXPLAIN IS 2003080821260803
THE TABLE H

Figure 41. MAKE command sample
The example in the following figure shows two TEST requests in a single job.

The requests test the access paths for DBRM CKPTDBRM against the previous binds. The first request checks access paths for collection ACCTPAY, and the second request checks access paths for the same DBRM in plan CUSTOMER.

Both reports in Figure 43 on page 115 show a new SQL statement and a change in access path for statements with old QUERYNO 369 and new QUERYNO 379.

```
//P390HT3 JOB ,TSCICS,CLASS=A,MSGCLASS=R,MSGLEVEL=(1,1),
// NOTIFY=&SYSUID,REGION=20M
//STEP009 EXEC PGM=CKPTDBRM
//STEPLIB DD DSN=db2.SDSNLOAD,DISP=SHR
// DD DSN=ckp.SCKPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSEXPLN DD SYSOUT=*
//SYSPRNT DD SYSOUT=* 
//DBRMIN DD DSN=ckp.SCKPDBRM,DISP=SHR
//SYSIN DD * 
-- THIS TEST WILL REPORT ON PROGNAME CKPTDBRM IN COLLECTION 
-- ACCTPAY AND PROGNAME CKPTDBRM IN PLAN CUSTOMER
-- TESTBIND PROCESSING REQUIRES SQLID SET TO CORRECT QUALIFIER
-- HIGH LEVEL ID TO RESOLVE SQL STATEMENTS AND PLAN_TABLE FOR
-- SAME ID
  CONNECT TO DSN1
  OPTIONS REPORTCHG MATCHQUERYNO
  SET CURRENT SQLID = 'P390H9'
  TEST DBRM CKPTDBRM AS PACKAGE ACCTPAY.*
  IN P390H9.PLAN_TABLE
  OPTIONS REPORTALL
  TEST DBRM CKPTDBRM AS PLAN CUSTOMER.*
  IN P390H9.PLAN_TABLE
```

Figure 42. JCL for TEST example

The report for this example is shown in the following figure. The access path rows that match are included and identified by an asterisk (*) because of the REPORTALL parameter.
1

CKP031I EXECUTING LICENSED PATH CHECKER V2R1
11 -- ALL LINES THAT START WITH -- ARE COMMENTS
-- THIS TEST WILL EVALUATE THE ACCESS PATHS FOR THE SAME DBRM
-- AS BOTH A PACKAGE AND PLAN
CONNECT TO DSN1
11 SET CATALOG QUALIFIER = 'SYSIBM'
-- TESTBIND PROCESSING REQUIRES SQLID SET TO CORRECT QUALIFIER
-- HIGH LEVEL ID TO RESOLVE SQL STATEMENTS AND PLAN_TABLE FOR
-- SAME ID
11 SET CURRENT SQLID = 'P390H'
11 TEST DBRM CKPTDBRM AS PACKAGE CKPTCOLL.*
11 IN P390H..PLAN_TABLE2
12 2003/08/08 SQL ID - ACCESS PATH FOR CKPTDBRM COLLID - CKPTCOLL OLD PLAN_TABLE - PLAN_TABLE2
IN QRYNO M CREATOR TNAME TBNO AC MC CREATOR ACCESSNAME IO SORTUJGO LK PF QBNO PLNO MXSQ
2 363 0 SYSIBM SYSTABLES 1 I 1 SYSIBM DSNDTX01 N NNNNNNNN IS L 1 1
1 369 0 SYSIBM SYSINDEXES 1 M 0 N NNNNNNNN IS L 1 1
1 369 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX02 Y NNNNNNNN IS 1 1 1
1 369 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX01 Y NNNNNNNN IS 1 1 2
1 369 1 SYSIBM SYSTABLES 2 I 2 SYSIBM DSNDTX01 Y NNNNNNNN IS 1 2
2 379 0 SYSIBM SYSINDEXES 1 M 0 N NNNNNNNN IS L 1 1
2 379 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX02 Y NNNNNNNN IS 1 1 1
2 379 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX01 Y NNNNNNNN IS 1 1 2
2 379 0 SYSIBM SYSINDEXES 1 M 0 N NNNNNNNN IS L 1 1
2 379 1 SYSIBM SYSTABLES 2 I 2 SYSIBM DSNDTX01 Y NNNNNNNN IS 1 2
CKP206I TEST COMPLETE FOR PROGRAM = CKPTDBRM VERSION =
CKP203I STATEMENTS WITH SAME ACCESS PATH 9 STATEMENTS WITH DIFFERENT ACCESS PATH 1 FOR PROGRAM CKPTDBRM
CKP204I QUERIES WITH MATCHING EXPLAIN 10 QUERIES WITHOUT MATCHING EXPLAIN 1 FOR PROGRAM CKPTDBRM
11 TEST DBRM CKPTDBRM AS PLAN CKPTPLAN.*
11 IN P390H..PLAN_TABLE2
12 2003/08/08 SQL ID - ACCESS PATH FOR CKPTDBRM APPLNAME CKPTPLAN LD PLAN_TABLE - PLAN_TABLE2
IN QRYNO M CREATOR TNAME TBNO AC MC CREATOR ACCESSNAME IO SORTUJGO LK PF QBNO PLNO MXSQ
2 363 0 SYSIBM SYSTABLES 1 I 1 SYSIBM DSNDTX01 N NNNNNNNN IS L 1 1
1 369 0 SYSIBM SYSINDEXES 1 M 0 N NNNNNNNN IS L 1 1
1 369 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX02 Y NNNNNNNN IS 1 1 1
1 369 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX01 Y NNNNNNNN IS 1 1 2
1 369 1 SYSIBM SYSTABLES 2 I 2 SYSIBM DSNDTX01 Y NNNNNNNN IS 1 2
2 379 0 SYSIBM SYSINDEXES 1 M 0 N NNNNNNNN IS L 1 1
2 379 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX02 Y NNNNNNNN IS 1 1 1
2 379 0 SYSIBM SYSINDEXES 1 MX 1 SYSIBM DSNXX01 Y NNNNNNNN IS 1 1 2
2 379 0 SYSIBM SYSINDEXES 1 MI 0 N NNNNNNNN IS L 1 1
2 379 1 SYSIBM SYSTABLES 2 I 2 SYSIBM DSNDTX01 Y NNNNNNNN IS 1 2
CKP206I TEST COMPLETE FOR PROGRAM = CKPTDBRM VERSION =
CKP203I STATEMENTS WITH SAME ACCESS PATH 9 STATEMENTS WITH DIFFERENT ACCESS PATH 1 FOR PROGRAM CKPTDBRM
CKP204I QUERIES WITH MATCHING EXPLAIN 10 QUERIES WITHOUT MATCHING EXPLAIN 1 FOR PROGRAM CKPTDBRM
COMPLETE RET CODE= 04
11 2003/08/08 SQL ID - ACCESS PATH FOR CKPTDBRM COLLID - CKPTCOLL OLD PLAN_TABLE - PLAN_TABLE2
DBRM CKPTDBRM HAS A CONNECT TOKEN OF 1727F0280E8AA0CC
STATEMENT# 287 DECLARE SELECT-1 CURSOR FOR SELECT NAME , TBNAME , COLNO , C
OLTYPE , LENGTH FROM SYSIBM . SYSCOLUMNS WHERE TBNAME = : H
STATEMENT# 293 DECLARE SEL
Command language examples

The following command language examples show some typical examples of using the command language for DB2 Path Checker.

The commands are entered from the SYSIN data set.

A sample of the JCL that you can use to run DB2 Path Checker with commands supplied in SYSIN is shown in the following figure. Modify the JCL to work with your installation configuration.

```
//STEP002 EXEC PGM=CKPPTHCK
//STEPLIB DD DSN=ckp.SCKPLOAD,DISP=SHR
// DSN=db2.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSEXPLN DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//DBRMIN DD DSN=ckp.SCKPDBRM,DISP=SHR  
//SYSSIN DD *  
-- ALL LINES THAT START WITH -- ARE COMMENTS  
-- ENTER YOUR COMMANDS HERE  
```

Figure 44. JCL for specifying commands in SYSIN

Command language example 1

The following request produces a report on the access path for program ACCTPAY.ORDERS from the data in the specified plan table.

REPORT ON PACKAGE ACCTPAY.ORDERS IN P390H.PLAN_TABLE

Command language example 2

The following example produces a report on the most current versions of all the programs for the collection ACCTPAY and the most current access path in the plan table. The START WITH clause starts the report at the first program name whose name is greater than or equal to A.

REPORT ON PACKAGE ACCTPAY.* IN P390H.PLAN_TABLE START WITH A

Command language example 3

The following example produces a report on all the program names that begin with AA in plan CUSTOMER. The START WITH clause is specified correctly, but all the program names that begin with AA would be greater than A in any case.

REPORT ON PLAN CUSTOMER.AA* IN P390H.PLAN_TABLE START WITH A

Command language example 4

The first collection and plan table are considered the new access path and the second collection and plan table are the old access path. Either the second user ID, plan table, or collection ID identifies the old access path. The target (second) program name for TEST and COMPARE requests is always an asterisk (*) because the same program name is used for both sides of the comparison.

COMPARE PACKAGE ACCTPAY.ORDERS IN userid.PLANTABLE TO ACCTPAY.* IN userid.PLANTABLE2
Command language example 5

The following request compares all the package names (program names) in collection ACCTPAY as if they had previously been bound into collection SAVECOLL. The DBRM name can be specified with wildcards, and the list of names to be processed will be selected from SYSPACKAGE. This request will start with program name ORDERS.

```
COMPARE PACKAGE ACCTPAY.* IN userid.PLANTABLE TO SAVECOLL.* IN userid.PLANTABLE START WITH ORDERS
```

Command language example 6

The first plan name and plan table are considered the new access path and the second plan name and plan table are the old access path. The following request compares the access paths for program ORDERS to data in the plan table as if the plan had been bound with the name TESTPLN2.

```
COMPARE PLAN CUSTOMER.ORDERS IN userid.PLANTABLE TO TESTPLN2.* IN userid.PLANTABLE
```

Command language example 7

The following request compares the access paths for all the programs in plan CUSTOMER to data in the plan table as if the plan had been bound with the name TESTPLN2. The list of names to be processed will be determined from SYSIBM.SYSDBRM.

```
COMPARE PLAN CUSTOMER.* IN userid.PLANTABLE TO TESTPLN2.* IN userid.PLANTABLE START WITH A
```

Command language example 8

TEST processing requires that the SQLID be set to the qualifier of the tables to be processed.

The following request reads the DBRM ORDERS from ddname DBRMIN, creates an Explain for each access path, and compares it to the previous access path. This will identify potential access path changes.

```
SET CURRENT SQLID = 'PROD'
TEST DBRM ORDERS AS PACKAGE COLLNAME.* IN userid.PLANTABLE
```

Command language example 9

The DBRM name can be specified with wildcards and the list of names to be processed will be determined from SYSIBM.SYSPACKAGE.

The following request processes all the programs in the collection COLLNAME starting with program NAXX. The START WITH clause is optional.

```
SET CURRENT SQLID = 'PROD'
TEST DBRM * AS PACKAGE COLLNAME.* IN userid.PLANTABLE START WITH NAXX
```

Command language example 10

The following report reads the DBRM ORDERS from ddname DBRMIN, Explain each access path, and compares it to the previous access path. This will identify potential access path changes.

```
SET CURRENT SQLID = 'PROD'
TEST DBRM ORDERS AS PLAN CUSTOMER.* IN userid.PLANTABLE
```
Command language example 11

The DBRM name can be specified with wildcards, and the list of names to be processed will be determined from SYSIBM.SYSDBRM. The following request processes all the program names that start with TEST from plan CUSTOMER.

```
SET CURRENT SOLID = 'PROD'
TEST DBRM TEST* AS PLAN CUSTOMER.* IN userid.PLANTABLE
```

Command language example 12

The following request assigns the optimization hint PROD to a statement in package PROGNAME of collection ACCTPAY. The timestamp and statement number uniquely identify the plan table row to be modified by the addition of a hint.

```
MAKE STATEMENT 134 WITH TIMESTAMP = '2001012311210955' IN PACKAGE ACCTPAY.PROGNAME IN userid.PLAN_TABLE BE HINT PROD
```
Chapter 6. Troubleshooting

Use these topics to diagnose and correct problems that you experience with DB2 Path Checker.

Recovery procedures

DB2 Path Checker runs as a normal DB2 application with updates to PLAN_TABLE and DSN_STATEMNT_TABLE for the TEST and EXPLAIN commands. You can recover from any failures by restarting the program.

Messages and codes

These topics describe the error messages and codes that are issued by DB2 Path Checker.

All DB2 Path Checker diagnostic information and messages are sent to SYSPRINT. DB2 Path Checker messages are issued from module CKPPTHCK.

Return codes

When DB2 Path Checker processing completes, a 0, 4, 44, or 16 code is returned. These return codes are explained in the following table.

<table>
<thead>
<tr>
<th>Return code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DB2 Path Checker processing completed normally. There were no changes to access paths.</td>
</tr>
<tr>
<td>4 or 44</td>
<td>DB2 Path Checker processing completed normally. There were changes to access paths. If you specify the ANLOUT DD statement and there are access path changes, the normal return code is 44. You can then run SQL Performance Analyzer.</td>
</tr>
<tr>
<td>16</td>
<td>DB2 Path Checker encountered a severe error. Processing did not complete normally. Contact IBM Software Support.</td>
</tr>
</tbody>
</table>

Messages

DB2 Path Checker messages follow a specified format with a three-letter code and a message number.

DB2 Path Checker messages follow the format of CKPnmt, where nnn is the message number and t is a one-character suffix that indicates the severity of the message: I for informational messages, W for warnings, E for errors, and S for severe errors.

- **CKP031I** A licensed copy of DB2 Path Checker is running.
  - **Explanation:** This message indicates that the instance of DB2 Path Checker that is running has a valid license.
  - **User response:** No action is required.

- **CKP035E** DB2 Path Checker encountered the following problem with the data set that the DBRMIN DD statement points to: additional-message
  - **Explanation:** One of the following error conditions occurred:
CKP036E • CKP044E

The program cannot open the data set.

DB2 Path Checker could not open the data set for DBRMIN DD that is specified in the job control statements for CKPPTHCK.

The list of members in the DBRMLIB is too large to process.

DB2 Path Checker could not process the list of members in the partitioned data set (PDS) that contains the DBRM library (DBRMLIB) because it has too many members.

The program cannot close the data set.

DB2 Path Checker could not close the data set for DBRMIN DD that is specified in the job control statements for CKPPTHCK.

User response: Verify that the DBRMIN DD card is present in the JCL and that it refers to a valid partitioned data set in the DBRM library (DBRMLIB). If too many members exist in the DBRMLIB, consider raising the REGION size in the job card.

CKP036E The ddname DBRMIN is missing.

Explanation: The TEST command was issued without the FROM CATALOG parameter. When the program attempted to read the DBRM, the ddname DBRMIN was missing.

User response: Either specify FROM CATALOG to read the input from the DB2 catalog, or add a ddname for DBRMIN that points to the appropriate DBRM library (DBRMLIB).

CKP038E The ddname for SYSIN is missing.

Explanation: The PARM field in the job control statement for CKPPTHCK did not specify the type of processing, and no SYSIN DD statement was present to specify a data set that contains command input for processing.

User response: Specify a process request in the PARM field, or specify a ddname for SYSIN to provide command input.

CKP039E DB2 Path Checker encountered an installation error that prevented the program from running.

Explanation: DB2 Path Checker could not initialize or process successfully.

User response: Review the output from all installation jobs (including SMP/E jobs). Correct any errors, verify that DB2 Path Checker is using the correct target library, and run the affected jobs again. If the problem persists, contact IBM Software Support.

---

CKP041E A TEST command failed because it first requires a SET CURRENT SQLID statement.

Explanation: The job issued a TEST command before processing the SET CURRENT SQLID statement. The job must process a valid SET CURRENT SQLID statement before it issues a TEST command.

User response: Add a valid SET CURRENT SQLID statement to the job prior to issuing the TEST command that failed, and then rerun the job.

CKP042E You cannot run commands without first issuing a CONNECT TO command that specifies a DB2 subsystem.

Explanation: A CONNECT TO command that specifies a DB2 system is required before you can issue other commands.

User response: Add a valid CONNECT TO command and ensure that it is the first command in the job.

CKP043E DB2 Path Checker could not process a Unicode DBRM because no MVS conversion services for UTF-8 and UTF-16 were available.

Explanation: DB2 Path Checker attempted to process a DBRM from a Unicode system on a non-Unicode system. Processing continues, but DB2 Path Checker ignores this SQL statement.

User response: Ask your system programmer or technical support group to verify that MVS Unicode conversion services are installed on the MVS image. They can do this by finding the current FMID for the services and querying the SMP/E target zone. If the problem persists, contact IBM Software Support.

CKP044E DB2 Path Checker failed to create the temporary DBRM for a TEST or EXPLAIN command.

Explanation: The DB2 Path Checker program issued a TEST or EXPLAIN command that did not contain a ddname for the DBRMOUT temporary data set. When you use the FROM CATALOG parameter with these commands, you must specify this ddname. The ddname points to the data set that DB2 Path Checker uses to write the information that it requires to rebuild the DBRM from the catalog.

DB2 Path Checker ignores this command and attempts to process the next request.

For more information about the DBRMOUT statement and the required properties of the data set that it points to, see [Job control statements](#) on page 174.

User response: Verify that the DBRMOUT DD card is present in the JCL and that it refers to a valid partitioned data set in the DBRM library (DBRMLIB). If the problem persists, contact IBM Software Support.
CKP045E  DB2 Path Checker could not build a DBRM because it could not find SQL statements in the DB2 catalog. RC: return-code.

Explanation: DB2 Path Checker was not able to retrieve the SQL statements that it needed to build a DBRM. The SQL statements were not found in the catalog. In the message text, return-code indicates an internal diagnostic code that is for the use of IBM Software Support.

This message typically indicates an installation error.

User response: Review the output from all installation jobs (including SMP/E jobs). Correct any errors, verify that DB2 Path Checker is using the correct target library, and run the affected jobs again. If the problem persists, contact IBM Software Support.

CKP046I  The DBRM dbrm-name was not found in the collection collection-id. The DBRM dbrm-name was not found in the plan plan-name.

Explanation: A DBRM that was specified in a TEST FROM CATALOG command is not in the catalog.

Only one of the sentences in the message text is displayed, depending on whether DB2 Path Checker was searching in a collection or a plan.

User response: Correct the request to point to a DBRM that is in the catalog.

CKP047E  DB2 Path Checker did not process command-or-program-name because the command or program requires a specification for a creator and a table name. CREATOR: table-qualifier TABLE NAME: table-name

Explanation: A required plan table or catalog table does not exist. In one scenario, command-or-program-name is one of the following user-issued commands:

• TEST
• EXPLAIN
• COMPARE
• REPORT

In a different scenario, command-or-program-name is CKPPTHCK, the main DB2 Path Checker program.

If command-or-program-name is TEST, EXPLAIN, COMPARE, or REPORT

DB2 Path Checker did not attempt to process the specified command because it requires a valid plan table. In this case, DB2 Path Checker requires a valid SQLID or schema that qualifies the plan table name. Because the OPTIONS CREATE TABLES keyword was not specified, DB2® Path Checker could not create a temporary table dynamically.

If command-or-program-name is CKPPTHCK

DB2 Path Checker could not find a DB2 catalog table by using the current catalog qualifier.

Command variables

command-or-program-name

Indicates either the command name or CKPPTHCK.

table-qualifier

Indicates the current creator, whether it is an SQLID or catalog qualifier.

table-name

Indicates the name of the plan table or catalog table.

User response: If command-or-program-name is TEST, EXPLAIN, COMPARE, or REPORT, perform one of the following actions:

• Specify the OPTIONS CREATE TABLES keyword and issue the SET CURRENT SQLID command, and then resubmit the job.
• Manually create the plan table and resubmit the job.

If command-or-program-name is CKPPTHCK, issue the SET CATALOG QUALIFIER command, and then resubmit the job.

CKP048I  DB2 Path Checker did not provide cost estimates because it requires a DSN_STATEMNT_TABLE to calculate run-time costs. CREATOR: table-qualifier TABLE NAME: table-name

Explanation: A TEST, EXPLAIN, or REPORT command requires a DSN_STATEMNT_TABLE to report estimated runtime costs. DB2 Path Checker processes the command, but does not report estimated costs.

table-qualifier

Indicates the current SQL ID or schema.

table-name

Indicates the name of the DSN statement table.

User response: If you want DB2 Path Checker to provide cost estimates, verify that the DSN_STATEMNT_TABLE exists for the current SQLID or schema. If not, create a DSN_STATEMNT_TABLE. Populate the table with cost data by issuing an EXPLAIN command, and then resubmit the batch job or command.
### CKP050I • CKP201I

**CKP050I**  
DB2 Path Checker did not process the **EXPLAIN** command because the **FROM CATALOG** option requires a plan or collection.  

**Explanation:** The **FROM CATALOG** option was specified for the **EXPLAIN** command, but the required plan or collection was not specified.  

**User response:** Specify a plan name or collection for the **EXPLAIN FROM CATALOG** command to process.

---

**CKP201I**  
The following input parameters were specified. **description**  

**Explanation:** This is a header message that precedes a value and description pair. Each pair describes input parameters that you supply to DB2 Path Checker in `SYSPARM` or the **PARM** field.

**User response:** No action is required.

---

### CKP051E • CKP202I

**CKP051E**  
DB2 Path Checker could not open the data set specified by the `ddname` `dd-name`. The status of the file is `file-status`.  

**Explanation:** DB2 Path Checker could not open the data set that is associated with the `ddname` `dd-name` because its properties do not match the required characteristics for a data control block (DCB).  

**User response:** Change the data set so that it has the required DCB characteristics, and then resubmit the job. The following scenarios describe the required characteristics:

**DBRMOUT points to a PDS**

- The data set must have these characteristics: `DCB=(RECFM=F,LRECL=80,BLKSIZEx=0)`.  
- The data set must be a valid QSAM sequential file such as a `&&temporary`, `VIO`, or permanent file.  
- The data set cannot be a JES spool file.  
- The data set must be allocated with `DISP=OLD`, `SHR`, or `NEW`.  

If the data set is allocated with `DISP=MOD`, DB2 Path Checker does not function correctly. It must be allocated with `DISP=NEW` or `OLD`.

**XMLSQLTX points to a data set that is not valid**

- The data set must have these characteristics: `DCB=(RECFM=E,LRECL=80,BLKSIZEx=x)`, where `x` is 0 to use a block size that is determined by the system or a valid multiple of 80.  
- The data set must be a valid QSAM sequential file such as a `&&temporary`, `VIO`, or permanent file.  
- The data set must be allocated with `DISP=OLD` or `NEW`.  

**CKP202I**  
The **DBRM** does not match standard IBM format.  

**Explanation:** The input DBRM does not match the format of a standard DBRM. DB2 Path Checker does not format this DBRM and continues processing with the next DBRM.  

**User response:** Bind the DBRM into a test package or plan. If the DBRM does not bind correctly, check the precompiler output for error messages. If the DBRM does bind correctly, contact IBM Software Support.

---

**CKP202E**  
The **DBRM** does not match standard IBM format.  

**Explanation:** The format of the input DBRM does not match the mapping of a standard DBRM. DB2 Path Checker does not format this DBRM and continues processing with the next DBRM.  

**User response:** Bind the DBRM into a test package or plan. If the DBRM does not bind correctly, check the precompiler output for error messages. If the DBRM does bind correctly, contact IBM Software Support.

---

**CKP202S**  
DB2 Path Checker encountered a DBRM error.  

**Explanation:** DB2 Path Checker could not process the DBRM.  

**User response:** Verify that the DBRM has not been modified since the precompile step. If the problem persists, contact IBM Software Support.

---

**CKP203I**  
For the program `dbrm-name`, `number-the-same` SQL statements had the same access path and `number-changed` SQL statements had different access paths.  

**Explanation:** This message is an informational message that is displayed at the completion of a `COMPARE` or `TEST` command that is comparing access paths. For each DBRM, `number-the-same` identifies the number of SQL statements that had the same access...
User response: No action is required.

**CKP208I**  DB2 Path Checker identified more than 8000 programs to include in the report, and will process only the first 8000.

**Explanation:** One or more wildcard characters were specified in the program name for this report. The number of programs that met the criteria exceeded 8000. Because this report can process a maximum of 8000 programs, it will include only the first 8000.

User response: Consider using a more restrictive wildcard specification to reduce the number of programs selected.

**CKP209I**  The data set member *member-name* was not found in the PDS specified in the ddname for DBRMIN.

**Explanation:** A program name specified in a TEST or EXPLAIN command did not exist in the partitioned data set (PDS) specified in the ddname for DBRMIN.

User response: Ensure that the ddname points to the PDS that contains the DBRM library (DBRMLIB) and that the specified data set name exists in the DBRMLIB.

**CKP210I**  DB2 Path Checker could not find any packages in SYSPACKAGE that met the criteria specified by the wildcard search for the program *dbrm-name*.

**Explanation:** One or more wildcard characters were specified for the program name *dbrm-name*. No programs in SYSIBM.SYSPACKAGE (or the table for packages specified in the SET CATALOG QUALIFIER field) met the criteria in the wildcard search.

DB2 Path Checker did not process any packages.

User response: Ensure that the specified program name exists in the catalog.

**CKP211I**  DB2 Path Checker could not find any plans in SYSDBRM that met the criteria specified by the wildcard search for the program. DBRM: *plan-or-application-name.dbrm-name*

**Explanation:** One or more wildcard characters were specified for the program name *dbrm-name*. No programs in SYSIBM.SYSDBRM (or the table for DBRMs specified in the SET CATALOG QUALIFIER field) met the criteria in the wildcard search.

DB2 Path Checker did not process any DBRMs.

User response: If you are processing the DBRM as a plan, specify PACKAGE as the object type. If that does not resolve the problem, check the wildcard syntax and ensure that the specified program name exists in the catalog.
CKP212I • CKP217I

CKP212I  DB2 Path Checker issued a COMPARE TO PREVIOUS command. CURRENT TIME STAMP: curr-time-stamp
CURRENT VERSION: curr-version
CURRENT BIND TIME: curr-bind-time
PREVIOUS TIME STAMP: prev-time-stamp
PREVIOUS VERSION: prev-version
PREVIOUS BIND TIME: prev-bind-time

Explanation: This message provides values that are related to a COMPARE TO PREVIOUS command.

curr-time-stamp
Indicates the creation time of the most recent program version. DB2 Path Checker is comparing this version with a previous bind or program version.
curr-version
Indicates the version number of the most recent program version.
curr-bind-time
Indicates the bind time of the most recent program version.
prev-time-stamp
Indicates the creation time of the previous bind or program version. DB2 Path Checker is comparing this previous version with the most recent version of the program as indicated by curr-version.
prev-version
Indicates the version number of the previous bind or program version.
prev-bind-time
Indicates the bind time of the previous bind or program version.

User response: No response is required.

CKP213S  The plan table referenced by the command has an invalid column definition.

Explanation: A column in the specified plan table does not match the required format.

Programmer response: Verify that the plan table has the correct number of columns and the correct name. Then check each column to ensure that the format is correct for the DB2 release. If necessary, reinitialize the plan table in the correct format for the DB2 release, and then rerun the job.

CKP214I  The EXPLAIN command failed with the following error code for the SQL statement that follows: SQLCODE:
sql-statement

Explanation: This message identifies the specific SQL error that caused the EXPLAIN command to fail and includes the SQL text. This message typically indicates that, according to DB2, the specified statement does not qualify for EXPLAIN processing.

This error is displayed only if OPTIONS SQLERROR is specified.

Programmer response: If the SQL statement cannot be processed by EXPLAIN commands, the failure is expected behavior and no action is required. If another program can perform EXPLAIN processing on the statement successfully without using DB2 Path Checker, contact IBM Software Support.

CKP215E  The definition of a column in the plan table is incompatible with any version of DB2 that DB2 Path Checker supports.

Explanation: DB2 Path Checker does not support the format of the plan table. Typically, this error occurs because an earlier version of DB2 Path Checker is running with a later version of DB2.

User response: Verify that you are running the current release of DB2 Path Checker. If not, upgrade to the current release and then rerun the job.

CKP217I  The job completed successfully and reports the following results: TOTAL:
type count

Explanation: This message provides statistics that describe the total number of objects that were processed for each type. DB2 Path Checker reports values for the following types:

DBRMS COMPARED
Indicates the total number of DBRMs that DB2 Path Checker processed in this step.

STATEMENTS COMPARED
Indicates the total number of SQL statements that DB2 Path Checker compared.

ACCESS PATH CHANGES
Indicates the total number of SQL statements for which access path data changed.

ACCESS PATH NO CHANGES
Indicates the total number of SQL statements for which access path data remained the same.

NEW SQL STATEMENTS
Indicates the total number of new SQL statements that DB2 Path Checker processed.
DELETED SQL STATEMENTS
Indicates the total number of deleted SQL statements.

DSN_STMTN COST INCR
Indicates the number of SQL statements for which DB2 estimated an increase in processing cost.
OPTION REPORTCOSTGT was in effect for this job.
User response: No action is required.

CKP218I The job reports a change to the following program: PROGRAM:

User response: No action is required.

CKP219I The number of SQL statements that exceeded the limit of percentage-number percent increase in CPU cost set by OPTIONS CPUPCT is number-of-statements.

Explanation: For the DBRM that DB2 Path Checker is processing, number-of-statements indicates the number of SQL statements that exceeded a user-specified threshold for increased CPU cost. You specify this threshold by setting a percentage value for OPTIONS CPUPCT.

If number-of-statements is greater than 0, DB2 Path Checker writes the REBIND command to PBINDOUT if the data set is available.
User response: No action is required.

CKP220I The original value of the column column-name was value.

Explanation: DB2 Path Checker issues this message for each row for which the value for the column changes as a result of processing a TEST or COMPARE command. The original value in the PLAN table for this column was value.

User response: Compare this original value with the changed value in the CKP221I message.

CKP221I The changed value of the column column-name is value.

Explanation: DB2 Path Checker issues this message for each row for which the value for the column changes as a result of processing a TEST or COMPARE command. The new value in the PLAN table for the column is value.

User response: Compare this changed value with the original value in the CKP220I message.

CKP223I The cost estimate in the DSN_STMTN_TABLE changed, and the new estimated value for service units is new-estimate.

Explanation: The cost estimate in the DSN_STMTN_TABLE changed when DB2 Path Checker compared the EXPLAIN results for the source SQL statement with the EXPLAIN results for the compared SQL statement. The estimate for service units that the SQL query consumes is now new-estimate.

User response: No action is required.

CKP224I The cost estimate in the DSN_STMTN_TABLE changed, and the previous estimated value for service units was prev-estimate.

Explanation: The cost estimate in the DSN_STMTN_TABLE changed when DB2 Path Checker compared the EXPLAIN results for the source SQL statement with the EXPLAIN results for the compared SQL statement. The previous estimate for service units that the SQL query consumes was prev-estimate.

User response: No action is required.

CKP225I The cost estimate in the DSN_STMTN_TABLE changed, and the estimated value for service units is estimate.

Explanation: The cost estimate in the DSN_STMTN_TABLE changed when DB2 Path Checker compared the EXPLAIN results for the source SQL statement with the EXPLAIN results for the compared SQL statement. The estimate for service units that the SQL query consumes is estimate.

This message is issued twice in succession. In the first message, estimate indicates the projected cost of the source query; in the second message, estimate indicates the projected cost of the compared query.

User response: No response is required.

CKP226I The cost estimate in the DSN_STMTN_TABLE changed, and the new estimated value for CPU consumption is new-estimate milliseconds.

Explanation: The cost estimate in the DSN_STMTN_TABLE changed when DB2 Path Checker compared the EXPLAIN results for the source SQL statement with the EXPLAIN results for the compared SQL statement. The estimate for CPU
consumption by this SQL query is now \textit{new-estimate} milliseconds.

This value is displayed in the PROCMS column of the DSN_STATEMNT_TABLE.

\textbf{User response:} No action is required.

\textbf{CKP227I} The cost estimate in the DSN_STATEMNT_TABLE changed, and the previous estimated value for CPU consumption was \textit{prev-estimate} milliseconds.

\textbf{Explanation:} The cost estimate in the DSN_STATEMNT_TABLE changed when DB2 Path Checker compared the EXPLAIN results for the source SQL statement with the EXPLAIN results for the compared SQL statement. The previous estimate for CPU consumption by this SQL query was \textit{prev-estimate} milliseconds.

This value is displayed in the PROCMS column of the DSN_STATEMNT_TABLE.

\textbf{User response:} No action is required.

\textbf{CKP228I} The cost estimate in the DSN_STATEMNT_TABLE changed, and the estimated value for CPU consumption is \textit{estimate} milliseconds.

\textbf{Explanation:} The cost estimate in the DSN_STATEMNT_TABLE changed when DB2 Path Checker compared the EXPLAIN results for the source SQL statement with the EXPLAIN results for the compared SQL statement. The estimate for CPU consumption by this SQL query is \textit{estimate} milliseconds.

This value is displayed in the PROCMS column of the DSN_STATEMNT_TABLE.

This message is issued twice in succession. In the first message, \textit{estimate} indicates the projected cost of the source query; in the second message, \textit{estimate} indicates the projected cost of the compared query.

\textbf{User response:} No action is required.

\textbf{CKP230I} The access path is based on the hint \textit{hint-name}.

\textbf{Explanation:} The message identifies the hint that was used to specify this access path.

\textbf{User response:} No action is required.

\textbf{CKP231I} DB2 Path Checker processed an SQL COMMIT.

\textbf{Explanation:} DB2 Path Checker issued an SQL COMMIT statement, and \texttt{OPTIONS COMMITCOUNT} was specified. The value of \texttt{COMMITCOUNT} determines the number of D8RMs that DB2 Path Checker processes before issuing an SQL COMMIT statement and this message.

\textbf{User response:} No action is required.

\textbf{CKP240I} The BINDIN control card does not specify \texttt{EXPLAIN(YES)}.

\textbf{Explanation:} This SYSPRINT message indicates that a bind control card was read from BINDIN that did not specify \texttt{EXPLAIN(YES)}.

Without \texttt{EXPLAIN(YES)}, the current BIND will create no access paths. Because the bind control cards did not specify \texttt{EXPLAIN(YES)}, the access path analysis that DB2 Path Checker performs will use information from previous binds.

\textbf{User response:} No action is required.

\textbf{CKP242I} No ddname was available for \texttt{PBINDOUT}. DB2 Path Checker cannot write any \texttt{REBIND} commands.

\textbf{Explanation:} The job specified \texttt{OPTIONS} for \texttt{CPUPCT} or \texttt{CPUPCT2}, but the ddname \texttt{PBINDOUT} was not available. DB2 Path Checker cannot write any \texttt{REBIND} commands without a data set to contain them.

\textbf{User response:} Specify a ddname \texttt{PBINDOUT} to point to a valid data set and rerun the job.

\textbf{CKP243I} No ddname was available for \texttt{RBINDOUT}. DB2 Path Checker cannot write any \texttt{REBIND} commands.

\textbf{Explanation:} The job specified \texttt{OPTIONS \texttt{CPUPCT}} or \texttt{OPTIONS \texttt{CPUPCT2}}, but the ddname \texttt{RBINDOUT} was not available. DB2 Path Checker cannot write any \texttt{REBIND} commands without a data set to contain them.

\textbf{User response:} Specify a ddname \texttt{RBINDOUT} to point to a valid data set and rerun the job.

\textbf{CKP244I} DB2 Path Checker cannot write \texttt{REBIND} commands to \texttt{PBINDOUT} because the program has new or changed SQL statements.

\textbf{Explanation:} DB2 Path Checker cannot write \texttt{REBIND} commands to the data set that is specified by the ddname \texttt{PBINDOUT}. \texttt{REBIND} commands cannot refer to programs that contain new or modified SQL statements that have been added since the previous BIND or \texttt{EXPLAIN}.

\textbf{User response:} No action is required.
CKP245I DB2 Path Checker will select program names from BINDIN for processing.

Explanation: The JCL for the job contained a DD statement for BINDIN, which specifies a data set that contains bind control cards. These bind control cards replace the program names in the DB2 Path Checker command with the program names from the BIND commands. DB2 Path Checker uses these program names to determine which plans or packages to process.

User response: No action is required.

CKP246I The bind control commands in the BINDIN data set specified the package to process. COLLECTION: collection-name PROGRAM: package-name.

Explanation: DB2 Path Checker selected the collection or package to process by parsing the bind control commands in the data set that you specified as the ddname for BINDIN.

User response: No action is required.

CKP247I COMPARE processing does not have a second access path to compare with the current access path.

Explanation: A COMPARE command could not find a second access path for the current version of the program to compare with the access path that has the most recent bind time.

This message is displayed only when the SEQBYVERSION option is in effect. This option changes the behavior of a COMPARE command so that it selects access paths by version for comparison. Under this setting, the command compares the access paths that have the most recent bind time to the previous access path for the same version.

User response: No action is required.

CKP301I No ddname was available for RBINDOUT. DB2 Path Checker cannot write any REBIND commands.

Explanation: The job specified OPTIONS for CPUPCT or CPUPCT2, but the ddname RBINDOUT was not available. DB2 Path Checker cannot write any REBIND commands without a data set to contain them.

User response: Specify a ddname RBINDOUT to point to a valid data set and rerun the job.

CKP302I No ddname was available for RBINDOUT. DB2 Path Checker cannot write any REBIND commands.

Explanation: The job specified OPTIONS CPUPCT or OPTIONS CPUPCT2, but the ddname RBINDOUT was not available. DB2 Path Checker cannot write any REBIND commands.

User response: See the error information in DB2 Messages and DB2 SQL Codes to determine the cause of

Chapter 6. Troubleshooting  127
CKP1003S • CKP1021E

CKP1003S The routine ISPTIAR was invoked from trace point trace-point.

Explanation: This message is for debugging. In the message text, trace-point indicates which internal code path was active when a DB2 error occurred.

User response: Contact IBM Software Support and provide the contents of this message.

CKP1004E A menu entry was not a valid option.

Explanation: A specified menu option is not an available option.

User response: Specify a valid selection from the menu.

CKP1010E No subsystem name was specified.

Explanation: A DB2 Path Checker function was requested, but a valid subsystem name was not specified.

User response: Specify a valid subsystem on the SETTINGS panel (option 0 on the main menu).

CKP1011E The connection with DB2 failed.

Explanation: DB2 Path Checker was unable to connect to the specified DB2 subsystem. In the message text, RCn represents an internal diagnostic code.

User response: Verify that the specified subsystem is running and is accessible from the system where you are running DB2 Path Checker, and then try again. If the problem persists, contact IBM Software Support and provide the information in the message.

CKP1013S The ISPF service service-name failed and returned the diagnostic code diagnostic-code. RC: return-code

Explanation: An ISPF service failed with a nonzero condition code. In the message text, service-name is the failing service, diagnostic-code is the internal return code that DB2 Path Checker issues, and return-code is the condition code from the ISPF service.

User response: See Interactive System Productivity Facility (ISPF) Messages and Codes for information about the ISPF return code that was included in this message and perform the recommended action. If the problem persists, contact IBM Software Support.

CKP1016I COMPARE processing detected no changed access paths.

Explanation: No changed access paths were discovered when DB2 Path Checker compared the previous EXPLAIN results with the current ones.

User response: No action is required.

CKP1017E The plan table that was specified on the panel as the previous version does not exist in this DB2 subsystem.

Explanation: The previous version of the plan table that was specified for comparison does not exist.

User response: Specify a valid plan table name on the panel.

CKP1018E The plan table that was specified on the panel as the current version does not exist in this DB2 subsystem.

Explanation: The current version of the plan table that was specified for comparison does not exist.

Programmer response: Specify a valid plan table name on the panel.

CKP1019E DB2 Path Checker cannot process the plan table because the name is not fully qualified and no default authorization ID is available.

Explanation: Plan table names must be fully qualified or you must provide a default authorization ID.

User response: Either specify a fully qualified plan table name or go to the SETTINGS panel (main menu option 0) and specify a default authorization ID.

CKP1020E The plan table name is too long to fit on the command line.

Explanation: The fully qualified plan table name must be short enough to fit on a single 80 character line.

User response: Specify a fully qualified plan table name that is short enough to fit on a single 80 character line.

CKP1021E DB2 Path Checker cannot process the plan table because the name is not fully qualified and no default authorization ID is available.

Explanation: Plan table names must be fully qualified or you must provide a default authorization ID.

User response: Either specify a fully qualified plan table name or go to the SETTINGS panel (main menu option 0) and specify a default authorization ID.
CKP1022E  An OPEN CURSOR operation failed for the cursor cursor-id. SQLCODE: sql-code

Explanation: A DB2 interface error occurred. In the message text, cursor-id is an identifier for the cursor and sql-code is the SQL error code that was returned by the OPEN CURSOR request.

User response: See DB2 Messages and DB2 SQL Codes for error information about the SQL code that was included in this message. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the SQL code.

CKP1023E  A FETCH request failed for the cursor cursor-id. SQLCODE=sql-code

Explanation: A DB2 interface error occurred. In the message text, cursor-id is the identifier for the cursor and sql-code is the SQL error code that was returned by the FETCH request.

User response: See DB2 Messages and DB2 SQL Codes for error information about the SQL code that was included in the message. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the SQL code.

CKP1024E  DB2 Path Checker cannot process the plan table because the name is not fully qualified and no default authorization ID is available.

Explanation: Plan table names must be fully qualified or you must provide a default authorization ID.

User response: Either specify a fully qualified plan table name or go to the SETTINGS panel (main menu option 0) and specify a default authorization ID.

CKP1025S  Invalid data was encountered in SYSIBM.SYSCOLUMNS. REASON: reason-code

Explanation: The DB2 catalog contains invalid data. In the message text, reason-code is an SQL error code from DB2.

User response: See DB2 Messages and DB2 SQL Codes for error information about the reason code that was included in the message. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the reason code.

CKP1026S  The VGET service failed. TYPE: type-code, RC: return-code

Explanation: An ISPF interface error occurred in the VGET service. In the message text, type-code is an internal diagnostic code and return-code is the return code from the VGET service.

User response: Verify that the TSO user has a valid ISPPROF data set allocated and has read access to the data set. If the problem persists, contact IBM Software Support and provide the information in the message.

CKP1027S  The VPUT service failed. TYPE: type-code, RC: return-code

Explanation: An ISPF interface error occurred in the VPUT service. In the message text, type-code is an internal diagnostic code and return-code is the return code from the VPUT service.

User response: Verify that the TSO user has a valid ISPPROF data set allocated and has update access to the data set. If the problem persists, contact IBM Software Support and provide the information in the message.

CKP1028E  No database name was specified in the tablespace name.

Explanation: The specified tablespace name did not contain a database name. DB2 requires a fully qualified tablespace name that includes the database name.

User response: Either provide a database name or omit the tablespace name.

CKP1029E  A CREATE TABLE request failed. SQLCODE: sql-code

Explanation: A DB2 interface error occurred. In the message text, sql-code is the SQL error code that was returned by the CREATE TABLE request.

User response: See DB2 Messages and DB2 SQL Codes for error information about the SQL code that was included in the message. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the SQL code.

CKP1030I  A plan table was created in DB2 Version 7 format, and consists of 51 columns.

Explanation: This message indicates that DB2 Path Checker created a plan table in Version 7 format. The plan table has 51 columns.

User response: No action is required.
<table>
<thead>
<tr>
<th><strong>CKP1031I</strong></th>
<th><strong>CKP1042E</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CKP1031I A plan table was created in DB2 Version 8 format, and consists of 58 columns.</td>
<td><strong>Explanation</strong>: This message indicates that DB2 Path Checker created a plan table in Version 8 format. The plan table has 58 columns. <strong>User response</strong>: No action is required.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1032E</strong></th>
<th><strong>Symptoms</strong>: A CREATE TABLE request failed. SQLCODE: sql-code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A DB2 interface error occurred. In the message text, sql-code is the SQL error code that was returned by the CREATE TABLE request. <strong>User response</strong>: See DB2 Messages and DB2 SQL Codes for error information about the SQL code that was included in the message. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the SQL code.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1033E</strong></th>
<th><strong>Symptoms</strong>: A DELETE request failed. SQLCODE: sql-code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A DB2 interface error occurred. In the message text, sql-code is the SQL error code that was returned by the DELETE request. <strong>User response</strong>: See DB2 Messages and DB2 SQL Codes for error information about the SQL code that was included in the message. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the SQL code.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1034I</strong></th>
<th><strong>Symptoms</strong>: The RESET function successfully deleted all rows in the plan table.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: The RESET function successfully deleted all rows from the specified plan table. <strong>User response</strong>: No action is required.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1035I</strong></th>
<th><strong>Symptoms</strong>: The CKPPTHCK program requires a BIND.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: The CKPPTHCK program that is running does not match the last bind of the program in the DB2 subsystem. You must bind CKPPTHCK. <strong>User response</strong>: Bind the program CKPPTHCK and run the job again.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1037I</strong></th>
<th><strong>Symptoms</strong>: A RESET operation was requested for a plan table that does not exist.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A RESET operation was requested for a nonexistent plan table. <strong>User response</strong>: Specify a valid plan table name.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1038I</strong></th>
<th><strong>Symptoms</strong>: The CREATE TABLE command specified a plan table that already exists.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A CREATE TABLE command specified a plan table that already exists. <strong>Programmer response</strong>: If the plan table name is correct, no action is required.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1039E</strong></th>
<th><strong>Symptoms</strong>: A DB2 COMMIT request failed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A DB2 interface error has occurred. In the message text, sql-code is the SQL error code that was returned by the COMMIT request. <strong>User response</strong>: Verify that the DB2 subsystem is active and check the z/OS system log for errors. See the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the information that is in the message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1040E</strong></th>
<th><strong>Symptoms</strong>: A DB2 COMMIT request failed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A DB2 interface error has occurred. In the message text, sql-code is the SQL error code that was returned by the COMMIT request. <strong>User response</strong>: See the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If the solutions in DB2 Messages and DB2 SQL Codes do not solve the problem, contact IBM Software Support and provide the information that is in the message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1041E</strong></th>
<th><strong>Symptoms</strong>: The plan name is too long. Plan names must be eight characters or less.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: The plan name contains too many characters. <strong>User response</strong>: Specify a valid plan name. Plan names must have 1 - 8 characters. <strong>Module</strong>: CKPA100</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CKP1042E</strong></th>
<th><strong>Symptoms</strong>: The TEST command requires a valid SQLID.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation</strong>: A valid SQLID must be specified on the SETTINGS panel before issuing a TEST command. <strong>Programmer response</strong>: Specify a valid SQLID on the SETTINGS panel. <strong>Module</strong>: CKPA100</td>
<td></td>
</tr>
</tbody>
</table>
The CUNLCNV routine could not be loaded.

Explanation: DB2 Path Checker was unable to retrieve the z/OS Unicode Services routine CUNLCNV.

Programmer response: Ensure that the load library that contains the CUNLCNV routine is available.

Module: CKPA100

The CUNLCNV routine did not initialize. RETURN CODE: return-code REASON CODE: reason-code

Explanation: The z/OS Unicode Services routine CUNLCNV did not completely initialize.

User response: See z/OS Messages and Codes to determine why CUNLCNV did not initialize, and correct the problem. If the problem persists, contact IBM Software Support.

Module: CKPA100

The CUNLCNV routine could not convert a character string. RETURN CODE: return-code REASON CODE: reason-code

Explanation: The z/OS Unicode Services routine CUNLCNV could not convert a string of characters.

Programmer response: See z/OS Messages and Codes to determine why CUNLCNV could not convert the data and correct the problem. If the problem persists, contact IBM Software Support.

Module: CKPA100

In the TO clause, you cannot specify the PREVIOUS or PREVIOUS VERSION option and at the same time specify names for the plan or collection and the plan table.

Explanation: In the TO clause, you must specify parameter values for only one of the following choices:

- The PREVIOUS or PREVIOUS VERSION option
- The names of the plan or collection and the plan table

The two choices are mutually exclusive.

User response: Specify parameter values for only one of the following choices:

- The PREVIOUS or PREVIOUS VERSION option
- The names of the plan or collection and the plan table

The job was created successfully.

Explanation: DB2 Path Checker successfully created the job.

User response: No action is required.

The ddname specified as an input dataset does not exist.

Explanation: A data set name that does not exist was specified in one of the DD statements on the JCL Options panel.

User response: Specify the name of an existing data set for the DD statement.

The command-name command was built successfully. Press END to save and continue or CANCEL to back out.

Explanation: The input for the command was valid and accepted, and the command was built.

User response: No action is required.

An attempt to display the panel-name panel failed. RC: return-code

Explanation: The ISPF display service failed while the panel panel-name was being displayed.

User response: See the ISPF Services Guide for error information about the return code that was included with the message and perform the recommended action. If the problem persists, contact IBM Software Support.

The job could not be generated because the data set name for the job must be an existing PDS with a specified member name.

Explanation: To generate a job, the partitioned data set (PDS) specified for the JOB DSN must exist and you must specify the member name.
User response: Specify an existing PDS and a member name for JOB DSN.

CKP1107E The EXPLAIN or TEST command could not be generated because a SET CURRENT SQLID command must be generated first.

Explanation: DB2 Path Checker requires a valid SQLID to issue an EXPLAIN or TEST command.

User response: Generate a SET CURRENT SQLID command that specifies a valid SQLID before you select the EXPLAIN or TEST command.

CKP1108E The FTOPEN service in ISPF failed. RC: return-code

Explanation: An error occurred while attempting to generate a job.

User response: See the ISPF Services Guide for error information about the return code that was included in the message and perform the recommended action. If the problem persists, contact IBM Software Support.

CKP1109E The ddname DBRMIN is missing. Specify DBRMIN on the JCL Options panel.

Explanation: An EXPLAIN command was issued with the FROM CATALOG parameter. When the program attempted to read the DBRM, the ddname DBRMIN was missing.

User response: Add a ddname for DBRMIN on the JCL Options panel that points to the appropriate DBRM library (DBRMLIB).

CKP1110E The data set specified on the Batch panel for JOB DSN could not be allocated to the ISPFILE DD. The allocation failed with return code return-code.

Explanation: The data set that was specified to hold the generated JCL could not be allocated because the data set was not valid. In the message text, return-code is an internal diagnostic code from ISPF.

User response: See the ISPF Services Guide for error information about the return code that was included in the message and perform the recommended action. If the problem persists, contact IBM Software Support.

CKP1111E The COMPARE command requires one or more values in the TO clause.

Explanation: In the TO clause, specify which access paths to compare with the access paths of the current plan or package. You can compare the current plan or package with any of the following access paths:

- Access paths for a plan or collection with a different name
  - Specify the name of the plan or package and the name of the plan table.
- Access paths for the next oldest (-1) edition of the current plan or package
  - Specify the PREVIOUS parameter.
- Access paths for the previous version of the current plan or package
  - Specify the PREVIOUS VERSION parameter.

User response: Either specify the names of a plan or collection and a plan table, or specify one of the PREVIOUS parameters.

CKP1112E The TO clause requires both the name of a plan or collection and a name for a plan table.

Explanation: In the TO clause, if you specify the name of a plan or collection then you must also specify the name of a plan table. If you specify the name of a plan table then you must also specify the name of a plan or collection.

User response: Specify values for both the name of a plan or collection and the name of a plan table.

CKP1113E You cannot specify PREVIOUS or PREVIOUS VERSION with BEFORE.

Explanation: The PREVIOUS or PREVIOUS VERSION options are mutually exclusive with the BEFORE option. You must specify one or the other.

User response: Specify one of the three values. If you specify PREVIOUS or PREVIOUS VERSION, you do not have to specify a plan or collection name or a plan table name. If you specify the BEFORE option, include a plan or collection name and a plan table name.

CKP1114E In the TO clause, specify both a plan or collection name and a plan table name.

Explanation: In the TO clause, both of the following values are required:

- A plan or collection name
- A plan table name

User response: Specify both a plan or collection name and a plan table name.

CKP1115E The FROM CATALOG parameter requires specifications for both object type and qualifier.

Explanation: The syntax of the EXPLAIN command requires that you specify the object type (PLAN or PACKAGE) and the name of the plan or collection.
CKP1116I • CKP1126W

CKP1116I The CONNECT command was built successfully. Press ENTER to continue.
Explanation: The CONNECT command was built and added to the syntax for the job.
User response: No action is required.

CKP1117I The validation of the specified data sets was successful. Press END to continue.
Explanation: The entries on the JCL Options panel are valid and meet all requirements. When you specify Generate Job, Edit Job, or Submit Job, DB2 Path Checker will process the job.
User response: No action is required.

CKP1118E The data set could not be validated because the attributes were not valid.
Explanation: A data set was not validated because its attributes did not match the requirements.
User response: Ensure that the specified data set was allocated with the appropriate data set attributes. For more information, see "Job control statements" on page 174.

CKP1119E The ISPF service DSINFO failed with return code return-code.
Explanation: DSINFO failed while the attributes of a data set were being validated during allocation.
User response: See the ISPF Services Guide for error information about the return code that was included in the message and perform the recommended action. If the problem persists, contact IBM Software Support.

CKP1120E The job could not be generated because of an error in the ISPF service FTINCL. RC: return-code
Explanation: The attempt to generate the job was unsuccessful. This error typically occurs because DB2 Path Checker could not access the skeleton library.
User response: Verify that the name of the skeleton library that is specified in the invocation CLIST refers to a valid data set.
See the ISPF Services Guide for error information about the return code that was included in the message and perform the recommended action. If the problem persists, contact IBM Software Support and supply the return code.

CKP1121S The ISPF service DSINFO failed with return code return-code.
Explanation: DSINFO failed while the attributes of a data set were being validated during allocation.
DSINFO ended with return code return-code.
User response: See the ISPF Services Guide for error information about the return code that was included in the message and perform the recommended action. If the problem persists, contact IBM Software Support.

CKP1122E The specified data set must be in PDS(MBR) format.
Explanation: The specified data set must be a partitioned data set (PDS) with a valid member name.
User response: Specify an existing PDS that exists and include the member name.

CKP1123E The job failed. RC: return-code
Explanation: The submitted job failed. In the message text, return-code is an internal diagnostic code.
User response: Contact IBM Software Support and provide the information in the message.

CKP1124E The CPUPCT and CPUPCT2 options cannot be used together in the same command. Use one option or the other.
Explanation: You cannot specify CPUPCT and CPUPCT2 together in a single command.
Programmer response: Specify either the CPUPCT or the CPUPCT2 option.

CKP1125E The command could not process any data. When the CPUPCT or CPUPCT2 options are in effect the command requires specifications for RBINDOUT and PBINDOUT.
Explanation: The CPUCPT or CPUPCT2 option was specified, and both options require that you specify RBINDOUT and PBINDOUT in the JCL. To write and store REBIND commands, DB2 Path Checker requires that these ddnames specify valid data sets.
User response: Specify valid data sets for RBINDOUT and PBINDOUT in the JCL.

CKP1126W The APCOMPARE and APREUSE options cannot be specified in the same OPTIONS command. The last option specified will be used.
Explanation: You cannot use the APCOMPARE option and the APREUSE option together in the same OPTIONS command.
**User response:** Specify a value of NO or NONE for either the APCOMPARE option or the APREUSE option.

**Explanation:** The value of the APREUSE option is not valid.

**User response:** Specify one of the following values for APREUSE: N (NONE), W (WARN), or E (ERROR).

**Explanation:** The value of the APREUSE option is not valid.

**User response:** Specify one of the following values for APREUSE or APCOMPARE option:

- CURRENT
- PREVIOUS
- ORIGINAL

**Explanation:** An invalid option was specified for the APREUSE or APCOMPARE option. Valid values are CURRENT, PREVIOUS, or ORIGINAL.

**User response:** Specify a valid value for the VERSION parameter.

**Explanation:** The value of the COPYID parameter was not valid. Valid values for COPYID are CURRENT, PREVIOUS, or ORIGINAL.

**User response:** Specify one of the following values for COPYID:

- CURRENT
- PREVIOUS
- ORIGINAL

**Explanation:** The EXPLAIN PACKAGE command is only supported in Version 10 or later of DB2.

**User response:** Use the EXPLAIN PACKAGE command only with a Version 10 or later of DB2.

**Explanation:** An EXPLAIN PACKAGE command was run against a version of DB2 that does not support this command.

**User response:** If you want to use the option or value, you must connect DB2 Path Checker to a version of DB2 that supports it.

**Explanation:** The version of the DB2 subsystem that is running in the subsystem ssid does not support the specified BIND option bind-option.

**User response:** Specify a valid value for the VERSION parameter.

**Explanation:** If you did not specify VERSION(AUTO) as an input option to the DB2 pre-processor or co-processor, this value is a user-specified character sequence. If you used VERSION(AUTO), then the value is an automatically-generated consistency token in character format (usually formatted as a time stamp, such as '20121222211216').

**User response:** Specify a valid value for the VERSION parameter.
| Explanation:  If AUTOBIND is enabled, DB2 will bind the package or plan the first time it is used. For that run, the loading process for the bind might affect the performance of the program.  
| User response:  Ensure that the AUTOBIND setting reflects your site's standards for DB2 behavior when it uses a package or plan that was created in a DB2 version earlier than V9.  

| CKP1143W  | The package or plan program-name was rebound with EXPLAIN(ONLY). This might cause a comparison to report that access paths did not change when differences do exist.  
| Explanation:  The EXPLAIN(ONLY) BIND or EXPLAIN(ONLY) REBIND option created access path information that was used in the comparison. The limitations of the EXPLAIN(ONLY) option might cause DB2 Path Checker to report that no access paths changed when one or more of them did change.  
| User response:  Consider running the TEST command again, or run the EXPLAIN PACKAGE command followed by the COMPARE command.  

| CKP1145E  | An invalid option was specified for DESCSTAT. Valid values are YES or NO.  
| Explanation:  The value of the DESCSTAT option is not valid.  
| User response:  Specify one of the following values for DESCSTAT: YES or NO.  

| CKP1146E  | An invalid option was specified for APPLCOMPAT. Valid values are V10R1 or V11R1.  
| Explanation:  The value of the APPLCOMPAT option is not valid.  
| User response:  Specify one of the following values for APPLCOMPAT: V10R1 or V11R1.  

---

### How to look up message explanations

You can use several methods to search for messages and codes.

#### Searching an information center

In the search box that is located in the top left toolbar of any Eclipse help system, such as the IBM Information Management Software for z/OS Solutions Information Center, enter the number of the message that you want to locate. For example, you can enter DFS1065A in the search field.

Use the following tips to improve your message searches:
- You can search for information on codes by entering the code; for example, enter -327.
- Enter the complete or partial message number. You can use the asterisk wildcard character (*) to represent multiple characters, and you can use the question mark wildcard character (?) to represent a single character.

The information center contains the latest message information for all of the Information Management products that are included in the information center.

#### Searching for messages on the web

You can use any of the popular search engines that are available on the web to search for message explanations. When you type the specific message number or code into the search engine, you are presented with links to the message information in IBM information centers.

#### Gathering diagnostic information

Before you report a problem with DB2 Path Checker to IBM Software Support, you need to gather the appropriate diagnostic information.
Procedure

Provide the following information for all DB2 Path Checker problems:

- A clear description of the problem and the steps that are required to re-create the problem
- All messages that were issued as a result of the problem
- Product release number and the number of the last program temporary fix (PTF) that was installed
- The version and mode of DB2 and z/OS that you are using
- The type and version of the operating system that you are using
- Diagnostic DD Card

//SYSDIAGN DD SYSOUT=*

This is a default setting that returns diagnostic information including the compile date of each module, which allows IBM Software Support to determine the maintenance level of the code and recreate the error if necessary. SYSDIAGN is automatically allocated to support First Failure Data Collection, or FFDC. The FFDC report is used to begin a diagnostic investigation the first time an error occurs. You can preallocate SYSDIAGN as shown below to disable FFDC, however, doing so might require additional work on your part to recreate the error if an error occurs.

//SYSDIAGN DD DUMMY

- Plan Table, a DB2 Path Checker input
- DBRMLIB, a DB2 Path Checker input
- Backup Plan Table, a DB2 Path Checker input
- DBRM

The DBRM is a unique file. You must use the TSO XMIT command to transform the DBRM into a .bin file so that it can be received at IBM in a readable format. To format the file correctly and transmit it:

1. Issue the TSO XMIT command: XMIT NODE.user
   DSN ("DBRM NAME")
   MEMBER (NAME)
   OUTDSN (OUT.DS.NAME.BIN)
   SEQ

2. Save the file on the hard disk drive as a .bin file. Ensure that the file always remains a .bin file; otherwise, the file will be unreadable when it arrives at IBM.
3. FTP the .bin file to IBM.
   IBM will perform the RECEIVE to complete the process.

Provide additional information based on the type of problem that you experienced:

For all DB2 Path Checker batch problems, provide the following information:

- A complete job log with MSGLEVEL(1,1) showing the error
- A complete job log with the dump of the abend
- A complete job log showing the loop, such as a message that is repeated multiple times
- A complete job log and a dump while in the hung condition
- A complete job log and a sample of the bad output

For all DB2 Path Checker Interactive problems, provide the following information:
• Screen prints showing the steps that occurred before the failure, as possible
Chapter 7. Reference

The reference information provides easy access to command syntax and other的技术 about DB2 Path Checker.

Details include customization information, samples, job control statements, and parameters.

Tools Customizer terminology and data sets

Before you use Tools Customizer, you should understand the Tools Customizer terminology and the data sets that Tools Customizer uses during customization.

Tools Customizer terminology

Tools Customizer uses several unique terms that you should be familiar with before you begin to use Tools Customizer.

Products and components

How an IBM Tool is packaged determines whether it is referred to as a product or as a component in the Tools Customizer documentation and interface. An IBM Tool that is ordered as a stand-alone entity (that is, not as part of a solution pack) is referred to as a product. An IBM Tool that is part of a solution pack is referred to as a component. Some IBM Tools are available in both formats; therefore, the same IBM Tool can be referred to as a product or as a component depending on how it is packaged.

DB2 entry

You can customize DB2 Path Checker on one or more DB2 entries. A DB2 entry can be any of the following items:

DB2 subsystem

A distinct instance of a relational database management system (RDBMS) that is not part of a data sharing group. An example of a DB2 subsystem name is DB01.

DB2 group attach name

DB2 Path Checker does not support DB2 group attach names.

DB2 data sharing member

A DB2 subsystem that is assigned by the cross-system coupling facility (XCF) to a data sharing group. An example of a DB2 data sharing member name is DB02.

Tools Customizer maintains the following lists of DB2 entries:

Associated list

The list of DB2 entries that are associated with DB2 Path Checker. If the product to be customized requires DB2 entries, you can customize DB2 Path Checker only on DB2 entries that are in the associated list. When you customize DB2 Path Checker, this list is displayed in the DB2 Entries, Associations, and Parameter Status section of the Customizer Workplace panel.
You can add and copy DB2 entries to the associated list. When you add or copy DB2 entries to the associated list, the entries are associated with DB2 Path Checker.

**Master list**

The list of all DB2 entries that are defined but are not associated with DB2 Path Checker. Tools Customizer obtains information about these DB2 entries either from entries that were created manually or from the customizations of other products that were discovered. If you remove a DB2 entry from the associated list, the DB2 entry is added to the master list. When you create a new DB2 entry, it is added to the master list, and when you associate the new entry with DB2 Path Checker, it is removed from the master list and added to the associated list. The master list is displayed on the Associate a DB2 Entry for Product panel.

If the associated list does not have the DB2 entries on which you want to customize DB2 Path Checker, you can associate existing entries from the master list to the associated list.

You can create new DB2 entries and copy existing entries to the master list.

**High-level qualifier**

The high-level qualifier is considered to be all of the qualifiers except the lowest level qualifier. A high-level qualifier includes a mid-level qualifier.

**Product parameters**

Parameters that are specific to DB2 Path Checker. These parameters are defined by DB2 Path Checker and are stored in a data member that is defined by DB2 Path Checker.

**DB2 parameters**

Parameters for a DB2 entry. These parameters are defined by Tools Customizer and are stored in a DB2 parameter data member.

**Status type**

**Product, LPAR, and DB2 entry status type**

After you specify the product that you want to customize, the product, the LPAR, and the DB2 entries have a status. The status is partly based on whether required parameters are defined. For some products, LPAR parameters or DB2 parameters might not be required. In these cases, the status is Not Required.

To customize DB2 Path Checker, all of the required parameters must be defined.

If required parameters for the the product parameters or DB2 parameters are not defined, the status of the parameters is Incomplete. Define values for parameters by manually editing them or by generating the customization jobs and specifying values for all of the required parameters that are displayed on the panels.

When values for all of the required parameters are defined, the status is Ready to Customize. Customization jobs can be generated only when all of the required parameters are defined and the status is Ready to Customize or Customized for the product parameters and DB2 parameters for the DB2 entries on which DB2 Path Checker will be customized.
The following table shows the meaning of the status types. Each status is defined differently for each type of parameter.

<table>
<thead>
<tr>
<th>Status</th>
<th>Product</th>
<th>LPAR</th>
<th>DB2 entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete</td>
<td>The required product parameters are not defined, or the required product parameters are defined but LPAR parameters, DB2 parameters, or both are not defined.</td>
<td>The required parameters are not defined.</td>
<td>The required parameters are not defined.</td>
</tr>
<tr>
<td>Discovered</td>
<td>The product parameter definitions were discovered by using the product Discover EXEC.</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Ready to Customize</td>
<td>The required product, LPAR, and DB2 parameters are defined, the status is Ready to Customize or Customized for the LPAR and at least one associated DB2 entry. You can generate the customization jobs.</td>
<td>The required LPAR parameters are defined or LPAR parameters are not required.</td>
<td>The required DB2 parameters are defined or DB2 parameters are not required.</td>
</tr>
<tr>
<td>Customized</td>
<td>The jobs are customized on the local LPAR.</td>
<td>The jobs are customized for the product or for all of the associated DB2 entries on the local LPAR.</td>
<td>The jobs are customized for the DB2 entry.</td>
</tr>
<tr>
<td>Errors in Customization</td>
<td>N/A</td>
<td>N/A</td>
<td>Errors occurred while the customization jobs were being generated.</td>
</tr>
<tr>
<td>Not Required</td>
<td>N/A</td>
<td>LPAR parameters are not required.</td>
<td>DB2 parameters are not required.</td>
</tr>
</tbody>
</table>

Related tasks:

“Creating and associating DB2 entries” on page 47
You can create new DB2 entries and associate them with DB2 Path Checker.
“Copying DB2 entries” on page 58
You can copy associated and not associated DB2 entries to other DB2 entries or to new DB2 entries.
“Removing DB2 entries” on page 59
You can remove DB2 entries from the associated list.

Data sets that Tools Customizer uses during customization

Tools Customizer uses several unique data sets during the customization process. Familiarize yourself with these data sets before you begin to use Tools Customizer.
Several different data sets are required to customize DB2 Path Checker with Tools Customizer. These data sets are supplied by DB2 Path Checker, supplied by Tools Customizer, or allocated by Tools Customizer.

DB2 Path Checker provides the following data sets:

**Metadata library**
Contains the metadata for the product to be customized. Tools Customizer uses the metadata to determine which tasks, steps, and parameters to display on the Product Parameters panel, the LPAR Parameters panel, and the DB2 Parameters panel. This data set also contains the templates that Tools Customizer uses to generate the customization jobs.

The metadata library naming convention is `high_level_qualifier.SCKPDYXIMZDENU`, where `high_level_qualifier` is all of the segments of the data set name except the lowest-level qualifier.

You specify the metadata library on the Specify the Metadata Library panel. READ access to this data set is required.

**Discover EXEC library**
Contains the DB2 Path Checker Discover EXEC. When you customize DB2 Path Checker, you can use the Discover EXEC to automatically retrieve and store product information, such as parameter values from an already customized product. Tools Customizer saves the discovered information in the data store.

The default name of the data set is the high-level qualifier for the metadata library plus a lowest-level qualifier. For DB2 Path Checker, the lowest-level qualifier is SCKPDYXIMZDENU. You can change the default value on the Discover Customized Product Information panel. EXECUTE access to this data set is required.

Tools Customizer provides the following data sets:

**Tools Customizer metadata library**
Contains the metadata for the DB2 and LPAR parameters that are required to customize DB2 Path Checker. Tools Customizer uses the metadata to determine which parameters to display on the DB2 Parameters panel and the LPAR Parameters panel. In addition, Tools Customizer uses information in the metadata library to determine whether additional DB2 and LPAR parameters need to be displayed on these panels. As you customize different products, different DB2 and LPAR parameters might need to be defined.

The default name of the data set is `DB2TOOL.CCQ110.SCCQDENU`. You can change the default value on the Tools Customizer Settings panel. READ access to this data set is required.

**Tools Customizer table library**
Stores information about jobs that are customized. Job information that is stored includes a description of the job, its member name and template name, the SSID, and when the job was generated.

The default name of the data set is `DB2TOOL.CCQ110.SCCQTENU`. WRITE access to this data set is required.

Tools Customizer requires that the following data sets exist during the customization process. If the data sets do not exist, Tools Customizer automatically allocates them.
Discover output data set
Contains the output that is generated when you run the DB2 Path Checker Discover EXEC. The DB2 Path Checker Discover EXEC retrieves the metadata and values for the parameters from a previous customization of DB2 Path Checker.

The default name of the data set is DB2TOOL.CCQ110.DISCOVER. You can change the default value on the Tools Customizer Settings panel or the Discover Customized Product Information panel. WRITE access to this data set is required.

Data store data set
Contains product, LPAR, and DB2 parameter values, and DB2 entry associations. Tools Customizer uses this data set to permanently store all information that is acquired about the product, DB2 subsystems, and LPAR when you customize products on the local LPAR.

The default name of the data set is DB2TOOL.CCQ110.DATASTOR. You can change the default value on the Tools Customizer Settings panel. WRITE access to this data set is required.

Customization library
Contains the customization jobs that Tools Customizer generates for DB2 Path Checker.

Tools Customizer checks whether a customization library name was specified for more than one instance of the same version of the same product. If the same customization library name is specified for more than one product of the same version, the CCQD123E message is issued to prevent you from overwriting previously generated customization jobs. Ensure that you specify unique qualifier for the customization library for each instance of the product.

To customize DB2 Path Checker, submit the members of the data set in the order in which they are displayed on the Finish Product Customization panel.

The data set naming convention is hlq.$LPAR_name$.xyzvrm, where:
- hlq is the value of the Customization library qualifier field on the Tools Customizer Settings panel (CCQPSET)
- LPAR_name is the four-character LPAR name
- xyzvrm is the three-letter product identifier with the version, release, and modification level

For example, the data set name might be DB2TOOL.PRODUCT.CUST.$MVS1$.XYZ410.

WRITE access to this data set is required.

Tools Customizer allocates the data sets for the discover output, the data store, and the customization library with the attributes that are shown in the following table:

<table>
<thead>
<tr>
<th>Data set</th>
<th>Organization</th>
<th>Record format</th>
<th>Record length</th>
<th>Block size</th>
<th>Data set name type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover output data set</td>
<td>PO</td>
<td>Variable block</td>
<td>16383</td>
<td>32760</td>
<td>LIBRARY</td>
</tr>
</tbody>
</table>

Table 11. Data set attributes for allocating the Discover output, data store, and customization library data sets
### Table 11. Data set attributes for allocating the Discover output, data store, and customization library data sets (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Organization</th>
<th>Record format</th>
<th>Record length</th>
<th>Block size</th>
<th>Data set name type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data store data set</td>
<td>PO</td>
<td>Variable block</td>
<td>16383</td>
<td>32760</td>
<td>LIBRARY</td>
</tr>
<tr>
<td>Product customization library</td>
<td>PO</td>
<td>Fixed block</td>
<td>80</td>
<td>32720</td>
<td>LIBRARY</td>
</tr>
</tbody>
</table>

**Restrictions:**

- Multiple users cannot simultaneously share the discover output data set, data store data set, Tools Customizer metadata library, and metadata library.
- You cannot share the data store data set across multiple LPARs with shared DASD or copy the data store data set to another LPAR. Tools Customizer creates many cross-references between product and DB2 associations. Therefore, if you share or copy the data store data set, member names that are empty or that do not exist might be generated.

### DB2 Path Checker command reference

Use this information to learn how to use the commands that are available in DB2 Path Checker. Refer to the topics about individual commands to find the correct command syntax.

You can pass commands to DB2 Path Checker from the SYSIN data set or include them in a batch job.

You can specify parameters that are equivalent to these commands by using the PARM field on the EXEC statement. If you specify both parameters and commands, all PARM field parameters are ignored.

**Related tasks:**

- "Running DB2 Path Checker by submitting batch jobs" on page 67
- Run DB2 Path Checker by submitting a batch job that contains the required load libraries, DD statements, options, and commands.

**Related reference:**
- "Job control statements" on page 174
- Job control statements specify DB2 Path Checker functions and what options to use.

### Syntax rules for specifying commands

DB2 Path Checker commands must adhere to the following specific syntax rules.

- Any line that uses dashes as its first two non-blank characters is considered to be a comment and is ignored.
- If a line begins with the comment characters (- -), the entire line is treated as a comment. A comment cannot appear on the same line as a command or a portion of a command.
- A single command can span two or more lines. However, each command must begin on a new line (more than one command on a line is not allowed).
* The keywords and values that make up a command must be separated by one or more blanks.

You can specify each command one or more times. Commands are processed using the values that are specified on the most recent OPTIONS command.

**Wildcard characters in command syntax**

The REPORT, COMPARE, EXPLAIN, and TEST commands can use wildcard characters to specify multiple objects for processing.

DB2 Path Checker supports the following wildcard characters:

- %  Any single character
- *  Any sequence of zero or more characters

Because DB2 Path Checker supports wildcard characters in its command syntax, you can organize your naming scheme to process multiple DBRMs that are related to the same application at the same time. For example, you can provide the wildcard specification `cmg*` for commercial mortgages to process DBRMs with the names `cmgpmnts`, `cmgnotic`, and `cmgforcl`:

If a command or parameter uses wildcard characters to specify the program name, the member list is retrieved from the DB2 catalog. Plan requests use SYSDBRM, and package requests use SYSPACKAGE. The normal qualifier for the catalog tables is SYSIBM unless you use the SET CATALOG QUALIFIER command to specify a different value.

The TEST and EXPLAIN commands select program names from the MEMBER list for the data set that you specify as the ddname for DBRMIN, unless you use the FROM CATALOG option. When you specify FROM CATALOG, the TEST and EXPLAIN commands select the program names from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE by using the same logic as COMPARE and REPORT.

**Query number considerations**

DB2 Path Checker will identify and process only the most current results of a BIND EXPLAIN(YES) in the plan tables.

It uses the unique timestamp created by DB2 and the query number to find the end of a specific EXPLAIN. DB2 Path Checker assumes that query numbers are assigned in ascending sequence.

Attention:

If your DB2 programs use the QUERYNO clause in EXPLAIN statements or in SELECT, INSERT, UPDATE or DELETE statements, ensure that the numbers are assigned in a strict ascending sequence.

Related information:

IBM Support

See the DB2 SQL Reference for more information.

**COMPARE command**

Follow the syntax and descriptions in this topic to use the COMPARE command.
**COMPARE command syntax**

The **COMPARE** command determines which access paths changed between binds or between a bind and previously-generated explain results in a plan table.

```
COMPARE object-type qualifier1.program IN tablename1 TO qualifier2.* IN tablename2 START WITH programe BEFORE timestamp
```

You can use the SET CURRENT SCHEMA command with the COMPARE command.

- **object-type**
  - Indicates the type of object to compare. Specify either PLAN or PACKAGE.

- **qualifier1**
  - Indicates the name of the plan or collection to report on. This name is considered to be the new plan or collection.

- **program**
  - Indicates the name of the program to report on. This name can contain wildcard characters.

- **tablename1**
  - Indicates the name of the plan table that contains the explain data for qualifier1.program. This name is considered to be the new plan table.

- **qualifier2**
  - Indicates the name of the plan or collection to compare against. Its name can be the same as or different from the plan or collection named in qualifier1. This name is considered to be the old plan or collection.

  Always specify qualifier2 as qualifier2.*, because the same name or list of names from qualifier1.program1 is used for the comparison.

  The object type of qualifier2 must be the same as qualifier1. Plans must be compared to plans, and packages must be compared to packages. Comparisons between different object types are not permitted.

- **tablename2**
  - Indicates the name of the plan table containing the Explain information for qualifier2.*. This name is considered to be the old plan table.

- **programe**
  - START WITH is an optional clause that enables you to start a previous run at a specific point. If the program named programe exists, processing will begin with that program. If programe does not exist, processing will resume with the next higher name that does exist.

- **timestamp**
  - BEFORE is an optional clause that allows you to limit processing to a specific subset of the rows in the plan table. Only plan table entries bound before the date and time specified by timestamp are included in the report. This is especially useful when multiple binds with EXPLAIN were done and, as a result, the plan table contains multiple entries for a particular
plan or package. The timestamp column in the DB2 plan table is not formatted as it might be externally; rather, it is a 16-byte character field. For example, BEFORE '20021222421165' would run the report only for plan table rows that were created before the specified time.

To determine a timestamp value, examine the previous REPORT DIRECTORY run. The ending timestamp value from one of the lines in a previous DIRECTORY report is typically a good value to use. However, the value you specify does not need to match a timestamp in the plan table exactly.

The BEFORE option is mutually exclusive with PREVIOUS and COMPARE PLAN or PACKAGE NAME TO PREVIOUS VERSION. You can only specify one keyword for a request.

**PREVIOUS**

With the PREVIOUS clause, you can compare the current access path to the access path of a previous BIND instance or a specific version.

If you specify PREVIOUS for a plan or package, DB2 Path Checker compares the current access path to the access path of the next oldest (-1) BIND instance of that plan or package. The VERSION column value of the current plan or package will match the VERSION column value of the next oldest BIND instance of the plan or package.

If you specify PREVIOUS VERSION, DB2 Path Checker will ignore all prior BIND instances of the specified plan or package that contain the same VERSION column value, and compare the current access path to the access path of the next oldest version. The VERSION column value of the current plan or package will be different than the VERSION column value of the next oldest plan or package.

COMPARE PLAN or PACKAGE NAME TO PREVIOUS VERSION skips multiple BIND instances that contain the same VERSION column value and compares the access path for the plan or package to the next oldest version. This clause is useful if a program had multiple BIND or REBIND commands, and you need to compare to the previous software version of a program.

The report will identify all SQL statements as NEW SQL if:

- The `PLAN_TABLE` does not contain access path data for multiple versions.
- The program was not compiled with versioning, such as `VERSION (AUTO)` or a specific version.

In all cases, the BIND TIME column is used to identify the previous version for the specified plan or package. The VERSION column is never compared to the BIND TIME column as a time-stamp or date-stamp value.

The PREVIOUS option is mutually exclusive with BEFORE, COMPARE PLAN, and PACKAGE NAME TO PREVIOUS VERSION. If you specify more than one of these options, the last option specified is the one that takes effect, and the others are ignored.

**Related concepts:**

“What does DB2 Path Checker do?” on page 3

DB2 Path Checker helps database administrators and application developers to optimize the performance of SQL statements and maximize the efficiency of application development and implementation on DB2.
“Scenario: Deploy a new version of an application” on page 16
In this scenario, the bank deploys a new version of its ATM application. The IT staff wants to anticipate any potential performance issues by analyzing the changed access paths before putting the new version into production.

Related tasks:
“Comparing access paths for different binds of the same program version” on page 78
Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

“Comparing two bound versions of a program” on page 76
Use the COMPARE command to compare the access paths for the current version of a program with the access paths for the previous version.

CONNECT command
Follow the syntax and descriptions in this topic to use the CONNECT command.

CONNECT command syntax
Use the CONNECT command to connect to a DB2 subsystem.

```
CONNECT TO ssid
```

*ssid* Indicates the name of the DB2 subsystem where processing is to occur.

The CONNECT command must be the first executable command in SYSIN. The command must appear before the first REPORT, COMPARE, TEST, EXPLAIN, or MAKE command in a control file (SYSIN) or batch job.

A single job can connect to multiple subsystems, one at a time, by specifying the CONNECT command multiple times. The CONNECT command uses the options from the most recently specified OPTIONS command.

Related tasks:
“Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80
Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.

“Comparing access paths for different binds of the same program version” on page 78
Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

“Generating access path information for a DBRM” on page 70
Use the EXPLAIN DBRM command to populate your plan table with access paths that reflect the current state of your DB2 environment.

“Generating access path information for a package” on page 71
Use the EXPLAIN PACKAGE command to populate your plan table with access paths that reflect the state of your DB2 environment at the point in time when DB2 bound a package.

“Determining potential access path changes since the previous bind” on page 73
Use the TEST command to determine which access paths change if you rebind the
Use the `COMPARE` command to compare the access paths for the current version of a program with the access paths for the previous version.

**EXPLAIN command**

Follow the syntax and descriptions in these topics to use the `EXPLAIN` commands that are available in DB2 Path Checker.

`EXPLAIN` commands determine the access paths for a DBRM that is related to a plan or a package without comparing those access paths to any history. Optionally, `EXPLAIN` commands can also write access path data to a user-specified plan table that is different from the default plan table `owner.PLAN_TABLE`. You can save the results and then compare them later with the actual access path data from a bind. You can also rebuild access path history in the plan table if the data is lost.

**Related concepts:**
- “What does DB2 Path Checker do?” on page 3
- DB2 Path Checker helps database administrators and application developers to optimize the performance of SQL statements and maximize the efficiency of application development and implementation on DB2.

**Related tasks:**
- “Generating access path information for a DBRM” on page 70
  - Use the `EXPLAIN DBRM` command to populate your plan table with access paths that reflect the current state of your DB2 environment.
- “Generating access path information for a package” on page 71
  - Use the `EXPLAIN PACKAGE` command to populate your plan table with access paths that reflect the state of your DB2 environment at the point in time when DB2 bound a package.

**EXPLAIN DBRM command**

The `EXPLAIN DBRM` command does line-by-line `EXPLAIN` processing for each SQL statement that is associated with a DBRM.

**EXPLAIN DBRM command syntax**

If you specify a ddname for `SYSEXPLN`, the `EXPLAIN DBRM` command writes a detailed report to the `SYSEXPLN` data set. The `EXPLAIN DBRM` command is similar to a `REPORT` command for a new DBRM.

```
x---EXPLAIN---DBRM---dbrm-name---TO---object-type---qualifier1---
\[\[\[\[---IN---qualifier2---table-name---\]
```

`dbrm-name`
The name of the DBRM to explain. You can point to any DBRM library that contains the edition or version that you want to explain.

Wildcard characters are allowed in the name. If you use wildcard characters, standard TSO wildcard rules apply. DB2 Path Checker selects the DBRM to be tested from the DBRMs that are available in the DBRMIN concatenation.

TO

This optional clause specifies the DBRM for which DB2 Path Checker does EXPLAIN processing. If the bind used the EXPLAIN(YES) option, DB2 Path Checker also reads the matching cost information in the related DSN_STATEMNT_TABLE.

object-type

Indicates the type of object to process. Specify either PLAN or PACKAGE.

qualifier1

Specifies the name of the plan or package to explain, which is the same DBRM name that the precompile step used. For example, you can use this value as an alias for the DBRM name. DB2 Path Checker writes the value in the PROGNAME column of the plan table. For plans, you can specify any plan name or collection name. For packages, you can specify any valid program name.

IN

This optional clause saves the access path data to the plan table that you specify. If the bind used the EXPLAIN(YES) option, DB2 Path Checker also saves the matching cost information to the related DSN_STATEMNT_TABLE.

qualifier2

Indicates the name of the schema or the SQLID for the specified plan table.

table-name

The name of the plan table where the access path data is to be saved. It can be a different plan table than the default plan table owner.PLAN_TABLE.

FROM CATALOG

This optional clause selects SQL statements from the DB2 catalog. If a DBRM is not available, command processing constructs one from the data in the catalog; however, the data from a previous bind must be present in the catalog. To use this option, you must specify a ddname for DBRMOU in the JCL for the job.

You can use a wildcard character to specify the program name, but DB2 Path Checker selects the identifier for the collection or plan from the DB2 catalog table SYSPRM, SYSPDBRM or SYSPRM, SYSPACKAGE. DB2 Path Checker selects members for processing from a directory list that it constructs from the DBRMIN concatenation.

PACKAGE
Identifies the source collection from which to retrieve the SQL. The collection-id value indicates the identifier for the collection that references the package.

**PLAN**

Specifies the name of the plan from which to retrieve the SQL. The plan-name value specifies the name of the plan.

If you specify FROM CATALOG, you must also specify either PACKAGE or PLAN.

**Related concepts:**

“What does DB2 Path Checker do?” on page 3

DB2 Path Checker helps database administrators and application developers to optimize the performance of SQL statements and maximize the efficiency of application development and implementation on DB2.

**Related tasks:**

“Generating information about potential changes to access paths for plans and packages” on page 69

Generate information about access paths at different points in time to determine how changes in your DB2 environment might affect the performance of your SQL statements.

**EXPLAIN PACKAGE command**

The EXPLAIN PACKAGE command extracts the access path information for a package from the DB2 directory.

**EXPLAIN PACKAGE command syntax**

This command is available only if you are running DB2 Version 10 and later.

```
EXPLAIN PACKAGE package-name COLLECTION collection-name TO qualifier..table-name VERSION version-name
COPYID CURRENT

package-name

Indicates the name of the package to explain. Wildcard characters are allowed in the name. If you use wildcard characters, standard TSO wildcard rules apply. DB2 Path Checker selects the packages to be tested from the catalog table SYSPACKAGE.

collection-name

Indicates the name of the collection that references the package. DB2 Path Checker selects the package to be tested from the collection in the system catalog.

TO
```
This optional clause copies the EXPLAIN data to a different plan table and
statement table that you specify here. Both tables must exist under a
separate schema name from the default owner, for example,
`qualifier.PLAN_TABLE` and `qualifier.DSN_STMT_TABLE`.

This option can be useful in situations where you want to save access path
data to a plan table other than the default `OWNER.PLAN_TABLE`. The
EXPLAIN PACKAGE command writes access path data to the default plan
table as part of command processing. The owner of the default plan table
is the `OWNER` that you specify by issuing the SET CURRENT SQLID command.

`qualifier`
Indicates the name of the schema for the specified plan table.

`table-name`
The name of the plan table where DB2 Path Checker copies the
access path data. To prevent DB2 Path Checker from copying the
data in the default plan table onto itself, this value must specify a
different plan table than the default plan table `sql-id.PLAN_TABLE`.

**VERSION**

This optional clause indicates the version of the package that you want to
explain. If you do not specify a version, DB2 Path Checker writes explain
data for all versions of the package.

`version-name`
Specifies the version of the package.

If you did not specify VERSION(AUTO) as an input option to the
DB2 pre-processor or co-processor, this value is a user-specified
character sequence. If you used VERSION(AUTO), then the value
is an automatically generated consistency token in character format
(typically formatted as a time stamp, such as ‘2012122221165’).

Command syntax requires that you surround this value with single
quotation marks.

**COPYID**

Specifies the bind (edition or version) for which DB2 Path Checker does
explain processing. Any option for COPYID other than CURRENT
produces no data unless both of the following conditions are met:

- EXTENDED PLAN MANAGEMENT is active
- EXTENDED PLAN MANAGEMENT was in effect at the time of the
  previous two or three binds of the package

**CURRENT**
Indicates the most recent bind that used any EXPLAIN option other
than ONLY.

**PREVIOUS**
Indicates the bind immediately previous to the most recent bind
that used any EXPLAIN option other than ONLY. This bind can be a
previous edition of the current version or a previous version.

**ORIGINAL**
Indicates the oldest recorded bind that used any EXPLAIN option
other than ONLY.
The following example explains the package PGM01 in the collection ACCTPAY
and writes a copy of the access path data to the plan table
ADMF006.PLAN_TABLE. DB2 Path Checker processes the oldest recorded bind for
version ENV27 of the package.

```
EXPLAIN PACKAGE PGM01 COLLECTION ACCTPAY TO ADMF006.PLAN_TABLE
COPYID ORIGINAL VERSION 'ENV27'
```

Related concepts:
```
“What does DB2 Path Checker do?” on page 3
```
DB2 Path Checker helps database administrators and application developers to
optimize the performance of SQL statements and maximize the efficiency of
application development and implementation on DB2.

Related tasks:
```
“Generating information about potential changes to access paths for plans and
packages” on page 69
```
Generate information about access paths at different points in time to determine
how changes in your DB2 environment might affect the performance of your SQL
statements.

**MAKE command**

Follow the syntax and descriptions in this topic to use the **MAKE** command.

**MAKE command syntax**

```
MAKE STATEMENT nnn WITH TIMESTAMP='timestamp'
IN object-type qualifier.program IN tablename
BE HINT hintname
```

**nnn**
Is the QUERYNO of the plan table row that is to be made into a hint. The
maximum QUERYNO that you can specify is 32767.

**TIMESTAMP**
Is a value of the TIMESTAMP column that, combined with the QUERYNO,
uniquely identifies the row.

**object-type**
Indicates the type of object to use. Specify either PLAN or PACKAGE.

**qualifier**
Is the name of the plan or collection that contains **program**.

**program**
The name of the program that contains statement **nnn**.

**tablename**
The name of the plan table that contains the row that is to be made into a
hint.

**hintname**
Is the hint name that you assign to row **nnn**.

Related concepts:
```
“Scenario: Test a package against the previous BIND” on page 14
```
In this scenario, the IT staff at the bank is considering rebinding its mortgage
application after it adds several new indexes to their DB2 database.
OPTIONS command

Follow the syntax and descriptions in this topic to use the OPTIONS command.

OPTIONS command syntax

Use the OPTIONS command to set DB2 Path Checker processing options.
REPORTCHG

Displays only changed access paths. This option is the default.
REPORTCHG, REPORTCHGSBS, REPORTALL, and REPORTTERSE are mutually exclusive. If you specify more than one of these options, the last option specified is the one that takes effect, and the others are ignored.

REPORTCHGSBS
Displays access path changes in the SYSPRINT report in a more readable side-by-side table. Values in the table are displayed only if differences exist between corresponding columns in the PLAN_TABLE. Blank values in the table indicate that the columns in the PLAN_TABLE are the same for both the current and previous statements. New or deleted statements are listed after the table.

REPORTCHG, REPORTCHGSBS, REPORTALL, and REPORTTERSE are mutually exclusive. If you specify more than one of these options, the last option specified is the one that takes effect, and the others are ignored.

REPORTALL
Displays all access paths, regardless of whether they have changed.

REPORTCHG, REPORTCHGSBS, REPORTALL, and REPORTTERSE are mutually exclusive. If you specify more than one of these options, the last option specified is the one that takes effect, and the others are ignored.

REPORTQUIET
Displays the same information as the REPORTCHG option in a more compact format that reduces the volume of reported data by omitting CKP212I, CKP206I, and CKP203I messages. This option is an alias for the REPORTTERSE option, and produces the same results.

REPORT80COL
Limits the width of the report to 80 characters.

REPORTTERSE
Displays the same information as the REPORTCHG option in a more compact format that reduces the volume of reported data by omitting CKP212I, CKP206I, and CKP203I messages. This option is an alias for the REPORTQUIET option, and produces the same results.

REPORTCHG, REPORTCHGSBS, REPORTALL, and REPORTTERSE are mutually exclusive. If you specify more than one of these options, the last option specified is the one that takes effect, and the others are ignored.

MATCHSQL1
Matches SQL statements based on the SQL characteristics rather than by statement number or physical sequence in the program. This matching technique is the default.

MATCHSQL1 is mutually exclusive with MATCHSQL2, MATCHQUERYNO, and MATCHSEQUENCE. If you specify multiple options, DB2 Path Checker uses only the last one.

MATCHSQL2
The MATCHSQL2 processing option matches SQL statements based on the SQL characteristics followed by the SQL statement text rather than by statement number or physical sequence in the program.

Because DB2 Path Checker compares the SQL statement text in the catalog, this option works only for the following commands:
• COMPARE PACKAGE
• TEST AS PACKAGE
• TEST FROM CATALOG

When you use MATCHSQL2, the SQL statements for the previous bind of
the package must also be in the catalog. If the SQL statements are not in
the catalog, the report indicates that none of the statements match and
flags all statements as changed.

MATCHSQL2 might flag changes in the sequence of SQL statements in a
program, even if the statements are exact duplicates.

MATCHSQL2 is mutually exclusive with MATCHSQL1,
MATCHQUERYNO, and MATCHSEQUENCE. If you specify multiple
options, DB2 Path Checker uses only the last one.

MATCHSEQUENCE

Matches SQL statements sequentially, for example, 1st to 1st, 2nd to 2nd,
and so on.

MATCHSEQUENCE is mutually exclusive with MATCHSQL1,
MATCHSQL2, and MATCHQUERYNO. If you specify multiple options,
DB2 Path Checker uses only the last one.

MATCHQUERYNO

Matches SQL statements by QUERYNO value.

MATCHQUERYNO is mutually exclusive with MATCHSQL1,
MATCHSQL2, and MATCHSEQUENCE. If you specify multiple options,
DB2 Path Checker uses only the last one.

MATCHCREATOR

Includes the creator and access creator in the access path comparison. This
option is the default.

NOMATCHCREATOR

Excludes the creator and access creator in the access path comparison.

MATCHCREATOR and NOMATCHCREATOR are mutually exclusive. If you specify multiple options, DB2 Path Checker uses only the last one.

NOSQLERROR

Suppresses error messages for SQL statements in a DBRM that cannot be
explained, such as Selects from SYSDUMMY1. This option is the default.

MATCHTABNO

Matches tables by TABNO value. This option is the default.

MATCHCORRNAME

Matches tables by CORRNAME value.

NOSQLERROR

Produces no error messages for SQL statements in a DBRM that are not
explainable. This option is the default.

NOSQLERROR and SQLERROR are mutually exclusive. If you specify
multiple options, DB2 Path Checker uses only the last one.

SQLERROR
Produces error messages for SQL statements in a DBRM that are not explainable.

**CPUPCT percentage**

Requests that DB2 Path Checker identify packages that are safe to rebind. DB2 Path Checker considers the package safe to rebind if either of the following conditions are true:

- The access path did not change
- The PROCMS for the new access path is less than the PROCMS for the old access path plus the percentage that is specified by percentage

If you specify CPUPCT 0, DB2 Path Checker generates rebind commands only for changed access paths for which it estimates a lower PROCMS.

**CPUPCT2 percentage**

Requests that DB2 Path Checker identify packages that are safe to rebind. The CPUPCT2 percentage option limits rebind activity to only packages that will have an improved access path. DB2 Path Checker considers the package safe to rebind if the PROCMS for the new access path is less than the PROCMS for the old access path plus the percentage that is specified by percentage.

If you specify CPUPCT2 0, DB2 Path Checker generates rebind commands only for changed access paths for which it estimates a lower PROCMS.

**NOCATALOGSQL**

Specifies that the TEST DBRM command includes the SQL statement text, when it is available, in the explain reports that DB2 Path Checker writes to the SYSEXPLN data set. This option is the default.

NOCATALOGSQL and CATALOGSQL are mutually exclusive. If you specify both options, DB2 Path Checker uses only the last one.

**CATALOGSQL**

Specifies that the COMPARE and REPORT commands also include the SQL statement text, when it is available, in the explain reports that DB2 Path Checker writes to the SYSEXPLN data set.

**NOREPORTCOSTGT**

Specifies that only changed access paths are included in the change reports that DB2 Path Checker writes to the SYSCHG data set. This option is the default.

NOREPORTCOSTGT and REPORTCOSTGT are mutually exclusive. If you specify both options, DB2 Path Checker uses only the last one.

**REPORTCOSTGT**

Two SQL statements are included in change reports if either of the following conditions are true:

- The access path changes
- The estimated costs increase in the PROCSU and PROCMS columns in the DSN_STATEMNT_TABLE

Estimated costs in the PROCSU column are in service (SVC) units, and estimated costs in the PROCMS column measure CPU consumption in milliseconds.

**REPORTCOSTALL**
Specifies that DB2 Path Checker reports the cost of each SQL statement, both before and after.

**NOREPORTALL**

This setting disables REPORTCOSTALL.

**DEGREE1**

Specifies that TEST or EXPLAIN commands do not evaluate parallel processing access paths in SQL applications. This option is the default.

DEGREE1 and DEGREEANY are mutually exclusive. If you specify both options, DB2 Path Checker uses only the last one.

**DEGREEANY**

Specifies that TEST or EXPLAIN command evaluate parallel processing access paths in SQL applications if the subsystem is set up to use parallel processing.

DB2 Path Checker issues a SET CURRENT DEGREE=ANY statement before issuing EXPLAIN SQL statements to retrieve the access path for the application SQL. After retrieving the access path, it then resets current degree to 1 before processing continues.

DEGREEANY and DEGREE1 are mutually exclusive. If you specify both options, DB2 Path Checker uses only the last one.

**NOCURRENTVERSIONONLY**

Disables a previous setting of CURRENTVERSIONONLY. This option is the default.

**CURRENTVERSIONONLY**

This option extracts the highest version ID from SYSPACKAGE and compares only the access path data for that version of the package. CURRENTVERSIONONLY is useful when a mass rebind has occurred, and access paths exist in the plan table that have current BIND_TIME values but are for previous versions of the package.

By default, this option also sets the MATCHVERSION option.

**NOMATCHVERSION**

Use this option if you have used and want to disable the MATCHVERSION option. NOMATCHVERSION is not a default option, so it must be specified in order to disable MATCHVERSION.

**MATCHVERSION**

Compares each access path to the previous access path for the same version by using the version ID from the DBRM specified in a TEST command.

This option is useful when multiple rebinds have occurred, and access paths exist in the plan table that have current BIND_TIME values but are for different versions of the package. Use MATCHVERSION only for REBIND analysis, not for migration with new or changed DBRMs.

You must issue the NOMATCHVERSION command to disable MATCHVERSION after specifying this option.

**CREATE TABLES**
Specifies that if output tables do not exist when you issue **EXPLAIN** or **TEST**
commands, DB2 Path Checker creates the required `owner.PLAN_TABLE`
and `DSN_STATEMNT_TABLE`.

The tables created by this option do not specify a table space, are used
only for the duration of the current run, and are dropped when the process
is finished.

**PKGSTABIL**

Use this option to invoke plan management (PLANMGMT) features for
processing rebinds in DB2 Version 9 and later. This option allows rebinds
to preserve the original access paths that are associated with the package.
You can specify the following values.

**BASIC**

When you specify BASIC, DB2 Path Checker includes the
PLANMGMT(BASIC) option in any **REBIND** commands that it
writes to the RBINDOUT and PBINDOUT data sets.

The values for PKGSTABIL are mutually exclusive. If you specify
multiple options, DB2 Path Checker uses only the last one.

**EXTENDED**

When you specify EXTENDED, **REBIND** commands include the
PLANMGMT(EXTENDED) option in the RBINDOUT and
PBINDOUT data sets.

If you previously specified PKGSTABIL BASIC and rebind again,
specify PKGSTABIL EXTENDED to preserve the access paths for
the current bind. If you specify PKGSTABIL BASIC for the second
rebind, DB2 saves only the previous bind.

When you migrate to a different version of DB2, specify
PKGSTABIL EXTENDED if you have the SPT01 space to handle
the volume. When you specify PKGSTABIL EXTENDED, DB2 Path
Checker preserves the access paths from the original package. This
option is equivalent to specifying PKGSTABIL BASIC. By
specifying PKGSTABIL EXTENDED, you ensure that the original
access paths for the package are available until you are far enough
into the migration to free the original package safely.

**IGNORE**

The IGNORE option cancels any existing PKGSTABIL setting, and
subsequent **REBIND** commands do not include the PLANMGMT
option.

**OFF**

The OFF option ensures that rebinds do not include PLANMGMT
features. DB2 Path Checker includes the PLANMGMT(OFF) option
in any **REBIND** commands that it writes to the RBINDOUT and
PBINDOUT data sets.

The following examples show how you can use the PKGSTABIL option:

**Example 1: Generating a backup for the rebind of packages that you TEST**

To generate a backup for the rebind of packages that you TEST,
issue the following command:

```sql
OPTIONS CATALOGSQL NOMATCHCREATOR
OPTIONS PKGSTABIL BASIC
```
This command generates the following statement to rebind cards in the RBINDOUT data set:

REBIND PACKAGE (ACCTPAYX.TEST01.(2005-04-11-18.34.53.754785)) -
EXPLAIN(YES) -
PLANMGMT(BASIC)

REBIND PACKAGE (ACCTPAYX.TEST02.(2005-04-11-18.35.13.673978)) -
EXPLAIN(YES) -
PLANMGMT(BASIC)

Example 2: Generating an original package as a copy of the current bind to hold in reserve and a new bind for to run in DB2

To generate an original package as a copy of the current bind to hold in reserve and a new bind to run in DB2, issue the following command:

OPTIONS CATALOGSQL NOMATCHCREATOR
OPTIONS PKGSTABIL EXTENDED

This command generates the following statement to rebind cards in the RBINDOUT data set:

REBIND PACKAGE (ACCTPAYX.TEST01.(2005-04-11-18.34.53.754785)) -
EXPLAIN(YES) -
PLANMGMT(EXTENDED)

REBIND PACKAGE (ACCTPAYX.TEST02.(2005-04-11-18.35.13.673978)) -
EXPLAIN(YES) -
PLANMGMT(EXTENDED)

DDNAMERC

This option changes the return code for the message CKP044E: DB2 Path Checker failed to create the temporary DBRM for a TEST or EXPLAIN command.

OPTIONS DDNAMERC=return-code specifies the return code that DB2 Path Checker uses when a ddname for DBRMOUT is not available. The default return code is 0 because DB2 Path Checker continues to process the next command

ABENDRC16

This option forces an ABEND when the return code from DB2 Path Checker is 16.

CCSID

This option specifies the target CCSID for conversion from DB2 Version 8 Unicode SQL text. Because this process uses z/OS conversion services, the CCSID value must be a valid CCSID on the z/OS system that runs DB2 Path Checker.

You need this option only when both of the following conditions are true:

- The DBRMs that you are processing were created by the precompiler from DB2 Version 8 or later
- The target character set that you want to use for the SQL statements that are in the explain output is not 0057

Attention: If the target character set is not CCSID 0057, the conversion service must be available to DB2 Path Checker at run time or the conversion will fail.

NOSEQBYVERSION
Disables the SEQBYVERSION option. This option is the default.

**SEQBYVERSION**

This option is the only way to process comparisons for multiple editions or versions of a package. SEQBYVERSION selects the access paths by version and compares the most recent BIND_TIME access path to the previous access path for that version.

SEQBYVERSION is useful when a mass rebind has occurred, and access paths exist in the plan table that have current BIND_TIME values but are for previous versions of the package.

**APPLCOMPAT**

**V11R1**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APPLCOMPAT(V11R1) option. For more information, see the DB2 documentation for the REBIND option APPLCOMPAT.

**V10R1**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APPLCOMPAT(V10R1) option. For more information, see the DB2 documentation for the REBIND option APPLCOMPAT.

If you specify this keyword in the OPTIONS command, you must specify a value or the command will fail.

**HISTORY TABLE qualified-table-name**

This option sets the name of the table that DB2 Path Checker uses to accumulate history information across multiple jobs. The history table stores a summary of the information that is in the change reports that DB2 Path Checker writes to the SYSCHG data set.

**NOTFOUNDRC return-code**

This option sets the return code that DB2 Path Checker uses when it issues the CKP205I message. This message is displayed when DB2 Path Checker cannot find any data in the plan table that matches your selection or masking criteria. See the message description for CKP205I.

**MATCHRC return-code**

This option sets the return code that DB2 Path Checker uses when all compared access paths match.

**NOMATCHRC return-code**

This option sets the return code that DB2 Path Checker uses when any compared access paths do not match.

**COMMITCOUNT number_of_objects**

This option specifies the number of objects, such as DBRMs, catalogs, or explain tables, that DB2 Path Checker commands access between COMMIT actions. Each program name (DBRM) that a command processes counts as a single unit of work. You can save costs that are related to commit processing by performing fewer commits. By performing more commits, you can release more DB2 locks that can prevent concurrent access to DBRMs, catalogs, or explain tables.
The DEFAULT value is zero (0), which disables commits. When the value is zero, DB2 holds any locks until the job ends.

DB2 Path Checker issues a commit and a message after processing each access or DBRM. If you specify a low number of objects, the effects on CPU consumption are minimal. If you specify too many objects, command processing withholds more DB2 resources from other jobs, increasing the risk of timeouts.

The following examples show how COMMITCOUNT works:

- If COMMITCOUNT = 1 and there are 1000 DBRMs, command processing issues 1000 commits and 1000 messages.
- If COMMITCOUNT = 100 and there are 1000 DBRMs, command processing issues a commit and a message after processing 100 objects, for a total of 10 commits and 10 messages.
- If COMMITCOUNT = 100 and there are 999 DBRMs, a commit and a message are issued after processing 100 objects, so command processing issues nine commits and nine messages. Termination processing for the program commits the last 99 DBRMs without issuing a message.

### APREUSE

**NONE**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APREUSE(NONE) option. For more information, see the DB2 documentation for the REBIND option APREUSE.

**WARN**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APREUSE(WARN) option. For more information, see the DB2 documentation for the REBIND option APREUSE.

**ERROR**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APREUSE(ERROR) option. For more information, see the DB2 documentation for the REBIND option APREUSE.

Set the DB2 option PLANMGMT(EXTENDED) with this option to preserve access path information from previous binds.

### APCOMPARE

**NONE**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APCOMPARE(NONE) option. For more information, see the DB2 documentation for the REBIND option APCOMPARE.

**WARN**

Specifies that the DB2 REBIND commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APCOMPARE(WARN) option. For more information, see the DB2 documentation for the REBIND option APCOMPARE.

**ERROR**
Specifies that the DB2 **REBIND** commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the APCOMPARE(ERROR) option. For more information, see the DB2 documentation for the **REBIND** option APCOMPARE.

Set the DB2 option PLANMGMT(EXTENDED) with this option to preserve access path information from previous binds.

**DESCSTAT**

*NO*

This default setting specifies that the DB2 **REBIND** commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the DECSTAT(NO) option. For more information, see the DB2 documentation for the **REBIND** option DECSTAT.

*YES*

Specifies that the DB2 **REBIND** commands that DB2 Path Checker writes to the RBINDOUT and PBINDOUT data sets include the DECSTAT(YES) option. For more information, see the DB2 documentation for the **REBIND** option DECSTAT.

If you specify this keyword in the OPTIONS command, you must specify a value or the command will fail.

Related tasks:

- “Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80
  Use the **TEST** command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.

- “Comparing access paths for different binds of the same program version” on page 78
  Use the **COMPARE** command to determine which access paths changed between binds of the same version of a plan or package.

- “Generating access path information for a DBRM” on page 70
  Use the **EXPLAIN DBRM** command to populate your plan table with access paths that reflect the current state of your DB2 environment.

- “Generating access path information for a package” on page 71
  Use the **EXPLAIN PACKAGE** command to populate your plan table with access paths that reflect the state of your DB2 environment at the point in time when DB2 bound a package.

- “Determining potential access path changes since the previous bind” on page 73
  Use the **TEST** command to determine which access paths change if you rebind the current version of a DBRM.

- “Comparing two bound versions of a program” on page 76
  Use the **COMPARE** command to compare the access paths for the current version of a program with the access paths for the previous version.

Related reference:

- “How to interpret reports” on page 103
  The reports that DB2 Path Checker produces use column headings that match fields in the default plan table (**owner**.PLAN_TABLE).

**REPORT command**

Follow the syntax and descriptions in this topic to use the **REPORT** command.
**REPORT command syntax**

The REPORT command generates one of the following access path reports:

- The default summary report displayed in ddname SYSPRINT.
- An optional detail report displayed in ddname SYSEXPLN. This report is not produced if the ddname is not present.
- The directory report, which is a summary of the bind and Explain activity as shown in the plan table. The directory format is used to report the results of BIND and EXPLAIN processing. The plan table that is used for the report can be any valid DB2 PLAN_TABLE.

![REPORT command syntax diagram]

**object-type**
Indicates the type of object to use. Specify either PLAN or PACKAGE.

**qualifier**
Indicates the name of the plan or collection to report on.

**program1**
Indicates the name of the program to report on. The name can contain wildcard characters.

**tablename**
Indicates the name of the plan table to use for generating reports. This parameter is required for the first request in a job step; after that, it is optional. If it is omitted, DB2 Path Checker uses the plan table from the previous request. If this parameter is omitted, also omit the keyword IN.

The plan table name, if specified, can or cannot have the creator ID (be fully qualified). If the creator ID is missing, DB2 Path Checker uses the creator ID from the previous request, or, if this request is the first, the creator ID from the last valid SET CURRENT SQLID command.

**program2**
START WITH is an optional clause that you use to start a report at a specific point. If the specified program does not exist, the report will start with the next name, in sequence, that does exist. For example, START WITH ABB would skip program ABA099 and begin processing with program ABB011.

**timestamp**
BEFORE is an optional clause that allows you to limit processing to a specific subset of the rows in the plan table. Only plan table entries bound before the date and time specified by timestamp will be included in the report. Including only this type of plan table entry in the report is especially useful when multiple binds with EXPLAIN were done and, as a result, the plan table contains multiple entries for a particular plan or package. The timestamp column in the DB2 plan table is not formatted as it might be externally; rather, it is a 16-byte character field. For example,
BEFORE '2002121222421165' would run the report only for plan table rows that were created before the specified time.

To determine a timestamp value examine the previous REPORT DIRECTORY run. The ending timestamp value from one of the lines in a previous DIRECTORY report is typically a good value to use. However, the value you specify does not need to match a timestamp in the plan table exactly.

DIRECTORY

This keyword is optional and generates a summary report that shows the timestamps for each bind of the selected plans or packages, assuming that EXPLAIN was specified at bind time. The summary report shows the beginning and ending timestamps for each bind. This report is especially useful for determining appropriate timestamp values to specify with the BEFORE keyword.

**SET command**

Follow the syntax and descriptions in this topic to use the SET command.

**SET command syntax**

Use the SET command to set the current SQLID, catalog qualifier, or schema. The SET command syntax has three forms. Ensure that you put a blank before and after the equal sign (=).

**SET command syntax, format 1**

```
SET CURRENT SQLID='sqlid'
```

*sqlid* A valid authorization ID. Normal DB2 rules for SQLID apply; the user who submits the job must have SYSADM authority or valid RACF (or equivalent) access to the group.

The specified SQLID will also be used as the default qualifier for the plan tables for the REPORT, COMPARE, and TEST commands. (If a request explicitly specifies a qualifier for a plan table name, it will override the SQLID from the SET command.)

You must issue this command before a TEST command can be processed. Also, the SQLID specified here will be used as the default qualifier (creator ID) for plan tables used by REPORT and COMPARE processing. (If a request specifies an explicit qualifier, that value will override the SQLID in the SET command.)

**SET command syntax, format 2**

```
SET CATALOG QUALIFIER='catalog-qualifier'
```

User's Guide
catalog-qualifier

The appropriate high-level qualifier for a shadow catalog.

You must issue this command if a shadow catalog is being used. The user who submits the job must have SELECT access to the catalog. This command is not required when the DB2 default catalog qualifier is SYSIBM.

Note:

1. The normal DB2 rules for SQLID apply. The user who submits the job must have valid RACF (or equivalent) access to the group or SYSADM.

2. Spaces are required before and after the equal sign (=). The specified SQL will also be used as the default qualifier for the plan tables for the REPORT, COMPARE, and TEST commands. (If a request explicitly specifies a qualifier for a plan table name, it will override the SQLID from the SET command.)

SET command syntax, format 3

```
SET CURRENT SCHEMA='schema'
```

schema  A valid schema name. SET CURRENT SCHEMA changes the default qualifier for SQL statements that do not specify a creator ID. The qualifier is changed from the value that is specified in SET CURRENT SQLID = sqlid to the value that is specified for schema.

SET CURRENT SCHEMA affects only TEST and EXPLAIN command processing, REPORT and COMPARE commands are not affected by the SCHEMA setting. See DB2 for z/OS SQL Reference for more information about the impact of SET CURRENT SCHEMA and the security requirements.

Related tasks:

“Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80
Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.

“Comparing access paths for different binds of the same program version” on page 78
Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

“Generating access path information for a DBRM” on page 70
Use the EXPLAIN DBRM command to populate your plan table with access paths that reflect the current state of your DB2 environment.

“Generating access path information for a package” on page 71
Use the EXPLAIN PACKAGE command to populate your plan table with access paths that reflect the state of your DB2 environment at the point in time when DB2 bound a package.

“Determining potential access path changes since the previous bind” on page 73
Use the TEST command to determine which access paths change if you rebind the current version of a DBRM.
Use the **COMPARE** command to compare the access paths for the current version of a program with the access paths for the previous version.

**TEST command**

Follow the syntax and descriptions in this topic to use the **TEST** command.

**TEST command syntax**

The **TEST** command typically evaluates an unbound DBRM and the current edition or version of the package or plan. The **TEST** command generates reports that tell you which access paths will change if you rebind.

If a **SET CURRENT SQLID** command does not specify a valid authorization ID prior to issuing the **TEST** command, the test will fail. In addition, any qualifier that you specify for a plan table name in the **TEST** command overrides the qualifier that you specify by issuing **SET CURRENT SQLID**. In either case, the **TEST** command requires that you specify **SET CURRENT SQLID**.

```
TEST DBRM program1 AS object-type qualifier.* IN tablename
START WITH program2 BEFORE timestamp FROM CATALOG
```

**program1**

Is the name of the program to test. This name can contain wildcard characters. The name you specify is the member name to be read from the DBRM library specified in the DBRMIN DD statement.

**object-type**

Indicates the type of object to use. Specify either PLAN or PACKAGE.

**qualifier.***

Is the name of the plan or collection to test.

Always specify qualifier as qualifier.*, because the same name or list of names from **program1** is used to generate fresh explain data and to perform the comparison.

**tablename**

Is the name of the plan table to use.

**program2**

If you specify the **START WITH** keywords to start a previous run at a specific point, **program2** identifies the program where you want processing to begin.

If the program (DBRM) named **program2** exists, processing will begin with that program. If **program2** does not exist, processing will resume with the next higher name that does exist.

**timestamp**
BEFORE is an optional clause that allows you to limit processing to a specific subset of the rows in the plan table. Only plan table entries bound before the date and time specified by timestamp will be included in the report. Including only this type of plan table entry in the report is especially useful when multiple binds with EXPLAIN were done and, as a result, the plan table contains multiple entries for a particular plan or package. The timestamp column in the DB2 plan table is not formatted as it might be externally; rather, it is a 16-byte character field. For example, BEFORE '20021212224242165' would run the report only for plan table rows that were created before the specified time.

To determine a timestamp value, examine the previous REPORT DIRECTORY run. The ending timestamp value from one of the lines in a previous DIRECTORY report is typically a good value to use. However, the value you specify does not need to match a timestamp in the plan table exactly.

FROM CATALOG
This parameter tells TEST to use the SQL statements in the DB2 catalog instead of a DBRM as input to the TEST command.

//RBINDOUT
Indicates the ddname that will contain REBIND commands when a TEST command for a package does not identify an access path change or the new PROCMS is less than the old PROCMS plus the percentage that is specified in OPTIONS CPUPCT or CPUPCT2.

//PBINDOUT
Indicates the ddname that will contain REBIND commands when a TEST command for a package identifies an access path change and OPTIONS CPUPCT2 was specified or the new PROCMS is more than the old PROCMS plus the percentage that was specified in OPTIONS CPUPCT or CPUPCT2.

Related concepts:
“What does DB2 Path Checker do?” on page 3
DB2 Path Checker helps database administrators and application developers to optimize the performance of SQL statements and maximize the efficiency of application development and implementation on DB2.

“Scenario: Test a package against the previous BIND” on page 14
In this scenario, the IT staff at the bank is considering rebinding its mortgage application after it adds several new indexes to their DB2 database.

Related tasks:
“Determining potential access path changes since the previous bind” on page 73
Use the TEST command to determine which access paths change if you rebind the current version of a DBRM.

“Testing an unbound DBRM against the previous program version” on page 75
Use the TEST command to compare a DBRM that you did not bind yet against the previous version of the package.

“Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80
Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.
Sample job to send SQL statements to SQL Performance Analyzer

This sample job shows you how to send SQL statements that have changed access paths to DB2 SQL Performance Analyzer for additional cost analysis.

About this task

You can run a batch job like the one in the sample to transfer data from DB2 Path Checker to DB2 SQL Performance Analyzer. The ANLOUT DD statement in the step PTHCHK1 allocates a temporary data set to which DB2 Path Checker writes SQL statements that have changed access paths. DB2 SQL Performance Analyzer uses the same data set as input. The ANLIN DD statement in the step ANLSTEP1 is a backward reference to the same data set.
//USERIDCR JOB(P390V6),'PTHCK-SQLPA',CLASS=A,MSGCLASS=X,REGION=8M,
// NOTIFY=&SYSUID
//
//****************************************************************************
// ** PATH CHECKER **
//****************************************************************************
//
//PTHCHK1 EXEC PGM=CKPPTHCK,
// PARM='DSN1,CUSTOMER,USERID,PLAN_TABLE,ORDERS,ORDERS,ALL'
//STEPLIB DD DSN=db2.SDSNLOAD,DISP=SHR
// DD DSN=ckp.SCKPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//DBRMIN DD DSN=ckp.SCKPDBRM,DISP=SHR
//
//ANLOUT DD DSN=&ANLOUT,DISP=(NEW,PASS),
// SPACE=(TRK,(5,1),RLSE),DCB=(LRECL=80,RECFM=FB,BLKSIZE=4000)
//
//****************************************************************************
// ** SQL PERFORMANCE ANALYZER **
//****************************************************************************
//
//** STEP 0: DELETE THE PREVIOUS RUN'S PERMANENT REPORTS, IF NECESSARY **
//
//ANLSTEP0 EXEC PGM=IEFBR14
//GOAWAY1 DD DSN=USERID.ANLREP.RPT,DISP=(MOD,DELETE)
//GOAWAY2 DD DSN=USERID.QTRACE.RPT,DISP=(MOD,DELETE)
//GOAWAY3 DD DSN=USERID.QLIMIT.RPT,DISP=(MOD,DELETE)
//
//** IN NORMAL RUNS, THESE MIGHT BE RENAMED, SAVED OR NOT CREATED AT ALL **
//
//****************************************************************************
// ** STEP 1: PARSER SCANS THE SQL LIBRARY OF CHOICE (ANLIN) FOR STMTS **
// ** AND REGISTERS THE ANLUSER; EXPLAIN READS THE PLAN TABLE'S **
// ** RECORDS AND LOOKS UP THE TABLES & INDEXES IN THE CATALOG. **
//****************************************************************************
//
//ANLSTEP1 EXEC PGM=ANLSCAN
//STEPLIB DD DSN=SYSx.ANL110.SANLLOAD,DISP=SHR
// DD DSN=db2.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//ANLCNTL DD DSN=USERID.UTIL.CNTL(ANLCNTL),DISP=SHR
//ANLPARM DD DSN=USERID.UTIL.CNTL(ANLPARM),DISP=SHR
//*******
//ANLIN DD DSN=.PTHCHK1.ANLOUT,DISP=(OLD,DELETE)
//*******
//ANLWORK DD DSN=&ANLWORK,DISP=(DELETE,DELETE),
// SPACE=(TRK,(5,1),RLSE),DCB=(LRECL=80,RECFM=FB,BLKSIZE=4000)
//ANLOUT DD DSN=&ANLOUT,DISP=(NEW,PASS),
// SPACE=(TRK,(5,1),RLSE),DCB=(LRECL=80,RECFM=FB,BLKSIZE=9040)
//ANLPAS DD DSN=&ANLPAS,DISP=(NEW,PASS),
// SPACE=(TRK,(5,1),RLSE),DCB=(LRECL=140,RECFM=FB,BLKSIZE=11200)
//ANLSEP DD DSN=&ANLSEP,DISP=(NEW,PASS),
// SPACE=(TRK,(1,1),RLSE),DCB=(LRECL=257,RECFM=FB,BLKSIZE=11308)
//ANLREL DD DSN=&ANLREL,DISP=(NEW,PASS),
// SPACE=(TRK,(1,1),RLSE),DCB=(LRECL=83,RECFM=FB,BLKSIZE=11869)
//SYSPRINT DD SYSOUT=*,DCB=LRECL=133
//
//****************************************************************************
// ** STEP 2: COMPUTE RESOURCE CONSUMPTION & PRODUCE SQL PA COST REPORT **
//****************************************************************************
//
//ANLSTEP2 EXEC PGM=ANLCOAST
//STEPLIB DD DSN=SYSx.ANL110.SANLLOAD,DISP=SHR
// DD DSN=db2.SDSNLOAD,DISP=SHR

Figure 45. Sample JCL for DB2 Path Checker SQL PA interface, Part 1
Figure 46. Sample JCL for DB2 Path Checker SQL PA interface, Part 2

An example of the output is shown in the following figure:
Figure 47. Sample SQL PA report on DB2 Path Checker SQL, Part 1
SQL PA shows the DBRM member name and the reason why DB2 Path Checker selected the SQL, if available, in its standard reporting format.

Job control statements

Job control statements specify DB2 Path Checker functions and what options to use.

ANLOUT DD
This DD statement is optional. If present, and if a TEST or COMPARE command is processed for a changed access path, the generated output is written to this ddname. When you process a TEST or COMPARE command for a changed access path, the output can be directed to another software product such as IBM DB2 SQL Performance Analyzer for z/OS. The ANLOUT DD statement only works with the COMPARE command if the full SQL is available in SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT.

If ANLOUT is specified, a COMPARE command detects a difference and the SQL text is available in SYSSTMT or SYSPACKSTMT, the SQL statement will be written to ANLOUT for SQL Performance Analyzer processing.

BINDIN DD
This DD statement is optional. It generates the list of plans or packages to be processed from bind control cards. DB2 Path Checker parses the Bind control cards and performs the PATH CHECK command from SYSIN for each plan or package that is specified in the BIND control cards. DB2 Path Checker ignores the plan and package name in the command and identifies the plan or package to be processed from the BIND control commands.
Only one command can be processed with BINDIN controlling the plan, collection, and program name. Processing only one command with BINDIN in control is useful when you use DB2 Path Checker with an existing process that creates and processes BIND and REBIND commands.

**DBRMIN DD**

This DD statement is required for the TESTPKG or CUSTOMER PARM requests or TEST or EXPLAIN commands. DBRMIN identifies the partitioned data set that contains the DBRM to be analyzed by the TEST command, the EXPLAIN command, or the TESTPKG and CUSTOMER parameters. If you specify more than one DBRM library, this DD statement can be concatenated. You specify the member name in the command or parameter by using the `program` variable.

**DBRMOUT DD**

This DD statement is used for the TEST and EXPLAIN commands that specify the FROM CATALOG option. It identifies the temporary or permanent data set that is used to rebuild the DBRM from the catalog, and it is ignored for all other functions.

This DD statement must have DCB=(RECFM=FB,LRECL=80,BLKSIZE=0) specified for DCB and must be a valid QSAM file such as a temporary, VIO, or permanent file. It cannot be a JES spool file. It must be allocated with DISP=OLD, SHR, or NEW. If DB2 Path Checker is allocated with DISP=MOD, DB2 Path Checker does not work correctly. It must be allocated with DISP=NEW or OLD.

**EXEC**

Starts DB2 Path Checker.

The format of the EXEC statement is:

```
EXEC PGM=CKPPTHCK,PARM='parameters'
```

**EXEC statement syntax**

```
EXEC statement syntax
//stepname--EXEC PGM=CKPPTHCK ,PARM='parameters'
```

**PGM=CKPPTHCK**

Indicates that you want to run the DB2 Path Checker program

**PARM=’parameters’**

Specifies the parameters that you want DB2 Path Checker to use. You can specify the parameters here, or you can specify them as commands that are read from SYSIN. If you specify both parameters and commands, all PARM field parameters are ignored.

The parameters that you can specify are:

`’ssid,request,creator,table,entity,prognames,report’`

The parameters are positional, and must be specified in the order shown. You can specify values for any valid combination of parameters. If any parameter (except the final one) is omitted, include the trailing comma that normally would follow it. You can specify the following individual parameters:

- `ssid` Specifies the DB2 subsystem name
request Specifies the type of processing. Select one of the following values:

RPTPKG
Read the current Explain information from the plan table and format a report for the specified package

RPTPLAN
Read the current Explain information from the plan table and format a report for the specified plan

COMPPKG
Compare and contrast Explain data from two plan tables for a specified package

COMPLAN
Compare and contrast Explain data from two plan tables for a specified plan

TESTPKG
Read DBRMLIB member and modify SQL for all explainable SQL to issue EXPLAIN PLAN with correct QUERYNO, then process same as COMPPKG

CUSTOMER
Read DBRMLIB member and modify SQL for all explainable SQL to issue EXPLAIN PLAN with correct QUERYNO, then process same as COMPLAN

creator Specifies the qualifier for the plan table name

table Specifies the name of the secondary plan table

entity Specifies the name of the plan or collection to be analyzed

proname Specifies the program name (DBRM name) to be processed

report Optional. Generates a report that lists all access paths, not just the ones that have changed. If you specify this parameter, the only valid value is ALL.

If you omit this parameter, a long (two lines) or short (one line) report is produced depending on the actual data being displayed.

JOB Starts the job.

RBINDOUT DD
This DD statement identifies an output data set that will contain recommended REBIND commands for every TEST DBRM AS PACKAGE command that completes successfully. RBINDOUT identifies these output data sets only when one of the following conditions is true:

• The TEST DBRM AS PACKAGE command did not identify any access path changes.
• The TEST DBRM AS PACKAGE command identified an access path change, but the change is less than the percentage that was specified in OPTIONS CPUPCT.
• OPTIONS CPUPCT2 was specified, the TEST DBRM AS PACKAGE command identified access path changes, and the change in PROCMS is less than the percentage that was specified in OPTIONS CPUPCT2.

The data set must be DCB=(RECFM=FB,LRECL=80,BLKSIZEx), where x is 0 to use a block size that is determined by the system or a valid multiple of 80.

This ddname is required when OPTIONS CPUPCT or OPTIONS CPUPCT2 is specified. If this ddname is specified without OPTIONS CPUPCT or OPTIONS CPUPCT2, this ddname captures optional REBIND commands from a TEST command.

The REBIND command will be in the following format:

REBIND PACKAGE (collid.progname.(version))-
OWNER (owner)-
EXPLAIN(YES)

PBINDOUT DD
This DD statement identifies an output data set that will contain REBIND commands that are not recommended for every TEST DBRM AS PACKAGE command that completes successfully. PBINDOUT identifies these output data sets only when one of the following conditions is true:

• OPTIONS CPUPCT was specified, and the PROCMS exceeds the value of the old PROCMS plus the percentage that was specified in OPTIONS CPUPCT.
• OPTIONS CPUPCT2 was specified, the TEST DBRM AS PACKAGE command identified an access path change, and the change in PROCMS exceeds the percentage that was specified in OPTIONS CPUPCT2.
• OPTIONS CPUPCT2 was specified, and there were unmatched SQL statements.
• OPTIONS CPUPCT2 was specified, and there were no access path changes.

The data set must be DCB=(RECFM=FB,LRECL=80,BLKSIZEx), where x is 0 to use a block size that is determined by the system or a valid multiple of 80.

The PBINDOUT ddname is optional, but it can be used only with OPTIONS CPUPCT or OPTIONS CPUPCT2. If PBINDOUT is not specified, the REBIND commands for packages that are not recommended to be rebound are not written.

The REBIND command will be in the following format:

REBIND PACKAGE (collid.progname.(version))-
OWNER (owner)-
EXPLAIN(YES)

When OPTIONS PKGSTABIL is specified, the format of the REBIND command includes a PLANMGMT statement that is based on the PKGSTABIL setting.

SYSCHG DD
This statement is optional. If present, it is used to write a summary list of the SQL statements that have changed access paths. You can assign SYSCHG to SYSOUT or to any sequential data set.

SYSEXPLN DD
This statement is optional. If present, it defines where DB2 Path Checker
will write a detailed Explain report for each SQL statement that is processed. This report is in an expanded format that is easier to read than the terse one- or two-line reports written to SYSPRINT.

**SYSIN DD**
This statement is optional. Defines the optional control data set where you provide commands to DB2 Path Checker. If present, it must contain valid DB2 Path Checker commands. If the SYSIN DD statement is omitted, the JCL PARM field must supply all necessary operating parameters. If both SYSIN and PARM input are supplied, the SYSIN information is used, and the PARM field is ignored.

SYSIN must be in format FB, with LRECL=80. The SYSIN statement can specify DD * or an input file.

**SYSPRINT DD**
Defines a sequential data set that is used for listing control statements, messages, and output reports. DB2 Path Checker writes a log of the control statements and its actions to the SYSPRINT DD statement and writes output reports to SYSPRINT (with one exception, see SYSEXPLN DD). You can assign SYSPRINT to SYSOUT or to any sequential data set.

**XMLSQLTX DD**
This statement is optional. It contains the SQL text for any SQL statements that are processed by the TEST command and have changed access paths. The data will be an XML document that can be imported as a workload into Eclipse-based tuning products, such as IBM Optim™ Query Workload Tuner for z/OS and IBM Data Studio.

Related tasks:

"Comparing access paths for a DBRM with access paths that you generated by issuing EXPLAIN commands” on page 80
Use the TEST command to determine which access paths change if you rebind in situations where no EXPLAIN data is available for the previous bind.

"Comparing access paths for different binds of the same program version” on page 78
Use the COMPARE command to determine which access paths changed between binds of the same version of a plan or package.

"Generating access path information for a DBRM” on page 70
Use the EXPLAIN DBRM command to populate your plan table with access paths that reflect the current state of your DB2 environment.

"Generating access path information for a package” on page 71
Use the EXPLAIN PACKAGE command to populate your plan table with access paths that reflect the state of your DB2 environment at the point in time when DB2 bound a package.

"Running DB2 Path Checker by submitting batch jobs” on page 67
Run DB2 Path Checker by submitting a batch job that contains the required load libraries, DD statements, options, and commands.

"Determining potential access path changes since the previous bind” on page 73
Use the TEST command to determine which access paths change if you rebind the current version of a DBRM.

"Comparing two bound versions of a program” on page 76
Use the COMPARE command to compare the access paths for the current version of a program with the access paths for the previous version.
DB2 Path Checker parameters

You can specify DB2 Path Checker parameters either by using the PARM field or by specifying them as commands that are read from SYSIN. If you specify both parameters and commands, all PARM field parameters are ignored.

This function is supported for compatibility with previous releases, but does not support all the functionality of the SYSIN commands.

The first six parameters passed to DB2 Path Checker are positional and are separated by a comma (,) or one or more spaces. The last parameter, ALL, is a keyword. The following example shows the syntax for the PARM field:

```
PARM='ssid,request,creator-id,secondary-plan-table,
plan-name or collection-id,program,ALL'
```

**ssid**

Is the name of the DB2 subsystem where processing is to occur.

**request**

Indicates the type of processing. Specify one of the following values:

- **RPTPKG**
  
  DB2 Path Checker reads the plan table information from PLAN_TABLE and creates a report for the specified collection identifier and program name.

- **RPTPLAN**
  
  DB2 Path Checker reads the plan table information from PLAN_TABLE and creates a report for the specified plan name and program name.

- **COMPPKG**
  
  DB2 Path Checker compares the results of two previous binds of a package, one from PLAN_TABLE and one from the secondary plan table. The collection identifier and program name are specified in the PARM field.

- **COMPPLAN**
  
  DB2 Path Checker compares the results of two previous binds of a plan, one from PLAN_TABLE and one from the secondary plan table. The plan name and program name are specified in the PARM field.

- **TESTPKG**
  
  DB2 Path Checker reads the DBRMLIB member identified by the *program* parameter, issue an EXPLAIN with the correct query numbers for all the explainable SQL statements to populate PLAN_TABLE, and then performs a package compare (COMPPKG) against the secondary plan table.

- **CUSTOMER**
  
  DB2 Path Checker reads the DBRMLIB member identified by the *program* parameter, issue an EXPLAIN with the correct query numbers for all the explainable SQL statements to populate PLAN_TABLE, and then perform a plan compare (COMPPLAN) against the secondary plan table.

**creator-id**

The qualifier for the names of the plan tables used for the requested function. For TESTPKG and CUSTOMER, this identifier must also have authority to execute the SQL, allowing DB2 Path Checker to use dynamic EXPLAIN.
secondary-plan-table
The name of the plan table that will be compared with the contents of
creator-id.PLAN_TABLE for COMPPKG, COMPPLAN, TESTPKG, and
CUSTOMER requests.

plan-name or collection-id
Because DB2 will store either a plan name (APPLNAME) for a plan DBRM or
a collection identifier (COLLID) for a package DBRM, DB2 Path Checker uses
the same field for either value. The type of request identifies whether the value
is a plan name or a collection identifier.

program
The name of the program (DBRM) that is being compared between two
existing plan tables or that is being reported on. DB2 Path Checker uses
the program-name value to identify the DBRM to be processed for report and
compare requests. You can use the asterisk (*) wildcard in the value in this
field; for example, ABCD*.

ALL
This print option causes DB2 Path Checker to report on all matching and
changed plan table rows. If this parameter is omitted, only changed columns
are displayed.

JCL examples that use the PARM field
The following JCL examples illustrate typical DB2 Path Checker job steps that use
the PARM field to supply parameters.

Modify the JCL to work with your installation configuration.

JCL example 1
The following example produces a report that is based on preexisting Explain data
in userid.PLAN_TABLE. The collection ID is DSNESPCS, and the program name is
DSNESM68.

//STEP1 EXEC PGM=CKPPTHCK,
   // PARM='ssid,RPTPKG,userid,,DSNESPCS,DSNESM68'
//STEPLIB DD DSN=db2.SDSNL0AD,DISP=SHR
   // DSN=ckp.SCKPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*  

JCL example 2
The following example compares the same rows as the previous report to the same
plan table. This comparison must be equal.

//STEP3 EXEC PGM=CKPPTHCK,
   // PARM='ssid,COMPPKG,userid,PLAN_TABLE,DSNESPCS,DSNESM68,ALL'
//STEPLIB DD DSN=db2.SDSNL0AD,DISP=SHR
   // DSN=ckp.SCKPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*  

JCL example 3
The following example compares the same rows as the previous report to a
different plan table (PLAN_TABLE2). The ALL parameter causes access path rows
that match to be flagged with an asterisk (*).
JCL example 4

The following example explains each explainable statement in the DBRM and compares the results to the previous bind with plan name ORDERS. You must specify the programe variable when using the PARM field. The Explain results are saved in PLAN_TABLE with applname and collid set to blanks and programe set to CKPPTHCK.

JCL example 5

The following example explains each explainable statement in the DBRM and compares the results to the previous bind with plan name ORDERS. For this example, the input plan table is PLAN_TABLE2. You must specify the programe variable when using the PARM field. The Explain results will be saved in PLAN_TABLE with applname and collid set to blanks, and programe set to CKPPTHCK.

Related tasks:

"Running DB2 Path Checker by submitting batch jobs" on page 67
Run DB2 Path Checker by submitting a batch job that contains the required load libraries, DD statements, options, and commands.
Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe on IBM’s intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in a way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purpose of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These sample programs have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.
Index

A

access path data
  example of generating 10
  access paths 1
  anticipating changes to 6
  changes since previous bind 73
  comparing 75
  comparing current and previous version 76
  comparing for existing binds 110
  comparing, testing 13
  defined 2
  differences based on bind or rebind operations 73, 75
  differences based on changes to DB2 environment 70
  differences between binds of a program version 78
  differences between DBRMs and EXPLAIN results 80
  evaluating changed 3, 6, 10
  evaluating effects of changed access paths 75
  generating data into plan table 10
  generating for current DB2 environment 70
  generating for unbound DBRMs 70
  sources for processing 13
  testing 75, 88
  testing unbound DBRM against previous version 75

accessibility features 19

ALL parameter
  in PARM field 179

ANLOUT DD job control statement 174

applications
  defined 2
  deployment scenario 16
  performance 6

APPLNAME 63

architecture 13

automation
  batch jobs 67

B

basic reports 101

batch jobs
  creating automatically 67
  generating 96
  generating commands 90
  submitting 67, 96

bind control cards
  selecting programs to process 68

BINDIN DD job control statement 68, 174

BINDIN processing 68

binds
  anticipating changes for new versions 75
  comparing current and previous version 76
  comparing differences between editions and versions 73
  determining changed access paths between versions 78
  evaluating effects of 6, 14
  evaluating effects of changed access paths 73, 75, 76, 78, 80
  identifying EXPLAIN results for the current 145
  testing DBRMs against previous EXPLAIN results 80
  testing unbound DBRM against previous version 75

c

CAF
  See DB2 call attachment facility

collection reports 101

collection reports
  changing display options 39

CKP CLIST 81

CKP531I message output 101

codes
  completion 119
  error 119
  return 119

collections
  defined 2

COLLID 63

column headings
  interpreting in reports 103

command language
  examples 116

command syntax
  using wildcard characters in 145

commands 67

COMPARE 75, 78, 145
  TO PREVIOUS 78

COMPARE PACKAGE 16

CONNECT 70, 71, 73, 75, 76, 78, 80
  syntax and descriptions 148

CREATE SYNONYM 64

EXPLAIN 63, 70, 145
  syntax and descriptions 149

EXPLAIN DBRM 70
  syntax and descriptions 149

EXPLAIN PACKAGE 71
  syntax and descriptions 151

MAKE 14, 65
  syntax and descriptions 153

OPTIONS
  syntax and descriptions 154

OPTIONS CPUPCT 75, 76

OPTIONS CREATE TABLES 64

OPTIONS MATCHSQL2 75, 76

OPTIONS MATCHSQL2 command 73, 78

OPTIONS REPORTALL 67

OPTIONS REPORTCHG 70, 75, 76

OPTIONS REPORTCHG command 73, 78, 80

OPTIONS REPORTCOSTGT 75, 76

OPTIONS REPORTCOSTGT command 73, 78, 80

output data sets for 101

parameters 144

REPORT 145
  syntax and descriptions 165

SELECT
  syntax and descriptions 146

SET
  syntax and descriptions 166

SET CURRENT SQLID 70, 73, 75, 76, 78, 80
  specifying 144
  syntax 144
  syntax rules 144

TEST 14, 63, 67, 75, 114, 145
  syntax and descriptions 168

TEST DBRM 75, 76, 80

© Copyright IBM Corp. 2000, 2014; Copyright HLS Technologies, 2000, 2014

187
commands (continued)
FROM CATALOG 73
that produce reports 101
comments
methods for providing xiii
COMPARE command 10, 75, 76, 78
compare-to-previous method 3
normal comparison method 3
COMPARE PACKAGE command 16
completion codes 119
COMPPKG
value of request parameter 179
COMPPLAN
value of request parameter 179
CONNECT command 70, 71, 73, 75, 76
syntax and descriptions 148
Copy DB2 Entries panel 58
cost analysis 18
cost estimates
relying on 73
CPUPCT option 73, 75, 76, 80
CREATE SYNONYM command 64
creator-id parameter
in PARM field 179
CUSTOMER
value of request parameter 179
customization
associated list
adding DB2 entries 47
overview 139
associating DB2 entries 47
browsing parameters 58
changing display options 39
changing parameters 39
component 139
copying DB2 entries 58
Create a DB2 Entry panel 47
creating DB2 entries 47
customization jobs
deleting 61
displaying 60
generating 54
maintaining 61
regenerating 54
renaming 61
sort sequence 55
submitting 55, 60
customization library
deleting jobs 61
maintaining 61
overview 142
recustomizing 61
renaming jobs 61
customization library qualifier
specifying 37
Customized status 139
Customizer Workplace panel 54
customizing a new version of a product 39
customizing a product for the first time 39
customizing settings 37
data sets
customization library 142
data store 142
Discover EXEC library 142
metadata library 142
data store
overview 142
customization (continued)
data store data set
specifying 37
DB2 data sharing members
adding 47
associating 47
copying 58
creating 47
DB2 entries 139
adding 47
associating 47
copying 58
creating 47
DB2 parameters
defining 51
editing 51
DB2 Parameters panel 51
DB2 subsystems
adding 47
associating 47
copying 58
creating 47
defining DB2 parameters 51
defining parameters 49, 54
defining product parameters 49
deleting DB2 entries 60
deleting jobs 42
Discover Customized Product Information panel 45
Discover EXEC
customizing a new version of a product 39, 41
overview 142
retrieving product information automatically 45
Discovered status 139
discovering previous versions 41
discovering product information 45
display options 39
displaying jobs 60
displaying panel text 39
editing parameters 39
editing product parameters 49
Errors in Customization status 139
Finish Product Customization panel 55
first-time 40
first-time customization 39
generating jobs 54
hiding panel text 39
high-level qualifier 139
Incomplete status 139
job sort order 55
jobs
deleting 61
displaying 60
maintaining 61
customization (continued)
jobs (continued)
  renaming 61
  sort order 55
  submitting 55, 60
LPARs 61
maintaining jobs 61
master list
  adding DB2 entries 47
  Associate DB2 Entry for Product panel 47
overview 139
maximizing information panels 39
metadata libraries
  specifying 43
metadata library
  overview 142
  specifying 37
modifying parameters 39
modifying settings 37
multiple instances 37
multiple-LPAR environment 61
Not Required status 139
options 39
panel display options 39
panels
  Associate DB2 Entry for Product 47
  Create a DB2 Entry 47
  Customizer Workplace 54
  DB2 Parameters 51
  Discover Customized Product Information 45
  Finish Product Customization 55
  Product Parameters 49
  Specify the Metadata Library 43
parameters
  browsing 58
  defining 49, 54
  viewing 58
preparing to use Tools Customizer 37
product 139
product parameters
  changing 42
  defining 49
  editing 42, 49
  modifying 42
Product Parameters panel 49
Ready to Customize status 139
recustomization 39, 42
recustomizing 42
recustomizing a product 39
removing DB2 entries 59
roadmaps 39
  customizing for the first time 40
  first-time customization 40
  recustomizing 42
Specify the Metadata Library panel 43
specifying data sets 37
starting Tools Customizer 35
status types
  Customized 139
  Discovered 139
  Errors in Customization 139
  Incomplete 139
  Not Required 139
  Ready to Customize 139
submitting jobs 55
terminology 139

customization (continued)
user job card settings
  specifying 37
viewing parameters 58
customization library
  overview 142
customization library qualifier
  specifying 37
customizing settings 37

D

data
  requirements 10
data changes
  anticipating performance problems related to 70
data requirements 10
data store
  overview 142
data store data set
  specifying 37
database request modules
  See DBRMs
DB2 Bind Manager
  access paths
    evaluating changed 17
    using with DB2 Path Checker 17
DB2 call attachment facility 67
DB2 catalog 13
DB2 environment
  changed access paths 6
DB2 group attach field
  specifying 37
DB2 load library 67
DB2 Path Checker
  architecture 13
  capabilities 6
  features and benefits 6
  overview 1, 3
  process flow 13
  reference 139
  running as a batch job 67
  scenarios 14
terminology 2
DB2 Path Checker Interactive
  comparing packages 84
  comparing plans 86
  generating a Directory report 83
  generating batch jobs 90
  generating commands 90
  invoking 81
plan tables
  creating 97
  deleting rows 98
  maintaining 97
  resetting 98
setting default values
  Authorization ID 82
  DB2 subsystem 82
  submitting batch jobs 90
DB2 SQL Performance Analyzer
  interfacing 170
  JCL example 170
  using with DB2 Path Checker 18
DB2 SQL Performance Analyzer and DB2 Path Checker 18
DBRMIN 70, 78
DBRMIN DD job control statement 174
DBRMLIB 3, 13
DBRMOUT 73
DBRMOUT DD job control statement 174
DBRMs defined 2
DD statements
  ANLOUT 174
  BINDIN 68, 174
  DBRMIN 174
  DBRMOUT 174
  EXEC 174
  JOB 174
  PBINDOUT 174
  RBindOUT 174
  SYSCHG 174
  SYSEXPLN 174
  SYSIN 174
  SYSPRINT 174
  XMLSQLTX 174
default values
  interactive mode 82
diagnostic information gathering 136
Directory report generating 83
Discover EXEC overview 142
display attributes
  customizing 19
  display options 39
  displaying panel text 39
documentation accessing 19
documentation changes 1
downloads 18
DSN_STATEMENT_TABLE defining indexes on 63
DSNTIAUL 63
dynamic explain commands 63
examples (continued)
  RBINDOUT DD 70, 71, 73, 75, 76, 78, 80
  RPTPKG long report 105
  SET CURRENT SQLID command 70, 73, 75, 76, 78, 80
  SYSCHG DD 70, 71, 73, 75, 76, 78, 80
  SYSCHG report 108
  SYSEXPLN DD 70, 71, 73, 75, 76, 78, 80
  SYSEXPLN report 107
  SYSPRINT report 106, 108
  TEST command 114
  TEST DBRM command 73, 75, 76, 80
  using PARM field 180
  EXEC job control statement 174
  EXPLAIN command 63, 70
  syntax and descriptions 149
  EXPLAIN data
  generating for current DB2 environment 70
  generating for unbound DBRMs 70
  identifying EXPLAIN results for the current bind 145
  EXPLAIN DBRM command 3, 70
  syntax and descriptions 149
  EXPLAIN PACKAGE command 3, 10, 71
  syntax and descriptions 151
  explain processing 70
  explain reports 101
  EXPLAIN(YES) 10

F
FAQs 18
feedback
  methods for providing xiii
first-time customization 40
fix packs 18
flat files 13
FROM CATALOG 73, 78
functions overview 3

G
generating explain data 70
  for packages 71
  for state of environment at bind time 71

H
hardware changes
  anticipating performance problems related to 70
hiding panel text 39
hints
  creating by using the MAKE command 65
history table 109

I
index changes
  anticipating performance problems related to 70
indexes
  access path analysis 6
  interactive mode
  comparing packages 84
  comparing plans 86
  generating a Directory report 83
  invoking 81
interactive mode (continued)
  plan tables
    creating 97
    deleting rows 98
    maintaining 97
    resetting 98
  setting default values
    Authorization ID 82
    DB2 subsystem 82

ISPF interface
  comparing packages 84
  comparing plans 86
  generating a Directory report 83
  invoking 81
  plan tables
    creating 97
    deleting rows 98
    maintaining 97
    resetting 98
  setting default values
    Authorization ID 82
    DB2 subsystem 82

J
JCL
  example PARM field usage 180
  job control statements
    ANLOUT DD 174
    BINDIN DD 68, 174
    DBRMIN DD 174
    DBRMOVOUT DD 174
    EXEC 174
    JOB 174
    overview 174
    PBINDOUT DD 174
    RBINDOUT DD 174
    SYSCHG DD 174
    SYSEXPLN DD 174
    SYSIN DD 174
    SYSPRINT DD 174
    XMLSQLTX DD 174
  JOB job control statement 174
  JOINs
    access path analysis 6

K
keyboard shortcuts 19

L
legal notices
  notices 183
  trademarks 184
LOAD job 15
long names
  restrictions on 67

M
MAKE command 14
  creating optimization hints 65
  syntax and descriptions 153
MATCHSQL2 option 73, 75, 76, 78
maximizing information on panels 39
messages 119
  methods for accessing 135
metadata library
  overview 142
  specifying 37
migration 3
  evaluating effects of 15
  example 10
  from test environment 10
  in stages 15
  scenario 15
  test to production 10
  modifying settings 37
  MySupport 19

N
name substitution 68
NOMATCHCREATOR 16
notices 183

O
optimization hints
  creating with the MAKE command 65
  options 39
  OPTIONS command
    syntax and descriptions 154
    OPTIONS CPUPCT command 75, 76
    OPTIONS CPUPCT2 10
    OPTIONS CREATE TABLES command 64
    OPTIONS MATCHSQL2 command 75, 76
    OPTIONS REPORTALL 3
    OPTIONS REPORTALL command 67
    OPTIONS REPORTCHG command 70, 75, 76
    OPTIONS REPORTCOSTGT command 75, 76
    overview 1

P
packages
  comparing 84
  defined 2
  evaluating for rebinds 10
  example 10
panel display options 39
panels
  Copy DB2 Entries 58
  parallel processing
    anticipating performance problems related to 70
parameters
  commands 144
  specifying in the PARM field 179
  syntax 179
  PARM field 67
path change analysis 112
PBINDOUT DD 70, 71, 73, 75, 76, 78, 80
PBINDOUT DD job control statement 174
plan tables 13, 63
  creating 97
  defined 2
  defining indexes on 63
  deleting rows 98
  maintaining 97
  managing 97

Index 191