

# Path Check

Version 3.1

## User's Guide



HLS Technologies, Inc.

---

3322 Sturbridge Lane  
Sugar Land, Texas 77479-2223

Phone (281) 265-3004

Toll-free (888) 494-9019

Fax (281) 265-3006

[hlstech@attglobal.net](mailto:hlstech@attglobal.net)

<http://www.hlstechnologies.com/>

**Path Check User's Guide, Version 3 Release1**

**Third Edition (November 2007)**

This edition applies to Version 3 Release 1 of the Path Check Product, and to all subsequent releases until superseded by new editions or modified by technical documentation updates.

Reader comments on this document are welcomed and encouraged. Comments may be sent to:

HLS Technologies, Inc.  
Technical Publications Group  
3322 Sturbridge Lane  
Sugar Land, Texas 77479-2223

**© Copyright HLS Technologies, Inc. 2005. All rights reserved.**

Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise without the prior written of the publisher, HLS Technologies, Inc

# Table of Contents

---

<b>Preface</b> .....	<b>iv</b>
<b>Who Should Use This Book</b> .....	iv
<b>Summary of Changes</b> .....	v
<b>Product Specifications</b> .....	<b>1</b>
<b>Method of Operation</b> .....	1
<b>How path checker helps</b> .....	2
<b>Technical Description</b> .....	2
<b>Installing Path Check</b> .....	<b>4</b>
<b>Uploading and Unpacking the Product File</b> .....	4
<b>Running the Installation Jobs</b> .....	5
<b>Customizing Path Check Interactive</b> .....	6
<b>Using Path Check</b> .....	<b>8</b>
<b>The Path Check Job</b> .....	8
<b>Sources of Input</b> .....	8
<b>Job Control Statements (JCL)</b> .....	8
<b>Explanation of the Job Control Statements (JCL)</b> .....	9
EXEC Statement: .....	9
SYSPRINT DD Statement: .....	9
SYSEXPLN DD Statement: .....	10
SYSCHG DD Statement:.....	10
DBRMIN DD Statement:.....	10
DBRMOUT DD Statement:.....	10
ANLOUT DD Statement: .....	10
SYSIN DD Statement: .....	10
BINDIN DD Statement:.....	11
RBINDOUT Statement: .....	11
HISTWORK Statement:.....	11
<b>Path Check Commands</b> .....	11
Command Syntax .....	11
Wild Cards .....	12
Commands .....	12

<u>OPTIONS Command</u> .....	13
<u>SET Command</u> .....	14
<u>CONNECT Command</u> .....	15
<u>REPORT Command</u> .....	16
<u>COMPARE Command</u> .....	17
<u>TEST Command</u> .....	20
<u>MAKE Command</u> .....	22
<u>EXPLAIN Command</u> .....	22
<b>Path Check Data Requirements</b> .....	24
<b>Path Check Interactive</b> .....	27
<b>Using Path Check Interactive</b> .....	27
<b>Appendices</b> .....	32
<b>Appendix A — JCL Examples</b> .....	32
<b>Appendix B — REPORT Examples</b> .....	36
Example 1 – SYSPRINT report .....	36
Example 2 – SYSEXPLN Report .....	36
Example 3 – SYSCHG Report .....	37
Example 4 – SYSPRINT report with access path structure change. ....	38
Example 5 – History Table .....	38
<b>Appendix C — Messages</b> .....	42
<b>Appendix D — Parameter style input processing</b> .....	53
<b>Appendix E — CREATE TABLES option</b> .....	56
<b>Appendix F — Path Check Name substitution using BINDIN DDNAME Support</b> .....	58
<b>Notices</b> .....	59
<b>Trademarks</b> .....	59

# Preface

---

This book describes how to install and use Path Check. Path Check is a software product that allows developers to anticipate and avert unfavorable access path changes before they occur. It allows users to identify access path changes that may occur due to development, migration, or system activity and decide on an appropriate course of action before the changes take effect and impact the production DB2 environment.

## Who Should Use This Book

Users of Path Check, system programmers who must install and support the product, and data base administrators (DBA's) or application programmers who need to determine the applicability of the product for their installations gain the most use from this guide. It will also be useful for anyone who needs to understand the function of Path Check and the role it plays in application development and data base administration.

---

## Summary of Changes — Version 3 Release 1

Path Check V3R1 contains the following changes and enhancements:

- Added support for DB2 version 9.
- Added the ability to specify the target CCSID for unicode conversion of SQL
- Extended support for EXPLAIN and TEST processing to handle more SQL constructs that are not handled by SQL EXPLAIN.
- Added RBINDOUT ddname to easily generate REBIND commands for packages that do not identify access path changes.
- Added support for EXPLAIN FROM CATALOG to specify the source.
- Added OPTION CONVERTHINT to create a hint for access path changes identified by a TEST command.

## Summary of Changes — Version 2 Release 5

Path Check V2R5 contains the following changes and enhancements:

- Added //BINDIN to generate commands from bind control cards. Path Check will parse the Bind control cards and perform a path check command for each plan or package specified for BIND or REBIND.
- Added COMPARE TO PREVIOUS VERSION. This will skip multiple binds for the latest version and compare the access path to the prior version. This is useful when a program has been rebound multiple times due to an access path issue.
- Added history table for a summary of changes detected in Path Check processing. This table contains the same information as the SYSCHG ddname report.
- Added ANLOUT support for the COMPARE command if the full SQL is available in SYSIBM.SYSSTMT or SYSIBM.SYSPACKSTMT.
- Added DSN\_STATEMNT\_TABLE data to EXPLAIN TO processing so the cost estimates will be saved when the access data is saved.
- Added structure change summary to the SYSPRINT report when the table are accessed in a different sequence
- Added OPTIONS CREATE TABLES to create PLAN\_TABLE and DSN\_STATEMNT\_TABLE if the tables are not predefined.
- Added the ability to report on access paths taken by dynamic SQL. This requires the HLS dynamic SQL capture product.
- Changed the SYSPRINT report to clean up headings and put all DEGREE ANY access data in a separate line.

- Added Path Check Interactive which permits online comparisons through a TSO session

## Summary of Changes — Version 2 Release 4

Path Check V2R4 contains the following changes and enhancements:

- Added the estimated costs from the matching DSN\_STATEMNT\_TABLE to COMPARE and REPORT commands if the DSN\_STATEMNT\_TABLE is available.
- Added new report lines (PCK220I and PCK221I) to identify changed data columns in PLAN\_TABLE. These lines will only appear if the structure of the access path has not changed. The same tables must be accessed in the same sequence.
- A new option CATALOGSQL to display the new and old SQL text from SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT for COMPARE and REPORT commands.
- Added DD name SYSCHG to provide a summary of only changed access paths.
- Added new OPTION REPORTCOSTGT to identify an SQL statement that has the same access path but the cost estimates from DSN\_STATEMNT\_TABLE have changed.
- Added a new key word to the TEST command to allow a TEST command to be issued for the current SQL from the catalog without using a DBRM.
- Added sample SPUFI commands to generate EXPLAIN, COMPARE and TEST commands to the install library.

## Summary of Changes — Version 2 Release 3

Path Check V2R3 contains the following changes and enhancements:

- A new set of options MATCHCREATOR and NOMATCHCREATOR. NOMATCHCREATOR is useful for comparisons between different environments where the CREATOR and ACCESSCREATOR are changed.
- A new set of options SQLERROR and NOSQLERROR to control how explain failures are handled.
- A new option to save the results of an EXPLAIN command to a plan table as if a bind for a plan or package had occurred

- Added new data to the existing reports to highlight the version being processed

## Summary of Changes — Version 2 Release 2

Path Check V2R2 contains the following changes and enhancements:

- A new REPORT option (DIRECTORY)
- A new COMPARE option (PREVIOUS)
- An additional method of matching SQL statements when comparing two *Explains* (the previous methods are still available). OPTIONS MATCHSQL1 matches the SQL based on the type of SQL and tables referenced.
- A new option, BEFORE, has been added to the REPORT, COMPARE, and TEST commands. This will allow specific access data for a specific bind to be retrieved or used for comparison
- A new command, EXPLAIN, allows “what if” determination of DB2 access paths without performing an actual bind

## Summary of Changes — Version 2 Release 1

Path Check V2R1 contains the following changes and enhancements:

- A new command language
- Removal of the SHORT parameter
- A different method of matching SQL statements when comparing two *Explains* (the previous method is still available)
- Fully qualified plan table names and the ability to specify new or different plan/package names.

A new command language now supports all of the functions previously provided by the JCL PARM field as well as new (additional) functions. Commands are passed to Path Check via the SYSIN DD statement. The SHORT parameter is no longer supported. The decision to provide a one- or two-line report is based on the *Explain* data being depicted. (If there is data for the second line such as ACCESS\_DEGREE is greater than 1, the second line will be generated.) When comparing two *Explains*, new matching logic is used. If the query numbers have changed (whether because of the addition or deletion of non-SQL statements, or for any other reason), the *Explain* entries will be

compared based on physical sequence. This is the default and is called `MATCHSEQUENCE`. The old matching logic may be requested by specifying `OPTION MATCHQUERYNO`.

The new ability to specify new or temporary plans and collections for comparison makes it simple to compare a test plan or collection to the production version. This allows the user to verify access paths in a test environment and then ensure the same access paths are chosen in production. The user can also create a new collection with `BIND COPY` processing and then analyze the new access paths. The new control of the plan names and collection names reduces the need for a second plan table.

# Product Specifications

---

## Method of Operation

Path Check has 2 different modes of operation. It can compare and report on the access paths for existing PLAN\_TABLE and DSN\_STATEMNT\_TABLE entries from BIND processing or it can test access paths without binding the associated DBRM or package. The TEST command examines each SQL statement in a DBRM and performs an *explain* to determine the statement's access path. It then compares the new access path to the original (existing) access path from the backup PLAN\_TABLE (plan table). Then it generates a report displaying both the old and new access paths.

The TEST command and EXPLAIN command examine each explainable SQL statement in a DBRM and performs an SQL explain to determine the statement's access path. The TEST command will compare the new access data to a previous version and report on any changes. The EXPLAIN command simply produces a report of the new access path but it has an option to save the access and cost data under a specified name in any valid PLAN\_TABLE and DSN\_STATEMNT\_TABLE.

You may configure the report to display all statements' access paths, or just those that have changed. Because Path Check does not bind or rebind existing plans or packages, it does not impact the existing production environment. Therefore you have the freedom to conduct "what if" type analyses with complete transparency. You can also use Path Check to evaluate the results of a bind by comparing the newly generated access path to the "old" access path information in the backup plan table, and generating a report showing the results of the comparison.

Path Check works with any valid DB2 plan table format (25, 28, 30, 34, 43, 46, 49, 51 or 58 columns); if the old and new plan tables have different formats (different numbers of columns), Path Check will compare as many columns as possible based on the data available. You can also use Path Check with the DB2 V8 PLAN\_TABLE and can compare access data from a V8 system to the prior V7 PLAN\_TABLE. Path Check allows you to safely analyze the effects of changes to catalog statistics, changes from RUNSTATS, or migrations to different environments, *i.e.*, DB2 or hardware changes.

Path Check allows you to:

- Conduct "What if" type analyses with complete transparency
- Evaluate the results of a bind through comparing the new access path to the old
- Generate summarized or detailed reports describing the access path changes
- Compare access data from a V8 system to a prior V7 PLAN\_TABLE
- Analyze the effects of changes to catalog statistics, RUNSTATS, or migrations without consequences

## How path checker helps

By using Path Checker you can quickly determine whether a bind of a new DBRM or REBIND of an existing DBRM will result in a changed access path.

You typically run Path Checker-

- When a new release or version of DB2 is installed
- Service is applied
- When migrating a large application from one system to another.

With Path Checker, you can see the effects of doing a bind before the actual bind process or following a rebind. Based on the results, you might re-code the SQL or defer rebinding the application. The Path Checker TEST command will process all the SQL in a DBRM and identify any changes in the access path. If there are no changes in the access path for a DBRM that is bound as a package and the RBINDOUT ddname is specified, Path Checker will generate REBIND commands to rebind the package.

The Path Checker COMPARE command is used to compare the access paths for a DBRM after the BIND or REBIND has been processed.

## Technical Description

Path Check is available on PC diskette or via Internet download. It does not require APF authorization, nor does it use “hooks” or “back doors” into DB2 or MVS.

Path Check requires a valid DB2 authorization code in order to execute.

Path Check requires the following:

- Read access to the DBRMLIB containing the member to be checked.
- SELECT authority for the plan table (required for the REPORT and COMPARE functions).
- UPDATE authority for the plan table (required for the TEST, EXPLAIN and MAKE functions).
- SELECT authority for SYSIBM.SYSDBRM and SYSIBM.SYSPACKAGES for plan and package wild carding.
- SELECT authority for SYSIBM.SYSTABLES and SYSIBM.SYSINDEXES for statistics.
- SELECT authority for SYSIBM.SYSTMT and SYSIBM.SYSPACKSTMT for OPTIONS CATALOGSQL.

- SELECT authority for the matching DSN\_STATEMNT\_TABLE if DB2 costs are to be reported.
- UPDATE authority for the matching DSN\_STATEMNT\_TABLE (required for TEST and EXPLAIN commands).
- A primary or secondary authorization ID (authid) that has sufficient authority to issue a dynamic *explain* for the SQL in the DBRM (required for the TEST and EXPLAIN functions). This is generally the same authority required to execute the SQL (see the *DB2 Administration Guide* for more information). Path Check uses dynamic SQL for everything except the SET CURRENT SQLID command, so the DYNAMICRULE (BIND) option of BIND will allow public usage if it is bound by an authid with sufficient authority.
- Path Check uses the Call Attach interface, so the DB2 load library must be part of the STEPLIB or JOBLIB concatenation in the Path Check job step.

# Installing Path Check

---

## Uploading and Unpacking the Product File

Path Check arrives in a file named PCKXMIT. The file is provided in TSO Transmit format.

First, upload the product file to your MVS system. You can use any valid file transfer program: FTP, IND\$FILE, etc. Whichever mechanism you use to do the transfer, you *must* do the following:

- Preallocate a data set on the MVS system to receive the transferred file; do not allow the receiving data sets to be automatically allocated by the file transfer program. Allocate the data set with the attributes RECFM=FB, LRECL=80, and any valid blocksize. You may use any valid data set name.
- Transfer the file in *binary* mode.

Once the product file has been uploaded to your MVS host, you must unpack it, *i.e.*, convert it back to its original format (partitioned data set). Use the TSO *Receive* command to do this. The format of this command is

```
TSO RECEIVE INDA( 'data.set.name' )
```

In the command format shown above, you must replace *data.set.name* with the actual data set name you assigned to the uploaded file. For example, if the receiving data set was named USER55.PCKXMIT, then you would issue the command shown below.

```
TSO RECEIVE INDA( 'USER55.PCKXMIT' )
```

When you issue the *Receive* command, you will be prompted with message INMR906A. At this time you can just press ENTER, which will cause the reconstituted PDS library to be created using your TSO ID as the high level qualifier, or you can enter the subcommand

```
DA('new.data.set.name')
```

...which will cause the PDS to be generated using *new.data.set.name* as the name. Once you execute the receive command successfully, you should have a PDS library containing several members. One of the members is named README; it contains essentially the same information as the remainder of this chapter.

## Running the Installation Jobs

Once the two installation files have been uploaded and received, you are ready to run the installation job(s).

There are two installation jobs:

- One actually installs the product.
- The other is an Installation Verification Procedure that briefly tests the product to make sure the installation completed correctly. (The IVP job is also useful for as an example of how to run Path Check).

*Note:* You can find JCL for both jobs provided in File 1; the member names are INSTALL and IVP.

Before submitting either job, you will need to tailor the supplied JCL to conform to the specific environment into which you are installing Path Check. You will need to add a valid JOB statement as well as specify various other parameters to conform to local standards and requirements. Each job contains comments that give specific instructions on what to modify.

Once the INSTALL job has run successfully, Path Check is installed and ready to use. The IVP job will verify that Path Check is correctly installed.

The installation file also contains sample SPUFI members to generate Path Check commands.

*Note:* The SPUFI members will generate long commands that will need to be split into FB,80 for the path check SYSIN.

- GENEXP01 Select a list of plans bound by a specific id and generate explain command if there is no access path data in PLAN\_TABLE.
- GENEXP02 Select a list of packages bound by a specific id and generate explain command if there is no access path data in PLAN\_TABLE.
- GENEXP03 Select a list of packages bound by a specific id and generate explain command if there is no access path data in PLAN\_TABLE that matches the current version in SYSPACKAGE.

**NOTE** The access data will be the current access path not what was chosen at initial bind.

- GENCMP01 Select a list of packages bound by a specific id and generate compare command to match data from 2 different PLAN\_TABLE's.

- GENCMP02 Select a list of packages bound by a specific id and generate compare command to match in the same PLAN\_TABLE but with different collection id's.
- GENCMP03 Select a list of packages bound by a specific id and generate compare command to match in the PLAN\_TABLE to the previous bind in the same PLAN\_TABLE.
- GENCMP04 Select a list of packages bound by a specific id and generate compare command to match in the PLAN\_TABLE to the previous bind in the same PLAN\_TABLE that is before a given timestamp value

<b>NOTE</b>	the <b>TIMESTAMP</b> column in the <b>PLAN_TABLE</b> is not datatype <b>TIMESTAMP</b> and the format matches the format of <b>PLAN_TABLE</b> not 'yyyy/mm/dd.hh.mm.ss.mmmmmm'
-------------	---

- GENTST01 Select a list of plans bound by a specific id and generate TEST command to compare access data for all the DBRM's in the plan as if it was rebound.
- GENTST02 Select a list of packages bound by a specific id and generate TEST command to compare access data as if the package was rebound.

## Customizing Path Check Interactive

Path Check Interactive is a feature of Path Check that permits online comparison of access paths from a TSO session. Path Check Interactive is an ISPF dialog that functions as a "front end" to the standard Path Check program. (Path Check is normally run as a batch job.)

Using standard ISPF features and protocols, Path Check Interactive presents the user with user-friendly ISPF screens where input parameters may be specified and standard Path Check functions requested. In this release, the COMPARE function is supported. Future releases will provide support for additional Path Check commands and functions.

Path Check Interactive is automatically installed by the INSTALL job shipped with Path Check. However, after the INSTALL job runs, some additional customization is required in order to use Path Check Interactive.

The INSTALL job creates and initializes several libraries. Among these is a CLIST library that contains the main Path Check CLIST (member name PATHCHK). This CLIST must be modified before use. The modification consists of specifying the data sets names for the libraries created by the INSTALL job. See the comments in the CLIST for details.

Once the CLIST has been customized, Path Check Interactive may be invoked in either of two ways:

1. By directly executing the “PATHCHK” CLIST. This can be done by issuing the command `TSO EXEC 'datasetname(PATHCHK)'` where *datasetname* is the name of the library containing the PATHCHK CLIST.
2. By selecting Path Check Interactive from a menu that has been suitably modified by your Path Check Installation specialist or system programmer.

Which method is used at your site depends on choices made by your Path Check installation specialist, or your installation’s system programmers.

# Using Path Check

---

## The Path Check Job

Path Check operates entirely in batch mode; to use it, you submit a batch job. See the sample JCL shown in Appendix A.

## Sources of Input

You control Path Check by using input parameters you specify in the JCL PARM field, or by commands supplied in SYSIN. HLS strongly recommends using SYSIN commands; support for the PARM field options is provided for compatibility with previous releases.

### SQL

If SYSIN commands are present, the PARM field parameters are ignored.

## Job Control Statements (JCL)

Figure 1 shows the job control statements for Path Check.

Statement	Use
JOB	Starts the job.
EXEC	Starts Path Check.
SYSPRINT DD	Defines a report data set used for listing control statements, messages, and output reports.
SYSEXPLN DD	Optional. Defines a report data set used for listing "Explain" output in a user-friendly format.
DBRMIN DD	Required for the TESTPKG or TESTPLAN function; defines a partitioned data set and member containing the DBRM to be tested.
ANLOUT DD	Optional. Used to route SQL from a DBRM specified in a TEST or COMPARE command to the SQL Performance Analyzer product if the access path had changed. The dataset must have DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)specified.
SYSCHG	Optional. Identification of changed SQL's only

DBRMOUT	Temporary dataset required for the FROM CATALOG option of the TEST or EXPLAIN command. The dataset must have DCB=(LRECL=80,BLKSIZE=0,RECFM=FB) specified.
SYSIN DD	Defines the optional control data set.
BINDIN DD	Causes path check to ignore the plan and package name in the commands and identify the plan or package to be processed from the BIND or REBIND control commands. The dataset must be DCB=(LRECL=80,RECFM=FB).
RBINDOUT DD	Identifies an output dataset that will contain rebind commands for every TEST DBRM as PACKAGE command that completes successfully and does not identify any changes of an access path.
HISTWORK	Required dataset for the OPTION HISTORY TABLE processing. Requires DCB=(RECFM=VB,LRECL=854,BLKSIZE=0) and must be specified for OPTIONS HISTORY TABLE processing.

Figure 1 - Job Control Statements for Path Check

## Explanation of the Job Control Statements (JCL)

### EXEC Statement:

The format of the EXEC statement is:

```
//stepname EXEC PGM=PATHCHK
```

where:

**PGM=PATHCHK**

specifies that you want to run the Path Check program.

**PARM='parameters'**

The JCL parm field is optional and is described in appendix E.

### SYSPRINT DD Statement:

Path Check writes a log of the control statements and its actions to the SYSPRINT DD statement. Output reports are also written to SYSPRINT (with one exception – see *SYSEXPLN DD Statement*, below).

You may assign SYSPRINT to sysout or to any sequential data set with LRECL=133.

**SYSEXPLN DD Statement:**

This DD statement is optional. If present, it is used to write a detailed *Explain* report for each SQL statement processed. This report is in an expanded format that is easier to read than the terse one- or two-line reports written to SYSPRINT.

You may assign SYSEXPLN to sysout or to any sequential data set with LRECL=133.

**SYSCHG DD Statement:**

This DD statement is optional. If present, it is used to write a summary list of the SQL statements with changed access paths.

You may assign SYSCHG to sysout or to any sequential data set.

**DBRMIN DD Statement:**

This DD statement is used for the TEST and EXPLAIN commands if the command does not specify FROM CATALOG. The DD statement is ignored for all other functions. It identifies the partitioned data set or concatenation that contains the DBRM to be analyzed.

**DBRMOUT DD Statement:**

This DD statement is used for the TEST command with the FROM CATALOG option. The DD statement is ignored for all other functions. It identifies the temporary dataset that is used to rebuild the DBRM from the catalog. It must have DCB=(RECFM=FB,LRECL=80,BLKSIZE=0) specified for DCB characteristics.

**ANLOUT DD Statement:**

This DD statement is optional. If present, and if a TEST or COMPARE command is processed for a changed access path, the generated output will be written to this DDNAME. This allows the output to be directed to another software product such as the SQL Performance Analyzer by IMSI, Inc. ANLOUT DD statement will only work with the COMPARE command if the full SQL is available in SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT.

If the ANLOUT ddname 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 SQLPA processing.

**SYSIN DD Statement:**

This DD statement is now required. It must be followed by one or more valid Path Check commands. (See *Path Check Commands*, below, for details of command syntax and usage).

### **BINDIN DD Statement:**

This DD statement is optional. It will cause Path Check to generate the list of plans or packages to be processed from bind control cards. Path Check will parse the Bind control cards and perform the path check command from SYSIN for each plan or package specified in the BIND control cards. Path Check will ignore the plan and package name in the command and identify the plan or package to be processed from the BIND control commands. Only one command can be processed with the BINDIN controlling the plan, collection and program name.

This is useful where path check is being used in conjunction with an existing process that creates and processes BIND and REBIND commands. (See *Path Check Name substitution using BINDIN DDNAME Support* in Appendix F for an example).

### **REBINDOUT Statement:**

This DD statement identifies an output dataset that will contain rebind commands for every TEST DBRM as PACKAGE command that completes successfully and does not identify any changes of an access path. The dataset must be DCB=(RECFM=FB,LRECL=80,BLKSIZE=x) where the blocksize is a valid multiple of 80 or 0 to use system determined block size.

The format of the rebind command will be:

```
REBIND PACKAGE (collid.progname.(version))-  
  OWNER (owner)-  
  EXPLAIN(YES)
```

### **HISTWORK Statement:**

Required dataset for the OPTION HISTORY TABLE processing. Requires DCB=(RECFM=VB,LRECL=854,BLKSIZE=0) and must be specified for OPTIONS HISTORY TABLE processing.

## **Path Check Commands**

Path Check command statements are read from ddname SYSIN. This section contains information about the command types and their function within Path Check.

### *Command Syntax*

Path Check commands must obey the following syntax rules:

- Any line whose first two nonblank characters are dashes (- -) is considered to be a comment and is ignored.
- Comment lines may be interspersed with command lines in any order (a comment line may appear between two portions of a command).

- Whole-line comments only are permitted; if a line begins with the comment characters (- -), the entire line is treated as a comment. A comment may not appear on the same line as a command or a portion of a command.
- A single command may span multiple lines. However, each command must begin on a new line; “stacking” of multiple commands on a line is not allowed.
- The keywords and values that make up a command must be separated by one or more blanks.

## Wild Cards

The REPORT, COMPARE, EXPLAIN and TEST commands permit the use of wild card characters to specify a range of one or more objects to be processed. The following wild card characters are supported:

- % represents any single character
- \* represents any sequence of zero or more characters

See the individual command descriptions for details of where and when wild cards may be used. The COMPARE and REPORT commands select the program names from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE for the wild card matching. The TEST and EXPLAIN commands select program names from the MEMBER list for DD name DBRMIN unless the commands use the option FROM CATALOG. The FROM CATALOG option causes the TEST and EXPLAIN commands to select the program names from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE using the same logic as COMPARE and REPORT.

## Commands

The following commands are available:

OPTIONS	Set processing options
SET	Set current SQL ID or catalog qualifier
CONNECT	Connect to a DB2 subsystem
REPORT	Generate an access path report for one or more programs
COMPARE	Compare the access paths for one or more programs to previously <i>Explained</i> access paths

*Warning:* You MUST have something to compare to in the command syntax.

TEST	<i>Explain</i> the access paths for a DBRM and compare the access paths to a previously <i>Explained</i> access path
MAKE	Mark an existing access path in a plan table as a hint

**EXPLAIN** List the access path that would be chosen for a DBRM in the event of a bind

## OPTIONS Command

The format of the OPTIONS command is:

```
OPTIONS option 1, option 2, . . . option n
```

The following *options* may be specified:

<b>REPORTCHG</b>	Display only changed access paths (this is the default)
<b>REPORTALL</b>	Display all access paths, regardless of whether they have changed or not.
<b>MATCHSEQUENCE</b>	Match SQL statements sequentially, <i>e.g.</i> , 1 <sup>st</sup> to 1 <sup>st</sup> , 2 <sup>nd</sup> to 2 <sup>nd</sup> , etc.
<b>MATCHQUERYNO</b>	Match up SQL statements using the QUERYNO value.
<b>MATCHSQL1</b>	Match SQL statements based on the SQL characteristics rather than by statement number or physical sequence in the program. This is the default matching technique.
<b>MATCHCREATOR</b>	This will include the creator and accesscreator in the access path comparison. This is the default.
<b>NOMATCHCREATOR</b>	This will exclude the creator and accesscreator in the access path comparison.
<b>NOSQLERROR</b>	This will suppress error messages for SQL statements in a DBRM that can't be explained such as selects from SYSDUMMY1. This is the default.
<b>SQLERROR</b>	This will produce error messages for SQL statements in a DBRM that do not explain correctly
<b>NOCATALOGSQL</b>	SQL text will only be included for the TEST command from the DBRM. This is the default.
<b>CATALOGSQL</b>	The COMPARE and REPORT commands will include the SQL statement text in the SYSEXPLN report when it is available.
<b>NOREPORTCOSTGT</b>	Only access path changes are included in the change report. This is the default
<b>REPORTCOSTGT</b>	A SQL statement will be included in the reports if either the access path changes or the estimated costs in the column PROCSU in DSN_STATEMNT_TABLE have increased.
<b>CREATE TABLES</b>	This will cause Path Check to create the required PLAN_TABLE and DSN_STATEMNT_TABLE to process the TEST and EXPLAIN commands. These tables will be dropped at the end of the process. They are created

as implicit tables where the create statement does not specify a tablespace.

**CCSID = nnnn** Where nnnn is a valid ccsid. The default is CCSID=37 but it should be specified to the local EBCDIC DB2 ccsid. This is useful for sites have SQL containing national characters. The default value is 37 which is the base ebcdic ccsid. The ccsid specified must be supported by the CUNLCNV z/os unicode conversion services.

**CONVERTHINT name** This will cause path check to create an access path HINT with the old access path data and the specified hint name, when a TEST command identifies a change in access path. The hint may not be a valid access path if table or index changes have been made.

The following options are mutually exclusive. If more than one in any set are specified on the same command, the last value specified is the one used.

- **REPORTCHG and REPORTALL**
- **MATCHSEQUENCE, MATCHQUERYNO, and MATCHSQL1**
- **MATCHCREATOR and NOMATCHCREATOR**
- **SQLERROR and NOSQLERROR**
- **CATALOGSQL and NOCATALOGSQL**
- **REPORTCOSTGT and NOREPORTCOSTGT**

## SET Command

The SET command has two forms. Format 1 is:

```
SET CURRENT SQLID = 'sqlid'
```

Where SQL ID is a valid authid. Note that normal DB2 rules for SQL ID apply; the user submitting the job must have SYSADM authority or valid RACF (or equivalent) access to the group. Also note that spaces are required before and after the equal sign (=).

The specified SQLID will be the default qualifier for the application SQL processed for the TEST or EXPLAIN command. DB2 BIND processing has 2 parameters to process access path information. The OWNER parameter identifies the authorization id that has the required SQL authorization for the BIND and is also the qualifier for the PLAN\_TABLE and DSN\_STATEMNT\_TABLE used to hold the access data. The QUALIFIER parameter is used to identify owner id of the tables that will be processed by the SQL. The TEST and EXPLAIN commands use SQL EXPLAIN which requires the current SQLID be both the owner of a PLAN\_TABLE and DSN\_STATEMNT\_TABLE and the qualifier for the application tables. If the OWNER parameter and QUALIFIER

parameter are different in the normal BIND processing. Path Check will need additional PLAN\_TABLE and DSN\_STATEMNT\_TABLE's defined under the application qualifier. If the PLAN\_TABLE and DSN\_STATEMNT\_TABLE do not exist, the TEST and EXPLAIN commands will fail with PCK241E message for the Path Check commands. The other possible solution is to specify OPTIONS CREATE PLANTABLE which will create the required tables and then drop them at the end of the run. DB2 version 7.1 and prior does not allow the PLAN\_TABLE or DSN\_STATEMNT\_TABLE to be either an alias or synonym.. DB2 version 8.1 will accept an ALIAS but not synonyms.

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 SQL ID from the SET command.)

Blanks are required before and after the equal sign (=).

This command is required before a TEST command can be processed.  
Format 2 of the SET command is as follows:

```
SET CATALOG QUALIFIER = 'SYSIBM'
```

This is required if a shadow catalog is being used. The user submitting the job must have SELECT access to the catalog. The shadow catalog must contain the rows defining the shadow catalog tables and support the following tables along with the plan\_tables and DSN\_STATEMNT\_TABLES used for processing.

```
SYSTABLES  
SYSINDEXES  
SYSDBRM  
SYSPACKAGE  
SYSSTMT  
SYSPACKSTMT  
SYSPACKLIST
```

## CONNECT Command

The format of the CONNECT command is:

```
CONNECT TO ssid
```

Where *ssid* is the name of the DB2 subsystem where processing is to occur. A single job may connect to multiple subsystems, one at a time. CONNECT must occur before the first REPORT, COMPARE, TEST, EXPLAIN, or MAKE processing command in the control file (SYSIN).

## REPORT Command

The REPORT command generates an access path report. There are 3 types of reports available:

- DDNAME SYSPRINT – the default summary report
- DDNAME SYSEXPLN – an optional detailed report that is not produced if the DDNAME is not present.
- DIRECTORY option of the REPORT command will produce a summary of the access data from a bind and EXPLAIN activity that is available in the PLAN\_TABLE in the SYSPRINT output.

The REPORT format is used for COMPARE and TEST processing. The PLAN\_TABLE used for the report can be any valid DB2 PLAN\_TABLE.

The format of the REPORT command is:

```
REPORT
  ON object-type qualifier.program
  IN tablename
  START WITH programe
  BEFORE timestamp
  DIRECTORY
```

where:

### **object-type**

is either PLAN or PACKAGE.

### **qualifier**

is the name of the plan or collection to report on.

### **program**

is the name of the program to report on. This name may contain wild card characters.

### **tablename**

is 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 omitted, Path Check will use the plan table from the last previous request. If this parameter is omitted, also omit the keyword IN.

The plan table name, if specified, may or may not have the creator ID (may or may not be fully qualified). If the creator ID is missing, Path check will use the creator ID

from the last previous request, or, if this is the first request, the creator ID from the last valid SET CURRENT SQLID command.

### **progname**

START WITH is an optional clause that allows you to start a report at a specific point. If *progname* does not exist, the report will start with the next higher name that does exist. 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 prior to the date and time specified by *timestamp* will be included in the report. This is especially useful when multiple binds with EXPLAIN have been done and the plan table thus contains multiple entries for a particular plan or package.

Example: BEFORE '2002121222421165' would run the report only for plan table rows created prior to the specified time. The timestamp value to specify is best determined from the output of a previous REPORT DIRECTORY run. The ending timestamp value from one of the lines in a previous DIRECTORY report is usually a good value to use. The TIMESTAMP column in a DB2 PLAN\_TABLE is not a date or timestamp value so the format of the timestamp value matches the character (16) format used by DB2. The timestamp value does not have to match an ending timestamp exactly - it can be a value for the beginning of a migration process.

## **DIRECTORY**

This optional keyword generates a summary report showing the timestamps for each bind of the selected plan(s) or package(s) — provided, of course, 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.

## **COMPARE Command**

*Warning:* You MUST have something to compare to in the command syntax.

The COMPARE command will compare the access path data for 2 different sets of access data from the same or different PLAN\_TABLE's. The COMPARE command will accept wild cards for the program name and select the members to be processed from SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE (or the equivalent shadow table).

The PLAN name or COLLECTION name can be the same or different but the program names must match exactly. If the plan tables are different names, they do not have to be the same format but the format must be a valid version supported by DB2.

The format of the COMPARE command is:

```
COMPARE
  object-type1 qualifier1.program
  IN tablename1
  TO qualifier2.*
  IN tablename2
  START WITH programe
  BEFORE timestamp
```

```
COMPARE
  object-type1 qualifier1.program
  IN tablename1
  TO PREVIOUS
```

```
COMPARE
  object-type1 qualifier1.program
  IN tablename1
  TO PREVIOUS VERSION
```

**object-type1**

is either PLAN or PACKAGE.

**qualifier1**

is the name of the plan or collection to report on; this is considered the “new” plan or collection.

**program**

is the name of the program to report on. This name may contain wild card characters.

**tablename1**

is the name of the plan table containing the *Explain* information for *qualifier1.program*. This is considered the “new” plan table.

**qualifier2**

is the name of the plan or collection to compare against; this is considered the “old” plan or collection.

- *qualifier2* is always specified with an asterisk (\*) as the program name because the same program name (*program* above) is used for both sides of the comparison.
- *qualifier2* must be the same as *qualifier1*. Plans must be compared to plans and packages must be compared to packages; “apples-to-oranges” comparisons are not permitted.

**tablename2**

is the name of the plan table containing the *Explain* information for *qualifier2.program*. This is considered the “old” plan table.

**progrname**

START WITH is an optional clause that allows you to start a previous run at a specific point. If the program named *progrname* exists, processing will begin with that program. If *progrname* 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 prior to the date and time specified by *timestamp* will be included in the report. This is especially useful when multiple binds with EXPLAIN have been done and the plan table thus contains multiple entries for a particular plan or package.

Example: BEFORE ‘2002121222421165’ would run the report only for plan table rows created prior to the specified time. The timestamp value to specify is best determined from the output of a previous REPORT DIRECTORY run. The ending timestamp value from one of the lines in a previous DIRECTORY report is usually a good value to use.

The TIMESTAMP column in a DB2 PLAN\_TABLE is not a date or timestamp value so the format of the timestamp value matches the character (16) format used by DB2. The timestamp value does not have to match an ending timestamp exactly - it can be a value for the beginning of a migration process. The BEFORE option is mutually exclusive with PREVIOUS or COMPARE PLAN or PACKAGE name TO PREVIOUS VERSION – only one of the keywords can be specified for a request.

**PREVIOUS**

Allows you to choose to compare to a previous version or a specific version.

If PREVIOUS is coded, then for each selected plan or package, the current access path will be compared to the next oldest (-1) edition of that plan or package version in the same PLANTABLE.

COMPARE PLAN or PACKAGE name TO PREVIOUS VERSION. This will skip multiple binds for the most current version and compare the access path to the prior version. This is useful when a program has had multiple BIND or REBIND commands and you need to compare to the prior software version of a program. If the plan\_table does not contain access path data for multiple versions, the report will identify all SQL statements as new SQL. If the program was not compiled with versioning (such as VERSION(AUTO) or a specific version), the report will identify all the SQL statements as new SQL.

The PREVIOUS option is mutually exclusive with BEFORE, and COMPARE PLAN or PACKAGE name TO PREVIOUS VERSION. – Only one of the keywords can be specified for a request.

## TEST Command

The TEST command will evaluate the access path data for a DBRM before the BIND has occurred. This allows the indexes and statistics to be evaluated before the production BIND has occurred. The TEST command requires DDNAME DBRMIN which must be a valid DBRMLIB or concatenation of DBRMLIB's. The TEST command only processes SQL that can be explained. SQL such as OPEN CURSOR, CLOSE CURSOR, UPDATE CURRENT OF CURSOR are ignored. The results of the TEST command normally only identify access path changes unless OPTION REPORTALL has been specified. SQL that does not have a changed access path will be displayed with a leading character of '\*' with OPTION REPORTALL. If ddname RBINDOUT is present and the TEST command processes a package successfully with no access path changes, then Path Checker will generate a REBIND command for that package.

The format of the TEST command is:

```
TEST
  DBRM dbrmname
  AS object-type qualifier.*
  IN tablename
  START WITH progrname
  BEFORE timestamp
  FROM CATALOG
```

### **dbrmname**

is the name of the program to test. This name may contain wild card characters.

### **object-type**

is either PLAN or PACKAGE.

### **qualifier**

is the name of the plan or collection to test. **Note:** *qualifier* is always specified with an asterisk (\*) as the program name because the same program name (*dbrmname* above) is always to generate fresh *Explain* data and do the comparison.

### **tablename**

is the name of the plan table to use.

**progrname**

START WITH is an optional clause that allows you to start a previous run at a specific point. If the program (DBRM) named *progrname* exists, processing will begin with that program. If *progrname* 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 prior to the date and time specified by *timestamp* will be included in the report. This is especially useful when multiple binds with EXPLAIN have been done and the plan table thus contains multiple entries for a particular plan or package.

Example: BEFORE '2002121222421165' would run the report only for plan table rows created prior to the specified time. The timestamp value to specify is best determined from the output of a previous REPORT DIRECTORY run. The ending timestamp value from one of the lines in a previous DIRECTORY report is usually a good value to use.

The TIMESTAMP column in a DB2 PLAN\_TABLE is not a date or timestamp value so the format of the timestamp value matches the character (16) format used by DB2. The timestamp value does not have to match an ending timestamp exactly - it can be an value for the beginning of a migration process.

**FROM CATALOG**

This is an optional clause that will cause Path Check to retrieve the SQL from the most current version in the catalog. When FROM CATALOG is specified, DD DBRMIN is not required and is ignored. This option requires a work DD name DBRMOUT with DCB=(LRECL=80,BLKSIZE=0,RECFM=FB) for processing.

**//WILDCARD**

This will override the normal selection criteria for the TEST command -  
The test command will select based on syspackage or sysdbrm data where the memberlist will select based on the member list from the input DBRMIN ddname

```
sample
//WILDCARD DD *
  MEMBERLIST TEST*
```

**//RBINDOUT**

This is a new ddname that will generate rebind commands after a TEST command that does not identify any changes of an access path.

*Note:* before using the TEST command, you must have issued a successful SET CURRENT SQLID command.

## MAKE Command

The MAKE command will identify an existing access path in a plan table as a HINT. This requires a BIND to make the hint active. The BIND will only accept a hint if the correct DSNZPARM has been set to activate hint's. (See the DB2 installation guide for OPTHINTS).

The format of the MAKE command is:

```
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.

### **timestamp**

is the timestamp value that, combined with the QUERYNO, uniquely identifies the row.

### **object-type**

is either PLAN or PACKAGE.

### **qualifier**

is the name of the plan or collection containing *program*.

### **program**

is the name of the program containing statement *nnn*.

### **tablename**

is the name of the plan table containing the row that is to be made into a hint. This name must be of the form *creatorid.tablename*.

### **hintname**

is the hint name to be assigned to this row.

## EXPLAIN Command

The EXPLAIN command will evaluate the access path's selected for a new DBRM without comparing the resulting access data to any history. It is the equivalent of a

REPORT command for a new DBRM. The results of an EXPLAIN command can be saved to compare with the actual access data from the BIND or to rebuild history in the plan table if the access data has been lost.

The format of the EXPLAIN command is:

```
EXPLAIN
  DBRM dbrmname
  TO object-type qualifier
  IN tablename
  FROM CATALOG object-type qualifier
```

### **dbrmname**

is the name of the program to test. Wildcard characters are allowed in the name; if wildcards are used, then the program(s) to be tested will be selected from those available in the DBRMIN concatenation using standard TSO wild carding rules.

TO is an optional clause that will cause Path Check to save the access data to the PLAN\_TABLE specified and the matching DSN\_STATEMNT\_TABLE cost information is saved also if the program had been bound with EXPLAIN(YES).

FROM CATALOG is an optional clause that you can use to choose the program name to explain. The program name can be wild carded but the collection identifier or plan identifier will be selected from the DB2 catalog table (either SYSIBM.SYSDBRM or SYSIBM.SYSPACKAGE depending on the command referencing a plan or package). The normal EXPLAIN command will select members from a directory list of DBRMIN – DBRMIN is not used for CATALOG processing.

### **object-type**

is either PLAN or PACKAGE.

### **qualifier**

is the name of the plan or collection to test. This can be any plan name or collection name. The program name will always be the dbrmname from the precompile.

### **tablename**

is the name of the plan table to save the access data into. This does not have to be the same plan table that is used for explain package. The normal EXPLAIN command will select members from a directory list of DBRMIN – DBRMIN is not used for FROM CATALOG processing. The creator id for the plan\_table specified is the same creator used for the matching DSN\_STATEMNT\_TABLE.

## **FROM CATALOG**

will select the SQL from the DB2 catalog and perform the EXPLAIN processing. DBRMIN is not used for FROM CATALOG processing. The work DDNAME DBRMOUT is required for FROM CATALOG processing. Required clause PLAN plname or PACKAGE collid identifies the source for the DBRM to be explained (either SYSSTMT or SYSPACKSTMT). The FROM CATALOG option requires an object qualifier to process.

## **Path Check Data Requirements**

Path Check depends on the information stored in PLAN\_TABLE and DSN\_STATEMENT\_TABLE from BIND EXPLAIN(YES) and SQL EXPLAIN statements. Every DB2 release has added data to the PLAN\_TABLE definition and V8 changed the data types and lengths of many of the columns. Path Check will compare access path information for any valid DB2 PLAN\_TABLE. It can also compare unlike PLAN\_TABLE's where one table is for a prior release of DB2.

Path Check expects the DSN\_STATEMENT\_TABLE to have the same creator id as the PLAN\_TABLE being processed. When a REPORT or TEST command specifies a target userid. PLAN\_TABLE, the estimated cost information will be retrieved from userid.DSN\_STATEMENT\_TABLE. When a COMPARE command references 2 different PLAN\_TABLES's, Path Check will process the cost information from the matching DSN\_STATEMENT\_TABLES;s. When a COMPARE command references the same PLAN\_TABLE but with different time stamp values of collection id's, the data values will match the correct information in DSN\_STATEMENT\_TABLE. The DSN\_STATEMENT\_TABLE is not required for normal Path Check operation. If DSN\_STATEMENT\_TABLE is not available, OPTIONS REPORTCOSTGT will not function for COMPARE and TEST commands.

### **Scenario 1:**

There are multiple programs that do not have access data in PLAN\_TABLE.

#### **Solution 1 –**

REBIND EXPLAIN(YES). This will generate access path data into PLAN\_TABLE but it will also generate new access paths. Run the REPORT command of Path Check to view the new access paths. Check the run times for the new access paths to see if they increase or decrease.

#### **Solution 2 –**

Update the member GENEXP01 for plans that don't have data and run the generated Path Check EXPLAIN commands to find the probable access paths.

Update member GENEXP02 (no versioning for DBRM's) or GENEXP03 (checks for version match in PLAN\_TABLE) for packages that don't have access data and run the generated Path Check EXPLAIN commands to find the probable access paths. Review the access paths and verify that they seem reasonable. If there are no obvious problems, do the REBIND EXPLAIN(YES) and then run COMPARE commands to verify that BIND choose the same access path.

This has several advantages:

1. The Path Check EXPLAIN commands do not update the access path. If there are problems with the catalog statistics or the access paths chosen, run RUNSTATS and rerun the EXPLAIN commands.
2. The EXPLAIN data establishes a base line for the expected access paths. This does require that you have access to the correct DBRM. HLS Technologies has other products that will assist in identifying mismatches between load modules, DBRM's and the existing data in the DB2 Catalog.

### **Scenario 2:**

There are a set of programs that are moving from a test environment to production. The test database is in the same DB2 subsystem as the production but the qualifier id's are different.

#### **Solution 1 to the first question –**

What access path changes have changed with the new programs?

*Note:* Normally this question should be asked before the migration process.

Compare the access paths in test to the existing access paths in production by using OPTIONS NOMATCHCREATOR and COMPARE PACKAGE testcollid.progname IN test.PLAN\_TABLE TO prodcollid.\* in prod.PLAN\_TABLE. This will identify new, changed and deleted SQL statements and access path changes where the same table is referenced with the same or different SQL.

#### **Solution 2 to the first question –**

What access path changes have changed with the new programs?

Specify DBRMIN ddname to the new DBRMLIB. SET CURRENT SQLID = 'prod' and issue TEST DBRM progname as prodcollid.\* in prod.PLAN\_TABLE.

#### **Solution for new programs that don't have any access data in production.**

SET CURRENT SQLID = 'prod' and EXPLAIN DBRM progname.

**Solution to the second question –**

Did the programs get the same access paths in production that were verified in test?

After the Binds for the programs in production complete, compare the access paths in production to the access paths that were tested by using `OPTIONS NOMATCHCREATOR` and `COMPARE PACKAGE collid.progname IN prod.PLAN_TABLE TO testcollid.*` in `test.PLAN_TABLE`.

# Path Check Interactive

## Using Path Check Interactive

The following narrative illustrates a typical Path Check Interactive session.

When you invoke Path Check Interactive, the first screen you see will be the Path Check Interactive main menu (Figure 2).

```
Menu Utilities Options Help
-----
Path Check Interactive 2.5.0
COMMAND ==>>
Select a function:
    0 SETTINGS Global options
    1 REPORT Display PLAN_TABLE information
    2 COMPPPL Compare access paths (plan DBRM)
    3 COMPPK Compare access paths (package)

    U UTILITY Utility functions
    X EXIT End dialog
```

Figure 2

Before you can use the Path Check Interactive functions, you must first specify some basic defaults. Selecting option 0 on the main menu causes the Settings panel to be displayed (Figure 3); this is where the default values are entered.

```
Menu Utilities Options Help
-----
Path Check Interactive - SETTINGS
COMMAND ==>> Scroll ==>> CSR

*DB2 subsystem . . . . . DB7G
*Default AUTHID for
PLAN_TABLES . . . . . DNS1 >

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

Figure 3

There are two values you must provide on the Settings panel: the name of the DB2 subsystem you wish to perform Path Check functions against, and the default authorization ID to be prefixed to non-fully-qualified plan table names. In our example, these values are DB7G and DNS1, respectively.

Pressing PF3 returns you to the main menu (Figure 2). Selecting option 1 (REPORT) on the main menu brings you to the REPORT panel (Figure 4). This panel may be used to generate a Path Check Directory report. The Directory report is a summary report showing the timestamps for each bind of the selected plan or package (provided, of course, that EXPLAIN was specified for the bind).

```

Menu Utilities Options Help
-----
Path Check Interactive - REPORT
COMMAND ==>>                               Scroll ==>> CSR

*PLAN_TABLE . . . . . P390H.PLAN_TABLE      >
*Mode (PLAN or PKG) . . . . . PKG
*Plan/collection . . . . . TESTCOLLX        >
*DBRM/package . . . . . TEST01              >
Version . . . . .                            >

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

```

**Figure 4**

The 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 when preparing to submit a batch Path Check job.

The REPORT panel requires you to specify the name of the plan table against which the report is to be run, whether the report is for a plan DBRM or a package, the plan or collection name, and the DBRM or package name. In the case of a package, you may also specify the package version. In our example, we have asked for a report on DBRM TEST01 in plan TESTCOLLX.

Pressing Enter on this panel produces the report shown in Figure 5. The report shows the beginning and ending timestamps for each bind of the plan/DBRM combination recorded in the plan table.

```

Menu Utilities Options Help
-----
Path Check Interactive - REPORT
COMMAND ==>>                               Row 1 to 6 of 6
                                           Scroll ==>> CSR

Plan/collection . . . . . : TESTCOLLX      >
DBRM/package . . . . . : TEST01           >
Version . . . . . :                          >

----- Timestamp ranges -----
----- Low ----- High ----- BIND_TIME -----
2006022112410620 2006022112410702 2006-02-21-12.41.06.190626
2006022112413033 2006022112413038 2006-02-21-12.41.30.323953
2006071112350876 2006071112350948 2006-07-11-12.35.08.746576
2006071215402728 2006071215402732 2006-07-12-15.40.27.266612
2006071215404728 2006071215404733 2006-07-12-15.40.47.273526
2006071217180631 2006071217180644 2006-07-12-17.18.06.302421
***** Bottom of data *****

```

**Figure 5**



```

Menu Utilities Compilers Help
-----
BROWSE      SYS06208.T094500.RA000.DNS1.R0133132      Line 00000000 Col 001 080
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
-----
Access Path Differences (Summary)
-----
Query 163      -- prefetch mode was NONE, is now SEQUENTIAL
Query 172      -- previously was a table space scan, now uses one or
                more indexes
                -- previously required data access, now runs in INDEXONLY
                mode

-----
Access Path Differences (Detail)
-----
Query nbr 163
PLAN_TABLE column PREFETCH is different.
WAS: (blank)
NOW: S

Query nbr 172
PLAN_TABLE column ACCESTYPE is different.
WAS: R
NOW: I
PLAN_TABLE column MATCHCOLS is different.
WAS: 0
NOW: 1
PLAN_TABLE column ACCESSNAME is different.
WAS: (blank)
NOW: USER_BD_IX1
PLAN_TABLE column INDEXONLY is different.
WAS: N
NOW: Y
***** Bottom of Data *****

```

**Figure 7**

Pressing PF3 twice returns us once again to the main menu. Selecting option U (Utility functions) displays the utility submenu (Figure 8). The utility submenu provides access to various utility functions for managing and maintaining plan tables. Two utility functions are presently supported, CREATE and RESET. The CREATE function allows you to create new plan tables in the currently accessed DB2 subsystem; the RESET function “clears” an existing plan table by deleting all rows, leaving the table empty. In our example, we have chosen to illustrate the RESET function.

```

Menu Utilities Options Help
-----
Path Check Interactive - Utilities
COMMAND ==>
Select a function:
      1 CREATE   Create a PLAN_TABLE
      2 RESET   Clear a PLAN_TABLE

```

**Figure 8**

Selecting option 2 (RESET) on the utility submenu brings up the RESET panel (Figure 9).

```
Menu Utilities Options Help
-----
Path Check Interactive - Clear PLAN_TABLE
COMMAND ==>
  *Plan table name . . . . . PLAN_TABLEX

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

**Figure 9**

This panel has only one input field, which is required: the name of the plan table to be cleared. Filling in the name of an existing plan table, as we have done in Figure 9, and pressing Enter causes all rows in the plan table to be deleted. Successful completion of this operation causes a status message to be displayed as shown in Figure 10.

```
Menu Utilities Options Help
-----
Path Check Interactive - Clear PLAN_TABLE
COMMAND ==>
  *Plan table name . . . . . PLAN_TABLEX

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

+-----+
| CKP1034I Table cleared (all rows deleted) |
+-----+
```

**Figure 10**

# Appendices

---

## Appendix A — JCL Examples

The following JCL samples illustrate a typical series of job steps for Path Check using the JCL parm field to specify processing options.

*Note:* use of the JCL parm field is optional. Path Check may be controlled either through the parm field, or by commands entered via the control data set (ddname (SYSIN). SYSIN and the parm field are mutually exclusive; use one or the other but not both. HLS recommends using the command interface; the JCL parm field is supported for compatibility with previous releases.

For each of the following examples, the command shown in the example is assumed to be preceded by the following JCL.

```
//          EXEC   PGM=PATHCHK
//STEPLIB  DD   DSN=pathchkloadlib,DISP=SHR
//          DD   DSN=db2loadlib,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSEXPLN DD   SYSOUT=*
//DBRMIN   DD   DISP=SHR,DSN=dbrmlib
//SYSIN    DD   *
-- LINES THAT START WITH--ARE COMMENTS--THIS TEST WILL REPORT ON ALL THE
PACKAGES IN COLLECTION--'TESTCOLL' AND ALL DBRMS IN PLAN
'TESTPLAN'
  CONNECT TO DSN1
  SET CATALOG QUALIFIER = 'SYSIBM'
  REPORT ON PACKAGE TESTCOLL.* IN P390H9.PLAN_TABLE
  REPORT ON PLAN      TESTPLAN.* IN P390H9.PLAN_TABLE
```

### Example 1

```
REPORT ON PACKAGE TESTCOLL.TESTDBRM IN P390H.PLAN_TABLE
```

This request will produce a report on the access path for program TESTCOLL.TESTDBRM from the data in the specified plan table.

### Example 2

```
REPORT ON PACKAGE TESTCOLL.* IN P390H.PLAN_TABLE START WITH A
```

This will produce a report on the most current versions of all the programs for the collection TESTCOLL and the most current access path in the plan table. The START

WITH clause will start the report at the first program name whose name is greater than or equal to A.

### Example 3

```
REPORT ON PLAN TESTPLAN.AA* IN P390H.PLAN_TABLE START WITH A
```

This will produce a report on all the program names that begin with 'AA' in plan TESTPLAN. The START WITH clause is specified correctly but all the program names that begin with 'AA' would be greater than 'A' in any case.

### Example 4

```
COMPARE PACKAGE TESTCOLL.TESTDBRM IN USERID.PLANTABLE TO TESTCOLL.*  
IN USERID.PLANTABLE2
```

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 should identify 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.

### Example 5

```
COMPARE PACKAGE TESTCOLL.* IN USERID.PLANTABLE TO SAVECOLL.* IN  
USERID.PLANTABLE START WITH TESTDBRM
```

This request will compare all the package names (program names) in collection TESTCOLL as if they had previously been bound into collection SAVECOLL. The DBRM name can be specified with wild cards and the list of names to be processed will be selected from SYSPACKAGE. This request will start with program name TESTDBRM.

### Example 6

```
COMPARE PLAN TESTPLAN.TESTDBRM IN USERID.PLANTABLE TO TESTPLN2.* IN  
USERID.PLANTABLE
```

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. This request will compare the access paths for program TESTDBRM to data in the plan table as if the plan had been bound with the name TESTPLN2.

### Example 7

```
COMPARE PLAN TESTPLAN.* IN USERID.PLANTABLE TO TESTPLN2.* IN  
USERID.PLANTABLE START WITH A
```

This request will compare the access paths for all the programs in plan TESTPLAN 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.

### Example 8

```
SET CURRENT SQLID = 'PROD'  
TEST DBRM TESTDBRM AS PACKAGE COLLNAME.* IN USERID.PLANTABLE
```

TEST processing requires that the SQL ID be set to the qualifier of the tables to be processed.

Read the DBRM TESTDBRM from DDNAME DBRMIN, *Explain* each access path and compare to the previous access path. This will identify potential access path changes.

### Example 9

```
SET CURRENT SQLID = 'PROD'  
TEST DBRM * AS PACKAGE COLLNAME.* IN USERID.PLANTABLE START WITH  
NAXX
```

The DBRM name can be specified with wild cards and the list of names to be processed will be determined from SYSIBM.SYSPACKAGE. This request will process all the programs in the collection COLLNAME starting with program NAXX. The START WITH clause is optional.

### Example 10

```
SET CURRENT SQLID = 'PROD'  
TEST DBRM TESTDBRM AS PLAN TESTPLAN.* IN USERID.PLANTABLE
```

Read the DBRM TESTDBRM from DDNAME DBRMIN, *Explain* each access path and compare to the previous access path. This will identify potential access path changes.

### Example 11

```
SET CURRENT SQLID = 'PROD'  
TEST DBRM TEST* AS PLAN TESTPLAN.* IN USERID.PLANTABLE
```

The DBRM name can be specified with wild cards and the list of names to be processed will be determined from SYSIBM.SYSDBRM. This request will process all the program names that start with TEST from plan TESTPLAN.

### Example 12

```
MAKE STATEMENT 134 WITH TIMESTAMP = '2001012311210955' IN PACKAGE  
TESTCOLL.PROGNAME IN USERID.PLAN_TABLE BE HINT PROD
```

This example assigns the optimization hint PROD to a statement in package PROGNAME of collection TESTCOLL. The timestamp and statement number uniquely identify the plan table row to be modified by the addition of a hint.

### Example 13

```
REPORT ON PACKAGE TESTCOLL.* IN P390H.PLAN_TABLE DIRECTORY
```

This command generates a summary report (DIRECTORY) for all occurrences of collection TESTCOLL.

### Example 14

```
REPORT ON PLAN TESTPLAN.TESTDBRM IN P390H.PLAN_TABLE BEFORE  
'2002121222435133'
```

This command lists the access path information for program TESTDBRM in plan TESTPLAN for binds prior to the specified date/time.

### Example 15

```
REPORT ON PLAN TESTPLAN.TESTDBRM IN P390H.PLAN_TABLE DIRECTORY
```

This command generates a summary report (DIRECTORY) for all binds of plan TESTPLAN and program TESTDBRM.

### Example 16

```
COMPARE PLAN TESTPLAN.TESTDBRM IN P390H.PLAN_TABLE TO PREVIOUS
```

This example compares the current access path data for plan TESTPLAN/program TESTDBRM with that of the last previous bind.

### Example 17

```
COMPARE PLAN TESTPLAN.TESTDBRM IN P390H.PLAN_TABLE  
TO TESTPLAN.* IN P390H.PLAN_TABLE BEFORE '2002121222435133'
```

This example compares the current access path data for plan TESTPLAN with that of the bind prior to the date and time specified.

### Example 18

```
SET CURRENT SQLID = 'P390H'  
EXPLAIN DBRM TESTDBRM
```

This set of commands will identify the access path(s) that would be chosen by DB2 if the DBRM were to be bound now.

# Appendix B — REPORT Examples

## Example 1 – SYSPRINT report

Example SYSPRINT report that shows the difference between the most current access path and the prior access path. The DBRM was pre-compiled with VERSION(AUTO) and is a single table SELECT statement

```

PCK031I EXECUTING LICENSED PATH CHECK   V2R44
11          OPTIONS CATALOGSQL REPORTCOSTGT
11          CONNECT TO DB7G
11          COMPARE   PACKAGE TESTCOLLX.TEST01 IN PUBLIC.PLAN_TABLE
              TO PREVIOUS

11 ACCESS PATH FOR      COLLID - TESTCOLLX      .TEST01  SQL ID-      OLD PLAN_TABLE -      RUN DA
PCK212I 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 QBNO PLNO MXSQ M
*   327 0 PUBLIC    SYSCOLUMNS                1  I  2 PUBLIC    DSNDXC01        N NNNNNNNN IS L  1  1
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 I
PCK221I DATA CHANGE FOR COLUMN MATCHCOLS     IS NOW 2
PCK221I DATA CHANGE FOR COLUMN ACCESSCREATOR  IS NOW PUBLIC
PCK221I DATA CHANGE FOR COLUMN ACCESSNAME     IS NOW DSNDXC01
PCK221I DATA CHANGE FOR COLUMN PREFETCH =     IS NOW L
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
PCK204I  QUERIES WITH MATCHING EXPLAIN      2      QUERIES WITHOUT MATCHING EXPLAIN      0      FOR PROGRAM
COMPLETE RET CODE=      04

```

REPORT Contents.

**The SYSPRINT report provides the access path data from the PLAN\_TABLE, estimated service unit costs from DSN\_STATEMNT\_TABLE if available and details all the changed data values from PLAN\_TABLE.**

## Example 2 – SYSEXPLN Report

Same example but SYSEXPLN version of the report

```

11 ACCESS PATH FOR      COLLID - TESTCOLLX      .TEST01  SQL ID-      OLD PLAN_TABLE -      RUN I
STATEMENT      327
STATEMENT#      327 DECLARE SELECT-1 CURSOR FOR SELECT NAME , TBNAME , COLNO , C
              OLTYPE , LENGTH FROM SYSCOLUMNS WHERE TBNAME = : H AND TBCRE

```

ATOR = : H

```
STEP      1  ACCESSES TABLE      PUBLIC .SYSCOLUMNS
            USING 2 COLUMN(S). OF INDEX PUBLIC .DSNDCX01
            THE ACCESS WILL USE LIST PREFETCH
            THE INTENT LOCK FOR THE TABLE IS IS
            THE TIMESTAMP FOR THIS EXPLAIN IS 2006022112412245
            THE TABLE HAS          -1 ROWS OF      969 BYTES.
            THE NON-UNIQUE INDEX HAS          -1 ENTRIES IN -1 LEVELS WITH CLUSTER RATIO 0
PCK221I DATA CHANGE FOR COLUMN ACESSTYPE          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
PCK221I DATA CHANGE FOR COLUMN ACESSTYPE          IS NOW I
PCK221I DATA CHANGE FOR COLUMN MATCHCOLS          IS NOW 2
PCK221I DATA CHANGE FOR COLUMN ACCESSCREATOR      IS NOW PUBLIC
PCK221I DATA CHANGE FOR COLUMN ACCESSNAME        IS NOW DSNDCX01
PCK221I DATA CHANGE FOR COLUMN PREFETCH =        IS NOW L
PCK206I COMPARE COMPLETE FOR PROGRAM = TEST01  VERSION = 2005-04-11-18.34.53.754785
                                           PREVIOUS VERSION = 2005-04-11-18.20.42.913225
```

REPORT Contents.

**The SYSEXPLN report provides the SQL text where available, an English like description of the access path, detailed list of access changes and estimated costs.**

### Example 3 – SYSCHG Report

Same example but SYSCHG version of the report

```
PCK218I PROGRAM TEST01      HAS A CHANGED ACCESS PATH
      QUERYNO                327
      OLD PROCSU              209
      NEW PROCSU              139
      COLLECTION              TESTCOLL
      NEW VERSION              2005-04-11-18.34.53.754785
      OLD VERSION              2005-04-11-18.20.42.913225
PCK217I TOTAL DBRMS COMPARED                                1
PCK217I TOTAL STATEMENTS COMPARED                          2
PCK217I TOTAL ACCESS PATH CHANGES                          1
PCK217I TOTAL ACCESS PATH SAME                              1
PCK217I TOTAL NEW SQL STATEMENTS                            0
PCK217I TOTAL DELETED SQL STATEMENTS                        0
PCK217I TOTAL DSN_STATEMNT COST INCR                        0
```

REPORT Contents.

**The SYSCHG report only identifies the presence of a change and the type of change. It is designed to be used for quick analysis.**

### Example 4 – SYSPRINT report with access path structure change.

An access path structure change is when the tables are not referenced in the same sequence. Notice the summary showing the different sequence for the tables accessed. The old access path referenced the SYSTABLES table first and the new access path references SYSCOLUMNS first.

```

11          COMPARE    PACKAGE TESTCOLLX.TEST07 IN PUBLIC.PLAN_TABLE
                TO PREVIOUS

11 ACCESS PATH FOR    COLLID - TESTCOLLX          .TEST07    SQL ID-          OLD PLAN_TABLE -          RUN I
   PCK212I TIMESTAMP USED FOR PREVIOUS COMMAND = 2006022112404737    VERSION = 2004-02-25-22.55.53.186647

      COLLID - TESTCOLLX          .TEST07    327
PLAN OLD TABLE          TNO MTD AC MC PLAN NEW TABLE          TNO MTD AC MC
  1 SYSTABLES            2 0 I  2          1 SYSCOLUMNS            1 0 I  2
  2 SYSCOLUMNS          1 1 I  2          2 SYSTABLES            2 1 I  2

IN  QRYNO M CREATOR  TNAME          TBNO AC MC CREATOR ACCESSNAME          IO SORTUJOG LK PF FN QBNO PLNO MXSQ
OLD  327 0 PUBLIC   SYSCOLUMNS          1 I  2 PUBLIC   DSNDX01          N NNNNNNNN IS L  1  1
OLD  327 1 PUBLIC   SYSTABLES            2 I  2 PUBLIC   DSNDTX01          N NNNNNNNN IS  1  2
PCK224I DSN_STATEMNT_TABLE ESTIMATED COST CHANGE - OLD EST SVC UNITS  34,192

IN  QRYNO M CREATOR  TNAME          TBNO AC MC CREATOR ACCESSNAME          IO SORTUJOG LK PF FN QBNO PLNO MXSQ
NEW  327 0 PUBLIC7   SYSTABLES            2 I  2 PUBLIC7   DSNDTX01          N NNNNNNNN IS  1  1
NEW  327 1 PUBLIC7   SYSCOLUMNS          1 I  2 PUBLIC7   DSNDX01          N NNNNNNNN IS  1  2
PCK223I DSN_STATEMNT_TABLE ESTIMATED COST CHANGE - NEW EST SVC UNITS  1

```

. REPORT Contents.

**This SYSPRINT report has 3 sections. The first 4 lines 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. Notice that there is no comparison of the data items from the access path because the structure of the access path has changed.**

### Example 5 – History Table

History table for a summary of changes detected in Path Check processing. This table contains the same information as the SYSCHG ddname report.

The DDL for the History table is:

```

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
, PROGNAME_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
, HINT_USED_OLD       VARCHAR(128)      NOT NULL WITH DEFAULT
, HINT_USED_NEW       VARCHAR(128)      NOT NULL WITH DEFAULT
, PROCSU_OLD          INTEGER           NOT NULL WITH DEFAULT
, PROCSU_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;

```

The Control Card for the History table is:

```

OPTIONS HISTORY TABLE P390H.HIST_TABLE

```

The work file to collect the data is:

```

//HISTWORK DD DSN=P390H.TEST.HISTWORK,DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(TRK,(30,30)),
//          DCB=(RECFM=VB,LRECL=854,BLKSIZE=0)

```

**Note:** If the HISTWORK work file ddname exists, and the OPTIONS HISTORY TABLE is not specified, the data is written to the work file and not inserted to the table.

The Data can be loaded with:

```

LOAD DATA LOG YES RESUME YES INDDN SYSREC00 INTO TABLE
P390H.HIST_TABLE
(
APPLNAME_OLD          POSITION(      5      )
CHAR(                  8) ,
APPLNAME_NEW          POSITION(     13      )
CHAR(                  8) ,
COLLID_OLD            POSITION(     21      )
VARCHAR                ,
COLLID_NEW            POSITION(    151      )
VARCHAR                ,
PROGNAME_NEW          POSITION(     281     )
VARCHAR                ,
VERSION_OLD           POSITION(     411     )
VARCHAR                ,
VERSION_NEW           POSITION(     477     )
VARCHAR                ,
QBLOCK_TYPE_OLD       POSITION(     543     )
CHAR(                  6) ,
QBLOCK_TYPE_NEW       POSITION(     549     )
CHAR(                  6) ,
QUERYNO_OLD           POSITION(     555     )
INTEGER                ,
QUERYNO_NEW           POSITION(     559     )
INTEGER                ,

```

```

HINT_USED_OLD          POSITION( 563 )
VARCHAR                ,
HINT_USED_NEW          POSITION( 693 )
VARCHAR                ,
PROCSU_OLD             POSITION( 823 )
INTEGER                ,
PROCSU_NEW             POSITION( 827 )
INTEGER                ,
CHANGE_REASON          POSITION( 831 )
CHAR( 20)
)

```

*Note:* 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.

All Path Check messages and diagnostics are written to SYSPRINT. Upon normal completion one of the following condition codes will be returned:

Return Code	Meaning
0	Path Check processing has completed successfully. There were no changes to Access Paths.
4	Path Check has determined that one or more access path changes occurred.
12	Trial version of Path Check has expired
16	Fatal error; contact Technical Support.
44	Path Check has determined that one or more access path changes occurred; data was passed to SQL/PA via ddname ANLOUT.

## Appendix C — Messages

Path Check messages are of the form PCKnnnt, where *nnn* is the message number and *t* is a one-character suffix with the value I, E, or S. These suffixes have the following meanings:

I	Informational
E	Error
S	Severe Error

Each Path Check message is listed below along with its corresponding explanation.

---

**PCK031I EXECUTING LICENSED PATH CHECK V2R4**

**Explanation:** Informational message. Indicates that the version of Path Check is a licensed version.

---

**PCK032I PATH CHECK TRIAL NOT ACTIVATED**

**Explanation:** Informational message. Indicates that the trial period has not begun and that Path Check is not yet operational. Contact HLS Technical Support.

---

**PCK033I EXECUTING TRIAL VERSION OF PATH CHECK**

**Explanation:** Informational message. Indicates that a trial copy of Path Check is executing.

---

**PCK034I TRIAL VERSION OF PATH CHECK EXPIRED**

**Explanation:** Information message. Indicates that the trial period has expired. Contact HLS Technical Support.

---

**PCK035E DBRMIN PROCESS FAILURE**

**Explanation:** Path Check was not able to open the DBRMIN. Verify that the ddname is present in the JCL and that it references a valid DBRMLIB.

---

**PCK036E DBRMIN DDNAME MISSING**

**Explanation:** Path Check did not detect a dd card for DBRMIN. Verify that the ddname is present in the JCL and that it references a valid DBRMLIB. This dd name is required for TEST and EXPLAIN commands except for the TEST command that specifies FROM CATALOG.

---

**PCK038E      SYSIN DDNAME MISSING**

**Explanation:** The JCL PARM field did not specify a process request, nor was any SYSIN DD statement present.

---

**PCK039E      CRITICAL INSTALL ERROR**

**Explanation:** Path check was not able to initialize or process successfully. Contact HLS technical support.

---

**PCK041E      NO SET CURRENT SQLID BEFORE TEST**

**Explanation:** The TEST command was issued without first issuing SET CURRENT SQLID. TEST requires that a valid SET CURRENT SQLID be successfully processed first.

---

**PCK042E      CONNECT TO DB2 REQUIRED BEFORE OTHER COMMANDS**

**Explanation:** Path Check did not receive a valid CONNECT TO ssid statement before the process request. Change the SYSIN commands to include a valid CONNECT statement.

---

**PCK043S      MVS CONVERSION SERVICES FROM UTF-8 AND UTF-16 REQUIRED TO PROCESS UNICOD DBRM**

**Explanation:** Path Check uses MVS conversion services to decode DB2 V8 unicode DBRM's. This error indicates that Path Check was not able to convert the SQL text.

---

**PCK044E      TEST FROM CATALOG REQUIRES DDNAME - DBRMOUT TEMPORARY DATASET**

**Explanation:** An attempt to create the temporary DBRM for a test command failed because the DBRMOUT ddname was missing. Severity: 16 Programmer

Response: Insert a DD card for DBRMOUT into the JCL stream. Refer to "DBRMOUT DD statement" on page 13, for more information on the DRBMOUT statement.

---

**PCK045S      ERROR IN BUILDING DBRM FROM SYSTEM CATALOG**

**Explanation:** Path Check was not able to retrieve the SQL text from the catalog. Contact HLS Technologies technical support.

---

**PCK047E      command REQUIRES PLAN\_TABLE**

**Explanation:** Path check will not attempt the test or explain command and return code = 8

---

**PCK048I      COST ESTIMATES REQUIRES DSN\_STATEMNT\_TABLE**

**Explanation:** Path check will process the test or explain command but will not report estimated costs.

---

---

**PCK053I DBRM IS EMPTY**

**Explanation:** The DBRM did not contain any SQL. This is the normal result of processing a program through the DB2 precompiler when it does not contain any SQL. The DBRM will be bypassed.

---

**PCK201I INPUT PARAMETERS**

**Explanation:** This is a routine message issued at the beginning of each run. It simply lists the input parameters supplied by the user in parameters.

---

**PCK202S DBRM ERROR**

**Explanation:** Path check was not able to successfully parse the DBRM. Verify that the DBRM has not been modified since the precompile and contact HLS Technical Support

---

**PCK203I STATEMENTS WITH SAME ACCESS PATH STATEMENTS WITH DIFFERENT  
ACCESS PATH FOR PROGRAM progname**

**Explanation:** Informational message. Normal message for the end of a COMPARE or TEST command that identifies the number of SQL statements that had changed access path for the same SQL.

---

**PCK204I QUERIES WITH MATCHING EXPLAIN nnn QUERIES WITHOUT MATCHING EXPLAIN  
nnn FOR PROGRAM progname**

**Explanation** Normal message for the end of a COMPARE or TEST command that identifies the number of SQL statements that did not match the previous access data.

---

**PCK205E NO DATA ON PLAN\_TABLE FOUND FOR dbrmname**

**Explanation:** The plan table does not contain any usable rows for the named DBRM.

---

**PCK206I PCK206I COMPARE COMPLETE FOR PROGRAM = TEST01 VERSION = 2006-04-24-  
22.45.2**

**BIND-TIME = 2006-12-20-11.44.1  
PREVIOUS VERSION = 2005-04-11-18.20.4  
PREVIOUS BIND-TIME = 2006-12-20-11.44.0**

**Explanation:** Normal information message at completion of a command. For EXPLAIN or TEST command this will identify the version identifier from the precompile of the DBRM. For COMPARE or TEST command it will identify the version information of the old access data. It shows the bind times for plan\_table rows.

---

---

**PCK207I      BYPASS SPECIFIED AND      PROGRAMS BYPASSED ENDING WITH**

**Explanation:** Informational message. Indicates that a START WITH option was specified and the report has started with the program after the named program.

---

**PCK208I      MORE THAN 8000 PROGRAMS IDENTIFIED - FIRST 8000 WILL BE PROCESSED**

**Explanation:** Informational message. Indicates that the number of programs identified by the wild card name exceeded 8000 and the report will only include the first 8000.

---

**PCK209I      MEMBER      NOT FOUND IN DBRMIN DDNAME**

**Explanation:** Informational message. Indicates that a program name specified in a TEST or EXPLAIN command was not available in the DBRMIN ddname.

---

**PCK210I      NO PACKAGES FOUND IN SYSPACKAGE FOR**

**Explanation:** Informational message. Indicates that a program name with wild card specified did not identify any program names from SYSIBM.SYSPACKAGE.

---

**PCK211I      NO DBRMS FOUND IN SYSDBRM FOR**

**Explanation:** Informational message. Indicates that a program name with wild card specified did not identify any program names from SYSIBM.SYSDBRM.

---

**PCK212I      TIMESTAMP USED FOR PREVIOUS COMMAND = 2006122011440166**  
**VERSION = 2005-04-11-18.20.42.913225**  
**BIND\_TIME = 2006-12-20-11.44.00.854993**  
**TIMESTAMP USED FOR CURRENT = 2006122011441849**  
**VERSION = 2006-04-24-22.45.29.330423**  
**BIND\_TIME = 2006-12-20-11.44.18.383696**

**Explanation:** Informational message. Indicates the timestamp value that was used for the PREVIOUS option on a COMPARE command.

---

**PCK213S      INVALID COLUMN DEFINITION FOR COLUMN**

**Explanation:** Severe error. The plan table referenced in a command contained a column that did not match the required format. If the plan table is usable with BIND EXPLAIN(YES), contact HLS technical support.

---

**PCK214I      EXPLAIN FAILED FOR STATEMENT   *sss*      SQLCODE =   *nnn***

**Explanation:** Path Check was unable to execute a successful EXPLAIN of the indicated statement. In the message text, *sss* is the statement number of the failing statement and *nnn* is the sqlcode returned by DB2 for the attempted EXPLAIN. Normally this indicates that the specified statement is not an explainable.statement according to DB2. If you are able to EXPLAIN the statement successfully without Path Check, contact HLS technical support.

---

**PCK215E DATA COLUMN DID NOT MATCH AVAILABLE FORMAT**

**Explanation:** Fatal error : The plan table definition did not match one of the DB2 versions supported by Path Check.

---

**PCK217I TOTAL type count**

**Explanation:** Informational. This message lists the total objects processed. Total types are:

**DBRMS COMPARED – Total number of DBRM's processed in this step execution.**

**STATEMENTS COMPARED – Total number of SQL statements with access path verified.**

**ACCESS PATH CHANGES – Total number of access path changes detected.**

**ACCESS PATH SAME – Total number of SQL statements with access data that remained the same.**

**NEW SQL STATEMENTS – Total number of new SQL statements detected.**

**DELETED SQL STATEMENTS – Total number of deleted SQL statements detected.**

**DSN\_STATEMNT COST INCR – Option REPORTCOSTGT was specified, the number of SQL statements with DB2 estimated increase in processing cost.**

---

**PCK218I PROGRAM progname change**

**Explanation:** Informational. This SYSCHG message lists the program name that has one of

**HAS A CHANGED ACCESS PATH**

**HAS A NEW SQL STATEMENT**

**HAS A DELETED SQL STATEMENT.**

---

**PCK220I DATA CHANGE FOR COLUMN colname WAS value**

**Explanation:** Informational. This SYSPRINT message identifies the old data value for detail data value changes for access paths that only had minor changes. This message does not appear when the table sequence changes for a JOIN, an access path changes to multiple index, or the number of access steps changes.

---

**PCK221I DATA CHANGE FOR COLUMN colname IS NOW value**

**Explanation:** Informational. This SYSPRINT message identifies the new data value for detail data value changes for access paths that only had minor changes. This message does not appear when the table sequence changes for a JOIN, an access path changes to multiple index, or the number of access steps changes.

---

**PCK223I DSN\_STATEMNT\_TABLE ESTIMATED COST CHANGE - NEW EST SVC UNITS value**

**Explanation:** Informational. This SYSPRINT message lists the new estimated service units for this SQL statement from DSN\_STATEMNT\_TABLE for an SQL statement that was identified by REPORTCOSTGT processing.

---

**PCK224I DSN\_STATEMNT\_TABLE ESTIMATED COST CHANGE - OLD EST SVC UNITS value**

**Explanation:** Informational. This SYSPRINT message lists the old estimated service units for this SQL statement from DSN\_STATEMNT\_TABLE for an SQL statement that was identified by REPORTCOSTGT processing.

---

**PCK225I DSN\_STATEMNT\_TABLE ESTIMATED COST – EST SVC UNITS value**

**Explanation:** Informational. This SYSPRINT message lists the estimated service units for this SQL statement from DSN\_STATEMNT\_TABLE for an SQL statement listed from a REPORT command

---

**PCK230I ACCESS PATH IS FROM HINT hint name**

**Explanation:** Informational. This SYSPRINT message identifies the hint that was used to specify this access path.

---

**PCK240I BINDIN CONTROL CARDS DO NOT SPECIFY EXPLAIN(YES)**

**Explanation:** Informational. This SYSPRINT message identifies a bind control card was read from BINDIN that did not specify EXPLAIN(YES). This means that any access path analysis will use information from prior BIND's. not the current access path.

---

**PCK1000S UNRECOVERABLE LOGIC ERROR IN PCKA100, SYMPTOM=*nnn***

**Explanation:** A program logic error has occurred. In the message text, *nnn* is an internal diagnostic code for use by Technical Support. Perform the same sequence of steps to determine if the error is re-creatable. If it is, record the exact sequence of actions that led to the error and the symptom value, and contact HLS Technical Support.

---

**PCK1001S MODULE *xxxxxxx* COULD NOT BE LOADED, REASON=*nnnn***

**Explanation:** A required program could not be accessed. In the message text, *xxxxxxx* is the name of the failing module and *nnnn* is an internal diagnostic code for use by HLS Technical Support. Make the specified module available to your ISPF session, either through STEPLIB or Link List. If you are certain that it is available and you still receive this message, contact HLS Technical Support.

---

**PCK1002E DSNTIAR FAILURE, RC=*nnnn***

**Explanation:** DSNTIAR was called to format a DB2 message and returned a non-zero condition code. In the message text, *nnnn* is the return code from DSNTIAR. Locate the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If that does not resolve the problem, contact HLS Technical Support and provide the information in the messages text.

---

**PCK1003S ISPTIAR ROUTINE INVOKED FROM TRACEPOINT *nn***

**Explanation:** This is a diagnostic message that is used for debugging purposes. It indicates which internal code path was active when a DB2 error occurred. Contact HLS Technical Support and provide them with the contents of this message if needed.

---

**PCK1004E INVALID ENTRY**

**Explanation:** This message is displayed when an invalid selection is made on a menu. For example, when the choice that was entered is not one of the available options. Enter a valid selection from the menu.

---

**PCK1009S Path Check has expired, contact HLS.**

**Explanation:** Self explanatory. The trial period has expired.

---

**PCK1010E NO SUBSYSTEM NAME HAS BEEN SPECIFIED**

**Explanation:** A Path Check function was requested but a valid subsystem name was not supplied. Specify a valid subsystem on the Settings panel (option 0 on the Main Menu).

---

**PCK1011E DB2 CONNECT FAILED, SYMPTOMS=*nnn,nnn,nnn***

**Explanation:** Path Check was unable to connect to the specified DB2 subsystem. In the message text, *nnn* represents an internal diagnostic code for use by HLS Technical Support. Verify that the specified subsystem is running and is accessible from the system where you are running Path Check, and try again. If this does not resolve the problem, contact HLS Technical Support.

---

**PCK1012E DB2 DISCONNECT FAILED, SYMPTOMS=*nnn,nnn,nnn***

**Explanation:** Path Check was unable to successfully disconnect from the specified DB2 subsystem. In the message text, *nnn* represents an internal diagnostic code for use by HLS Technical Support. Verify that the specified subsystem is running and is accessible from the system where you are running Path Check, and try again. If this does not resolve the problem, contact HLS Technical Support.

---

**PCK1013S xxxxxx n FAILED, RC=*nnn***

**Explanation:** An ISPF service failed with a nonzero condition code. In the message text, *xxxxxx* is the failing service, *n* is an internal diagnostic code, and *nnn* is the condition code returned by

the ISPF service. Locate the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If that does not resolve the problem, contact HLS Technical Support and provide the information in the messages text.

---

**PCK1016I NO DIFFERENCES WERE DETECTED**

**Explanation:** This is an informational message. The COMPARE operation did not detect any changed access paths.

---

**PCK1017E THE 'OLD' PLAN\_TABLE DOES NOT EXIST IN THIS DB2 SUBSYSTEM**

**Explanation:** The plan table specified on the panel does not exist. Specify a valid plan table name.

---

**PCK1018E THE 'NEW' PLAN\_TABLE DOES NOT EXIST IN THIS DB2 SUBSYSTEM**

**Explanation:** The plan table specified on the panel does not exist. Specify a valid plan table name.

---

**PCK1019E PLAN\_TABLE NAME(S) MUST BE FULLY QUALIFIED (NO DEFAULT AUTHID WAS SPECIFIED ON THE SETTINGS PANEL)**

**Explanation:** The PLAN table name that was entered on the panel is not fully qualified, and no default authorization ID has been assigned. Either specify a fully qualified PLAN table name, or go to the SETTINGS panel (Main Menu option 0) and specify a default authorization ID.

---

**PCK1020E FULLY QUALIFIED PLAN\_TABLE NAME IS TOO LONG TO FIT ON ONE PATH CHECK COMMAND LINE**

**Explanation:** The plan table name must be short enough to fit on a single line. Specify a PLAN table with a shorter name.

---

**PCK1021E PLAN TABLE NAME(S) MUST BE FULLY QUALIFIED (NO DEFAULT AUTHID WAS SPECIFIED ON THE SETTINGS PANEL)**

**Explanation:** The plan table name that was entered on the panel is not fully qualified and no default authorization ID has been assigned. Either specify a fully qualified plan table name or specify a default authorization ID on the SETTINGS panel (main menu option 0).

---

**PCK1022E OPEN FAILED FOR CURSOR nnnn, SQLCODE=ccc**

**Explanation:** A DB2 interface error has occurred. In the message text, *ccc* is the SQLCODE associated with an OPEN CURSOR request. Verify that the subsystem that was specified on the SETTINGS panel is valid and that the subsystem is available.

---

**PCK1023E FETCH FAILED FOR CURSOR nnnn, SQLCODE=ccc**

**Explanation:** A DB2 interface error has occurred. In the message text, *ccc* is the SQLCODE associated with an FETCH request. Verify that a valid subsystem is specified on the SETTINGS panel and that the subsystem is available.

---

**PCK1024E PLAN TABLE NAME(S) MUST BE FULLY QUALIFIED (NO DEFAULT AUTHID WAS SPECIFIED ON THE SETTINGS PANEL)**

**Explanation:** The plan table name that was entered on the panel is not fully qualified, and no default authorization ID has been assigned. Either specify a fully qualified plan table name or a default authorization ID on the SETTINGS panel (main menu option 0).

---

**PCK1025S INVALID DATA IN SYSIBM.SYSCOLUMNS, SYMPTOM=*nnn***

**Explanation:** The DB2 catalog contains invalid data. In the message text, *nnn* is a diagnostic code for use by HLS Technical Support. Locate the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If that does not resolve the problem, contact HLS Technical Support and provide the information in the message text.

---

**PCK1026S VGET FAILURE, TYPE *n*, RC=*nnn***

**Explanation:** An ISPF interface error has occurred. In the message text, *n* is an internal diagnostic code for use by Technical Support and *nnn* is the return code from the VGET service. Verify that the TSO user has a valid ISPPROF data set allocated and that the user has read access to the data set.

---

**PCK1027S VPUT FAILURE, TYPE *n*, RC=*nnn***

**Explanation:** An ISPF interface error has occurred. In the message text, *n* is an internal diagnostic code for use by HLS Technical Support and *nnn* is the return code from the VPUT service. Verify that the TSO user has a valid ISPPROF data set allocated and that the user has update access to the data set.

---

**PCK1028E NO DATABASE NAME WAS SPECIFIED (DB2 SYNTAX DOES NOT PERMIT UNQUALIFIED TABLESPACE NAME)**

**Explanation:** If a table space name is specified, a database name must also be specified. Either provide a database name or leave the table space name blank.

---

**PCK1029E CREATE TABLE FAILED, SQLCODE=*ccc***

**Explanation:** A DB2 interface error has occurred. In the message text, *ccc* is the SQLCODE associated with a CREATE TABLE request. Verify that the TSO user has a valid ISPTABL data set allocated and that the user has update access to the data set.

---

**PCK1030I 51 COLUMN PLAN TABLE CREATED**

**Explanation:** This is an informational message. A plan table has been created in Version 7 format (51 columns).

---

---

**PCK1031I 58 COLUMN PLAN TABLE CREATED**

**Explanation:** This is an informational message. A plan table has been created in Version 8 format (58 columns).

---

**PCK1032E CREATE TABLE FAILED, SQLCODE=ccc**

**Explanation:** A DB2 interface error has occurred. In the message text, *ccc* is the SQLCODE associated with a CREATE TABLE request. Verify that the TSO user has a valid ISPTABL data set allocated and that the user has update access to the data set.

---

**PCK1033E DELETE FAILED, SQLCODE=ccc**

**Explanation:** A DB2 interface error has occurred. In the message text, *ccc* is the SQLCODE associated with a DELETE request. Verify that a valid subsystem is specified on the SETTINGS panel and that the subsystem is available.

---

**PCK1034I TABLE CLEARED (ALL ROWS DELETED)**

**Explanation:** This is an informational message. All rows have been deleted from the specified plan table.

---

**PCK1035I BIND REQUIRED -- THE PATHCHK PROGRAM CONTOKEN IS NOT CURRENT**

**Explanation:** The PATHCHK program that is being run does not match the last bind recorded in the DB2 subsystem. A bind is required. Bind the PATHCHK program.

---

**PCK1036S PCKPTHCK LINKAGE FAILURE, RC=nnn**

**Explanation:** A failure occurred in the PATHCHK program. In the message text, *nnn* is the return code (condition code) from PATHCHK. Locate the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If that does not resolve the problem, contact HLS Technical Support and provide the message information.

---

**PCK1037I TABLE DOES NOT EXIST**

**Explanation:** A RESET operation was requested for a nonexistent plan table. Specify a valid table name.

---

**PCK1038I TABLE ALREADY EXISTS**

**Explanation:** A CREATE operation was requested for a plan table that already exists. If the plan table name is correct, no action is required.

---

**PCK1039E DB2 COMMIT FAILED, SYMPTOMS=nnn**

**Explanation:** A DB2 interface error has occurred. In the message text, *nnn* is the SQLCODE associated with a COMMIT request. Verify that a valid subsystem is specified on the SETTINGS panel and that the subsystem is available.

---

**PCK1040E DB2 COMMIT FAILED, SYMPTOMS=*nnn***

**Explanation:** A DB2 interface error has occurred. In the message text, *nnn* is the SQLCODE associated with a COMMIT request. Locate the error information in DB2 Messages and DB2 SQL Codes. Where applicable, perform the recommended action. If that does not resolve the problem, contact HLS Technical Support and provide the message information.

---

**BND104E COMPARE FAILED, ONE OR MORE DBRMS DID NOT MATCH THE DB2 CATALOG OR PLAN NOT FOUND**

**Explanation:**

---

**CKP1001S Module DSNTIAR could not be loaded, reason=9250**

**Explanation:**

---

## Appendix D — Parameter style input processing

### EXEC Statement

The format of the EXEC statement is:

```
//stepname EXEC PGM=PATHCHK,PARM='parameters'
```

where:

**PGM=PATHCHK**

specifies that you want to run the Path Check program.

**PARM='parameters'**

The JCL parm field is supported for compatibility with previous releases. Path Check may be controlled either through the parm field, or by commands entered via the control data set (ddname (SYSIN)). SYSIN and the parm field are mutually exclusive; use one or the other but not both. HLS recommends using the command interface.

When using the parm field to control Path Check, the following parameters may be specified:

```
'ssid,request,creator,table,entity,progrname,report'
```

You may supply values for any valid combination of the above parameters. The parameters are positional, i.e., must be specified in the order shown. If any parameter (except the final one) is omitted, include the trailing comma that normally would follow it. The individual parameters are described below.

*ssid*

Specify the DB2 subsystem name.

*request*

Specify the processing to be done. Acceptable values are:

**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.

**COMPPLAN** 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.

**TESTPLAN** Read DBRMLIB member and modify SQL for all explainable SQL to issue EXPLAIN PLAN with correct QUERYNO then process same as COMPPLAN.

*creator*

Specify the qualifier for the plan table name.

*table*

Specify the name of the secondary plan table.

*entity*

Specify the name of the plan or collection to be analyzed.

*progname*

Specify the program name (DBRM name) to be processed.

*report*

This is an optional parameter. The only allowable value is:

**ALL** Specify ALL to generate a report listing all access paths, not just the ones that have changed.

EXAMPLE JCL for the old JCL parameter processing

Example 1

The example shown below will produce a report based on preexisting *explain* data in *userid.PLAN\_TABLE*. The collection ID is DSNESPCS and the program name is DSNESM68.

```
//STEP1 EXEC PGM=PATHCHK,  
// PARM='ssid,RPTPKG,userid,,DSNESPCS,DSNESM68'  
//STEPLIB DD DSN=db2.SDSNLOAD,DISP=SHR  
// DD DSN=PATHCHK.LOAD,DISP=SHR  
//SYSPRINT DD SYSOUT=*
```

## Example 2

This next example will compare the same rows as the previous report to the same plan table. This should always compare equal.

```
//STEP3 EXEC PGM=PATHCHK,
//      PARM='ssid,COMPPKG,userid,PLAN_TABLE,DSNESPCS,DSNESM68,ALL'
//STEPLIB DD DSN=DSN610.SDSNLOAD,DISP=SHR
//      DD DSN=P390H.PDS.LOAD2,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

## Example 3

This example will compare the same rows as the previous example to a different plan table (PLAN\_TABLE2). All the access path information will be included in the SYSPRINT and SYSEXPLN reports. SYSPRINT will have a 1 in column 2 for old access path, 2 in column 2 for new access path and an asterisk (\*) for access paths that are the same. SYSEXPLN will have a separate line to identify the beginning of the old and new access path information.

```
//STEP3 EXEC PGM=PATHCHK,
//      PARM='ssid,COMPPKG,userid,PLAN_TABLE2,DSNESPCS,DSNESM68,ALL'
//STEPLIB DD DSN=DSN610.SDSNLOAD,DISP=SHR
//      DD DSN=P390H.PDS.LOAD2,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSEXPLN DD SYSOUT=*
```

## Example 4

This example will *Explain* each explainable statement in the DBRM and compare the results to the previous bind with planname TESTDBRM. The *Explain* results will be saved in PLAN\_TABLE with *applname* and *collid* set to blanks and *progrname* set to 'PATHCHK'.

```
//STEP5 EXEC PGM=PATHCHK,
/
PARM='ssid,TESTPLAN,userid,PLAN_TABLE,TESTDBRM,progrname,ALL'
//STEPLIB DD DSN=DSN610.SDSNLOAD,DISP=SHR
//      DD DSN=P390H.PDS.LOAD2,DISP=SHR
//SYSPRINT DD SYSOUT=*
//DBRMIN DD DSN=P390H.PDS.DBRMLIB,DISP=SHR
```

## Appendix E — CREATE TABLES option

SQL explain processing requires you to combine 2 items which are normally specified separately in bind processing.

1. The target plan\_table which is the current sqlid.plan\_table
2. The target tables that will be processed by the SQL statements

In bind processing, these items are specified separately. The plan\_table is qualified by the userid in the owner parameter and the application tables are qualified by the userid specified for the qualifier parameter.

There are multiple ways to achieve this:

- A. Create a plan\_table under the id used for the qualifier parameter and set current sqlid to that value before issuing the test or explain commands.
- B. Use the plan\_table from the owner parm and create alias's or synonyms to the application tables using the following spufi to generate create synonym commands. This only allows for one set of applications to be mapped to the OWNER.plan\_table at a time.
- C. Specify OPTIONS CREATE TABLES before the TEST or EXPLAIN command. This will cause Pathcheck to create the required PLAN\_TABLE and DSN\_STATEMENT\_TABLE to process the TEST and EXPLAIN commands. These tables will be dropped at the end of the process. They are created as implicit tables where the create statement does not specify a tablespace.

The following SPUFI will generate the synonyms 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 SYNONUM 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  
;
```

## Appendix F — Path Check Name substitution using BINDIN DDNAME Support

### BINDIN DD statement example

This DD statement is optional. It will generate commands from bind control cards. Path Check will parse the Bind control cards and perform a path check command for each plan or package specified in the control cards. Path Check will ignore the plan and package name in the commands and identify the plan or package to be processed from the BIND control commands. Only one command can be processed with the BINDIN controlling the plan, collection and program name.

For example the following bind control commands

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

With the following path check commands

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

Will identify 2 packages to be processed

1. collid=TESTCOLLX and program name TEST01
2. collid=TESTCOLLX and program name TEST02

As if the commands had specified

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

## Notices

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

HLS TECHNOLOGIES, INC. PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR 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. HLS may make improvements and/or changes in the product(s) and or the program(s) described in this publication at any time without notice. Information concerning non-HLS products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HLS has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HLS products. Questions on the capabilities of non-HLS products should be addressed to the suppliers of those products.

## Trademarks

IBM, DB2, MVS, ISPF, OS/390 and z/OS are trademarks of IBM Corporation. SQL Performance Analyzer (SQL/PA) is a trademark of IMSI, Inc.